



**AN INVESTIGATION OF THE DESIGN AND USE OF  
FEED-FORWARD ARTIFICIAL NEURAL NETWORKS IN  
THE CLASSIFICATION OF REMOTELY SENSED IMAGES**

*by*

**TAŞKIN KAVZOĞLU**

*MSc. Geographical Information Systems*



Thesis submitted to the University of Nottingham for  
the degree of Doctor of Philosophy

May 2001

## ABSTRACT

Artificial neural networks (ANNs) have attracted the attention of researchers in many fields, and have been used to solve a wide range of problems. In the field of remote sensing they have been used in a variety of applications, including land cover mapping, image compression, geological mapping and meteorological image classification, and have generally proved to be more powerful than conventional statistical classifiers, especially when training data are limited and the data in each class are not normally distributed.

The use of ANNs requires some critical decisions on the part of the user. These decisions, which are mainly concerned with the determinations of the components of the network structure and the parameters defined for the learning algorithm, can significantly affect the accuracy of the resulting classification. Although there are some discussions in the literature regarding the issues that affect network performance, there is no standard method or approach that is universally accepted to determine the optimum values of these parameters for a particular problem.

In this thesis, a feed-forward network structure that learns the characteristics of the training data through the backpropagation learning algorithm is employed to classify land cover features using multispectral, multitemporal, and multisensor image data. The thesis starts with a review and discussion of general principles of classification and the use of artificial neural networks. Special emphasis is put on the issue of feature selection, due to the availability of hyperspectral image data from recent sensors. The primary aims of this research are to comprehensively investigate the impact of the choice of network architecture and initial parameter estimates, and to compare a number of heuristics developed by researchers. The most effective heuristics are identified on the basis of a large number of experiments employing two real-world datasets, and the superiority of the optimum settings using the 'best' heuristics is then validated using an independent dataset. The results are found to be promising in terms of ease of design and use of ANNs, and in producing considerably higher classification accuracies than either the maximum likelihood or neural network classifiers constructed using *ad*

*hoc* design and implementation strategies. A number of conclusions are drawn and later used to generate a comprehensive set of guidelines that will facilitate the process of design and use of artificial neural networks in remote sensing image classification.

This study also explores the use of visualisation techniques in understanding the behaviour of artificial neural networks and the results produced by them. A number of visual analysis techniques are employed to examine the internal characteristics of the training data. For this purpose, a toolkit allowing the analyst to perform a variety of visualisation and analysis procedures was created using the MATLAB software package, and is available in the accompanying CD-ROM. This package was developed during the course of this research, and contains the tools used during the investigations reported in this thesis.

The contribution to knowledge of the research work reported in this thesis lies in the identification of optimal strategies for the use of ANNs in land cover classifications based on remotely sensed data. Further contributions include an in-depth analysis of feature selection methods for use with high-dimensional datasets, and the production of a MATLAB toolkit that implements the methods used in this study.

# ACKNOWLEDGEMENTS

This thesis presents the culmination of more than three years of work that would not have been successfully accomplished without the support of many individuals. First of all, I would like to express my sincere gratitude and thanks to my supervisors, Professor Paul Mather and Dr. Michael McCullagh. I am particularly grateful to Professor Paul Mather for his continuous support, help and guidance throughout the period of this study. Without all his help and suggestions, this work would not have been presented in its current form. As well as many important things in life, I learnt remote sensing from him. I am therefore highly indebted to him.

I am very grateful to my sponsor, the Turkish Government, for giving me the opportunity and financial support to pursue my MSc and PhD degrees at the University of Nottingham. I would like to also acknowledge the Ordnance Survey of Great Britain (© Crown Copyright NC/00/1004) and Logica UK Ltd for providing the datasets that enabled me to carry out this study. The contribution of Elveden Farms Ltd is also acknowledged.

I would like to convey my thanks to all the staff members of the School of Geography, Nottingham University for their co-operation and support. I am grateful to Dr. Michael Steven for his editorial comments. Thanks are also due to all my research colleagues, particularly Jasmee Jaafar, Carlos Vieira, Nurul Salmi, Heriberto Gomez, Premalatha Balan and Saleem Muhammad, who have always provided invaluable assistance and advice as well as their friendship.

Special recognition is due to my parents for their continuous support and encouragement throughout my life in particular during my education. The opportunity they afforded me enabled this work to come about.

Last but definitely not least, I send my deepest gratitude and love to my wife Şebnem for her endless patience and understanding during all those years that kept me busy, sometimes distressful and perplexed. It was from her that I found tranquillity, support and encouragement.

Finally, I am sure there are many that have not been mentioned above, but they are acknowledged and certainly not forgotten.

*To*

*my wife, Sebnem,*

*and my little son, Berkay.*

# TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>i</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>iii</b>
<b>TABLE OF CONTENTS.....</b>	<b>v</b>
<b>LIST OF FIGURES .....</b>	<b>ix</b>
<b>LIST OF TABLES .....</b>	<b>xiii</b>
<b>CHAPTER I: INTRODUCTION.....</b>	<b>1</b>
1.1 Overview .....	1
1.2 Statement of Problem .....	3
1.3 Research Aims and Objectives.....	6
1.4 Main Contribution to Research .....	8
1.5 Overview of the Thesis.....	8
<b>CHAPTER II: CLASSIFICATION .....</b>	<b>11</b>
2.1 Introduction .....	11
2.2 Definition.....	12
2.3 Philosophy of Classification.....	14
2.4 Taxonomy of Classification Techniques.....	17
2.4.1 Unsupervised Classification.....	20
2.4.2 Supervised Classification.....	22
2.4.2.1 The Parallelepiped Classifier.....	24
2.4.2.2 The Minimum Distance Classifier.....	27
2.4.2.3 The Maximum Likelihood Classifier .....	29
2.5 Artificial Neural Networks.....	32
2.6 Incorporation of Spatial Information.....	34
2.6.1 Texture and Context.....	35
2.6.2 Using Ancillary Data .....	38
2.7 Classification Accuracy Assessment.....	39
2.7.1 Overall Accuracy .....	40
2.7.2 Kappa Coefficient .....	43
2.7.3 Accuracy Maps .....	46
2.8 Summary .....	49

<b>CHAPTER III: ARTIFICIAL NEURAL NETWORKS .....</b>	<b>51</b>
3.1 Introduction .....	51
3.2 History of Artificial Neural Networks.....	54
3.3 Network Structure .....	56
3.4 Learning Algorithms .....	58
3.5 Activation (Transfer) Functions .....	64
3.6 Encoding.....	67
3.7 Generalisation.....	70
3.8 Pruning Algorithms .....	72
3.9 Problems in the Use of Artificial Neural Networks .....	76
3.10 Summary .....	85
<b>CHAPTER IV: VISUALISATION OF HIGH-DIMENSIONAL DATA .....</b>	<b>88</b>
4.1 Introduction .....	88
4.2 Graphical Visualisation Techniques.....	89
4.2.1 Chernoff Faces .....	89
4.2.2 Parallel Coordinate Plots.....	92
4.2.3 Andrews' Plots.....	94
4.3 Projection Methods.....	97
4.3.1 Linear Projection Techniques .....	99
4.3.1.1 Principal Components Analysis .....	99
4.3.1.2 Factor Analysis.....	106
4.3.2 Nonlinear Projection Techniques.....	108
4.3.2.1 Multidimensional Scaling.....	108
4.3.2.2 Sammon's Nonlinear Mapping.....	109
4.3.2.3 The Self-Organising Map.....	113
4.4 Summary and Conclusions.....	117
<b>CHAPTER V: FEATURE SELECTION.....</b>	<b>119</b>
5.1 Introduction .....	119
5.2 Test Sites, Data and Analysis Tools.....	121
5.2.1 Test Site 1 .....	121
5.2.2 Test Site 2 .....	124
5.3 Filters and Wrappers .....	129
5.4 Class Separability Indices .....	131

5.4.1	Divergence and Transformed Divergence Indices.....	131
5.4.2	Bhattacharyya Distance .....	142
5.4.3	Jeffries-Matusita Distance .....	145
5.5	Statistical Tests.....	149
5.5.1	Hotelling's $T^2$ .....	149
5.5.2	Wilks' $\Lambda$ Criterion .....	155
5.6	Mahalanobis Distance Classifier .....	160
5.7	Search Techniques.....	169
5.7.1	Sequential Forward Selection and Sequential Backward Selection	171
5.7.2	Branch-and-Bound Search Method.....	173
5.7.3	Genetic Algorithms.....	174
5.8	Using Artificial Neural Networks for Feature Selection.....	179
5.9	Using Pure Pixels for Delineation of Land-Cover Classes in Test Site 2 ....	181
5.10	Conclusions .....	188
<b>CHAPTER VI: ISSUES RELATED TO THE DESIGN AND USE OF ARTIFICIAL NEURAL NETWORKS .....</b>		<b>195</b>
6.1	Introduction .....	195
6.2	Number of Input Nodes.....	196
6.3	Number of Hidden Nodes.....	204
6.4	Number of Output Nodes .....	211
6.5	Learning Rate and Momentum.....	212
6.6	Initial Weight Range .....	221
6.7	Number of Training Samples .....	230
6.8	Stopping Criterion for the Training Process.....	235
6.9	Output Encoding.....	239
6.10	Validating the Conclusions .....	242
6.10.1	Test Site and Data .....	242
6.10.2	Optimum Setting for Artificial Neural Networks .....	244
6.10.3	A Worst-Case Scenario for ANN Design and Use .....	247
6.11	Summary .....	249
<b>CHAPTER VII: CONCLUSIONS.....</b>		<b>251</b>
7.1	Introduction .....	251
7.2	Summary of the Thesis.....	252

7.3	Conclusions .....	253
7.4	Guidelines for the Effective Use of Artificial Neural Networks.....	256
7.5	Future Work and Recommendations.....	258
7.6	Final Remarks.....	259
	<b>REFERENCES .....</b>	<b>260</b>
	<b>APPENDIX A: VISUALISATION TOOLKIT FOR ANALYSING ARTIFICIAL NEURAL NETWORKS .....</b>	<b>278</b>
A.1	Introduction .....	278
A.2	Installing and Running the Toolkit.....	280
A.3	Prior Analyses .....	282
	A.3.1 Preparing Pattern Files .....	283
	A.3.2 Graphical Analysis.....	286
	A.3.3 Reducing the Dimensions of the Dataset .....	286
	A.3.4 Visualising and Cleaning the Data.....	290
A.4	Training and Testing.....	292
	A.4.1 Network File Creation.....	293
	A.4.2 Preparation and Running Batch Files.....	293
	A.4.3 Training and Testing Artificial Neural Networks .....	294
	A.4.4 Displaying Classification Results for a Test Image .....	295
	A.4.5 Classifying Images Using the Mahalanobis Distance and Maximum Likelihood Classifiers .....	296
A.5	Posterior Analyses .....	297
	A.5.1 Analysing Individual Pixels.....	298
	A.5.2 Displaying Pixels Selected for Pattern Files .....	300
	A.5.3 Analysis of Network Weights .....	300
	A.5.4 Accuracy Assessment .....	302
	A.5.5 Reliability Analysis.....	304
	A.5.6 Creating IDRISI Image Files .....	305
	A.5.7 Making a GIF Animation of a Training Process.....	305
A.6	Summary.....	306
	<b>APPENDIX B: LIST OF PUBLICATIONS .....</b>	<b>307</b>

# LIST OF FIGURES

Figure 1.1 A simple four-layer fully connected feed-forward neural network .....	3
Figure 2.1 Schematic diagram for parallelepiped classification method .....	25
Figure 2.2 More accurate definition of decision regions using stepped borders...	26
Figure 2.3 Schematic diagram for minimum distance classifier .....	28
Figure 2.4 Schematic diagram for maximum likelihood classifier .....	31
Figure 2.5 A four-layer feed-forward neural network.....	34
Figure 2.6 ANN output activation levels presented with colour tones .....	47
Figure 2.7 ANN output activation levels presented as a grey scale .....	48
Figure 2.8 Output activation levels using another colour spectrum (hot).....	49
Figure 3.1 A neural network processing node.....	56
Figure 3.2 A simple three layer feed-forward neural network structure .....	57
Figure 3.3 Major neural network learning algorithms. ....	58
Figure 3.4 The descent in weight space. ....	61
Figure 3.5 Typical error surface with local minima.....	62
Figure 3.6 The sigmoid activation function .....	65
Figure 3.7 The ‘tanh’ activation function.....	66
Figure 3.8 Pruning artificial neural networks.....	73
Figure 3.9 Effect of the learning set size on the error rate .....	78
Figure 3.10 Error rates of learning and test sets for number of hidden units.....	81
Figure 4.1 Chernoff faces.....	91
Figure 4.2 Parallel coordinate representation of two $n$ -dimensional points.....	92
Figure 4.3 Parallel coordinate representation of Elveden dataset .....	93
Figure 4.4 Andrews’ diagram of eight five-dimensional data vectors.....	96
Figure 4.5 Andrews’ diagram for Elveden dataset.....	96
Figure 4.6 The difference between a linear and nonlinear projection method.....	98
Figure 4.7 Scree diagram of principal components for Elveden dataset.....	101
Figure 4.8 The result of PCA for the Elveden dataset.....	102
Figure 4.9 The basic structure of a linear PCA network.....	103
Figure 4.10 Architecture of a five-layer bottlenecked AANN.....	105
Figure 4.11 Sammon’s nonlinear mapping algorithm for Elveden data .....	111
Figure 4.12 Kohonen’s Self-Organizing Map.....	115
Figure 5.1 First area of interest near Littleport.....	123

Figure 5.2 Ground truth data for the first test site.....	123
Figure 5.3 Location of the second area of interest near Thetford .....	125
Figure 5.4 Ground truth data for second test site .....	126
Figure 5.5 SIR-C SAR image containing the study area.....	126
Figure 5.6 Two approaches to feature subset selection.....	129
Figure 5.7 ANN classification of the test image for the solution found by SFS based on divergence for the first test site.....	136
Figure 5.8 Spatial pattern of output activations for the first test site .....	137
Figure 5.9 ANN classification of the test image for the solution found by SFS based on divergence for the second site.....	141
Figure 5.10 Spatial pattern of output activations for the second site .....	141
Figure 5.11 ANN classification result for the solution found by SFS based on Hotelling's $T^2$ for the first test site .....	151
Figure 5.12 Spatial pattern of output activations for the first test site .....	152
Figure 5.13 ANN classification result for the solution found by SFS based on Hotelling's $T^2$ for the second test site .....	154
Figure 5.14 Spatial pattern of output activations for the second site .....	154
Figure 5.15 ANN classification result for the solution found by SFS based on Wilks' $\Lambda$ for the first test site.....	157
Figure 5.16 Spatial pattern of output activations for the first site.....	157
Figure 5.17 ANN classification result for the solution found by SFS based on Wilks' $\Lambda$ for the second test site .....	159
Figure 5.18 Spatial pattern of output activations for the second site .....	159
Figure 5.19 MDC classification results for the first site .....	165
Figure 5.20 Artificial neural network classification results for the first site.....	165
Figure 5.21 MDC classification results for the second site.....	168
Figure 5.22 Artificial neural network classification results for the second site ..	168
Figure 5.23 The solution space of the feature subset selection problem.....	169
Figure 5.24 A taxonomy of feature selection algorithms.....	171
Figure 5.25 Process of Genetic Algorithms .....	176
Figure 5.26 Example of single-point crossover.....	177
Figure 5.27 Example of multi-point crossover.....	177
Figure 5.28 Bit mutation on the third bit.....	178
Figure 5.29 Ground control data including 75 fields for Thetford site .....	182

Figure 5.30 Result of ANN classification of whole image using the network trained for the best divergence solution .....	183
Figure 5.31 Spatial pattern of output activations presented in grey scale .....	183
Figure 5.32 Spatial pattern of output activations lower than 0.5 .....	184
Figure 5.33 Training pixels .....	186
Figure 5.34 Testing pixels .....	186
Figure 5.35 All the pixels .....	187
Figure 5.36 ANN evaluation of solutions found by SFS and GA procedures for the first dataset using network structure of 8-15-7 .....	190
Figure 5.37 ANN evaluation of solutions found by SFS and GA procedures for the second dataset using network structure of 8-15-7 .....	191
Figure 6.1 Changes in MSE using the Noncontributing Units pruning method .	198
Figure 6.2 Overall accuracy versus bands eliminated by the Noncontributing Units method at each pruning step .....	198
Figure 6.3 Changes in MSE using the Skeletonization node pruning method....	199
Figure 6.4 Overall accuracy versus bands eliminated by the Skeletonization method at each pruning step .....	199
Figure 6.5 Relationship between divergence and number of input bands.....	200
Figure 6.6 CPU time as a function of number of input nodes.....	201
Figure 6.7 Change in overall accuracy with respect to the number of inputs and number of iterations used for training .....	203
Figure 6.8 Effect of number of hidden nodes on classification accuracy for the first test site.....	209
Figure 6.9 Effect of number of hidden nodes on classification accuracy for the second test site.....	209
Figure 6.10 Comparison of the performances of the heuristics for the first and the second test datasets .....	210
Figure 6.11 Comparison of using constant and varied learning rates .....	217
Figure 6.12 Training using 2,100 samples at 0.1 and 0.9 momentum rates.....	217
Figure 6.13 Training using 175 samples at 0.1 and 0.9 momentum rates.....	218
Figure 6.14 Overall accuracies produced for different learning rate configurations for the first dataset.....	219
Figure 6.15 Overall accuracies produced for different learning rate configurations for the second dataset .....	219

Figure 6.16 Overall accuracies produced for different configurations of the learning rate and the momentum for the first dataset.....	220
Figure 6.17 Overall accuracies produced for different configurations of the learning rate and the momentum for the second dataset .....	221
Figure 6.18 Variations in the overall accuracy depending on the initial weight range for the Littleport dataset.....	227
Figure 6.19 Variations in the overall accuracy depending on the initial weight range for the Thetford dataset.....	229
Figure 6.20 Effect of number of training samples on classification accuracy for two datasets.....	234
Figure 6.21 The effect of employing different output encoding methods on the classification accuracy for the first dataset.....	241
Figure 6.22 The effect of employing different output encoding methods on the classification accuracy for the second dataset .....	241
Figure 6.23 Ground truth for Littleport site for the crop season of 2000.....	243
Figure A.1 Menus of the toolkit covering all the analyses.....	281
Figure A.2 Items listed under Data Analysis menu heading .....	283
Figure A.3 Menus used for training and test pattern file creation.....	283
Figure A.4 Illustration of rectangular area selection.....	284
Figure A.5 Visualising classification results in three-dimensional form .....	291
Figure A.6 New boundary determination for forest class .....	292
Figure A.7 Items listed under the Classification menu heading.....	292
Figure A.8 Items listed under the Evaluate menu heading .....	297
Figure A.9 Locating the pixel closest to the clicked point on the figure .....	298
Figure A.10 Location of the selected pixel on ground truth image.....	299
Figure A.11 Location of the selected pixel on the satellite image .....	299
Figure A.12 Analysing the characteristics of a pixel using histograms .....	300
Figure A.13 Pixels selected for ground truth .....	301
Figure A.14 Analysis of the weights in the network.....	301
Figure A.15 Accuracy assessment using a histogram .....	302
Figure A.16 Accuracy assessment with contingency matrix .....	303
Figure A.17 Reliability analysis for each pixel used in testing.....	304

## LIST OF TABLES

Table 2.1 Example confusion matrix .....	42
Table 3.1 Types of decision regions in the input data space.....	64
Table 4.1 Crime data of US cities .....	91
Table 5.1 Detailed information for the images used for the first test site .....	122
Table 5.2 Detailed information for the training and test files for the first site .....	124
Table 5.3 Detailed information for the images used in the second test area .....	125
Table 5.4 Separability comparison of raw and filtered SIR-C images.....	127
Table 5.5 Cross-correlation between SIR-C polarisation images .....	128
Table 5.6 ANN classification results of the solution attained by divergence using SFS for the first dataset .....	134
Table 5.7 ANN classification results of the best band combination found by divergence using GA for the first dataset.....	135
Table 5.8 ANN classification results of the best band combination found by divergence using SFS for the second dataset .....	137
Table 5.9 ANN classification results of the best band combination found by divergence using GA for the second dataset .....	138
Table 5.10 ANN classification results of the best band found by transformed divergence using SFS for the first dataset.....	139
Table 5.11 ANN classification results of the best band found by transformed divergence using GA for the first dataset .....	139
Table 5.12 ANN classification results of the best band found by transformed divergence using GA for the second dataset.....	140
Table 5.13 ANN classification results for the band combination attained by the Bhattacharyya distance using SFS for the first dataset .....	143
Table 5.14 ANN classification results for the band combination attained by the Bhattacharyya distance using GA for the first dataset.....	144
Table 5.15 ANN classification results for the band combination attained by the Bhattacharyya distance using SFS for the second dataset .....	144
Table 5.16 ANN classification results for the band combination attained by the Bhattacharyya distance using GA for the second dataset .....	145
Table 5.17 ANN classification results for the band combination attained by the Jeffries-Matusita distance using SFS for the first dataset .....	147

Table 5.18 ANN classification results for the band combination attained by the Jeffries-Matusita distance using GA for the first dataset.....	147
Table 5.19 ANN classification results for the band combination attained by the Jeffries-Matusita distance using SFS for the second dataset .....	148
Table 5.20 ANN classification results for the band combination attained by the Jeffries-Matusita distance using GA for the second dataset .....	148
Table 5.21 ANN classification results for the band combination attained by Hotelling's $T^2$ using SFS for the first dataset .....	150
Table 5.22 ANN classification results of the best band combination found by Hotelling's $T^2$ using GA for the first dataset .....	152
Table 5.23 ANN classification results for the band combination found by Hotelling's $T^2$ using SFS for the second dataset.....	153
Table 5.24 ANN classification results of the best band combination found by Hotelling's $T^2$ using GA for the second dataset.....	153
Table 5.25 ANN classification results for the band combination attained by Wilks' $\Lambda$ using SFS for the first dataset .....	156
Table 5.26 ANN classification results of the best band combination found by Wilks' $\Lambda$ using GA for the first dataset .....	158
Table 5.27 ANN classification results of the best band combination found by Wilks' $\Lambda$ using SFS and GA for the second dataset .....	158
Table 5.28 ANN classification results of the best solution found by MDC based on average accuracy using SFS for the first dataset .....	162
Table 5.29 ANN classification results of the best solution found by MDC based on overall accuracy using SFS for the first dataset.....	162
Table 5.30 ANN classification results of the best band combination found by MDC based on quality measure using SFS for the first dataset .....	163
Table 5.31 ANN classification results of the best band combination found by MDC based on average accuracy using GA for the first dataset .....	163
Table 5.32 ANN classification results of the best band combination found by MDC based on overall accuracy using GA for the first dataset .....	164
Table 5.33 ANN classification results of the best band combination found by MDC based on quality measure using GA for the first dataset .....	164
Table 5.34 ANN classification results of the combination found by MDC based on overall and average accuracies using SFS for the second dataset	166

Table 5.35 ANN classification results of the combination found by MDC based on quality measure using SFS for the second dataset.....	166
Table 5.36 ANN classification results of the combination found by MDC based on overall and average accuracies using GA for the second dataset	167
Table 5.37 ANN classification results of the combination found by MDC based on quality measure using GA for the second dataset.....	167
Table 5.38 ANN classification results of the best band combination found by divergence using sequential forward selection technique.....	182
Table 5.39 ANN classification of five test datasets using 8-10-7 net structure ..	185
Table 5.40 Solutions found by search techniques using nine separability measures for the first dataset.....	189
Table 5.41 Solutions found by search techniques using eight separability measures for the second dataset.....	189
Table 6.1 Analysis of time required to train neural networks having number of inputs ranging from 5 to 24.....	202
Table 6.2 Heuristics proposed by researchers to compute the optimum number of hidden layer nodes.....	206
Table 6.3 Heuristics for optimum learning rate and momentum term .....	213
Table 6.4 Initial weight ranges used by some researchers .....	226
Table 6.5 Results of three major training termination methods for datasets.....	238
Table 6.6 Optimum setting of network structure and learning parameters .....	245
Table 6.7 ANN results obtained from the configurations given in Table 6.6 .....	246
Table 6.8 Configuration of the network and the learning algorithm for the worst-case scenario .....	248
Table 6.9 ANN results obtained for the configurations given in Table 6.8 .....	248

# CHAPTER I

## INTRODUCTION

### 1.1 Overview

Land cover mapping is an important economic activity. At global scales, knowledge of land cover is needed for the application of Global Climate Models (GCM). At regional scales, governments seek to monitor crop production and the spatial and temporal distributions of their natural resources, while at local scales farmers may wish to use modern technology to assess the rate of growth of crops in order better to manage their use of fertilisers and irrigation water.

Remotely sensed data are now widely used to provide the information required at these different scales. New satellite-borne instruments carried by platforms such as Terra and Landsat-7 provide multispectral data in the visible and near infrared regions of the spectrum at resolutions ranging from 1km to 30m. Local areas are imaged by the IKONOS sensors at a resolution of 4m in multispectral mode. There is no doubt that the range and quality of data (measured by spatial and radiometric resolution) will continue to improve in the coming years.

The work reported in this thesis focuses on the regional scale, at which crop monitoring and crop inventories are the main concern. Since the launch of Landsat-1 (ERTS-1) in 1972, this field of study has attracted considerable interest and substantial experience, both theoretical and practical, has been accumulated.

In recent years, as Geographical Information Systems (GIS) databases have been built up, spatial data in the form of digital maps and digital elevation models have become more widely available to augment satellite image data. This in turn has meant that information processing techniques have become more sophisticated.

During the past 20 years, statistical classification methods, such as the minimum distance and the maximum likelihood classifiers, have been widely used. However, these methods have their restrictions, related particularly to the distribution assumptions and limitations in the input data types. In the past decade, the artificial neural network approach, theoretically a more sophisticated and robust method of image classification, has been introduced and employed in remote sensing applications. Although this approach has been used in a wide range of scientific disciplines for a variety of applications since the early 1980s, their use in remote sensing area is relatively new, dating only from the early 1990s. Studies have shown that artificial neural networks (ANNs) are more robust than conventional statistical methods in terms of producing classification results with higher accuracies and requiring fewer training samples. The most important characteristic of ANNs is perhaps their non-parametric nature, assuming no *a priori* knowledge, particularly of the frequency distribution of the data. Because of their adaptability and their ability to produce high-quality results, the use of artificial neural networks has spread in the scientific community at large, leading to an increasing amount of research in the remote sensing field. Currently, there are a number of journals devoted to neural network research and a considerable number of textbooks published illustrating their applications in a diversity of fields.

One of the earliest studies discussing the use of artificial intelligence techniques for remote sensing data was carried out by Estes *et al.* (1986) who suggested the use of such techniques for intelligent onboard processing, advanced database interrogation, and automated analysis of multispectral imagery. Researchers have applied neural network classifiers to remotely sensed data for several different purposes. For example; Benediktsson *et al.* (1990), Kanellopoulos *et al.* (1992), Paola and Schowengerdt (1995a), and Bruzzone *et al.* (1997) compared the results of maximum likelihood classification, which is the most elaborate statistical

method of image classification, with artificial neural network classifiers, and found that ANNs can produce more accurate results than a maximum likelihood classifier. The use of radar images in classification was evaluated by Hara *et al.* (1994) and Chen *et al.* (1996); multispectral data classification using ANN techniques was reported by Bischof *et al.* (1992), Heermann and Khazenie (1992), Civco and Waug (1994), and Abuelgasim *et al.* (1996), among others. Issues related to the accuracy of ANN classifications are discussed by Paola and Schowengerdt (1997), Kanellopoulos and Wilkinson (1997), and Foody (1999). Articles by Paola and Schowengerdt (1995b) and Atkinson and Tatnall (1997) review the use of artificial neural networks for remote sensing data.

**1.2 Statement of Problem**

Although various types of neural network models have been developed, the most widely used model in the literature is the feed-forward neural network, also known as the multilayer perceptron. In feed-forward neural networks, there are three types of layers consisting of processing nodes that are fully interconnected to each other, except that there are no interconnections between nodes within the same layer. These layers are the input, hidden and output layers. A feed-forward neural network (Figure 1.1) usually comprises one input layer, one or two hidden layers and one output layer. The input layer nodes correspond to individual data sources, such as the Landsat TM bands. Hidden layers are used for computations, and the values associated with each node are estimated from the sum of the

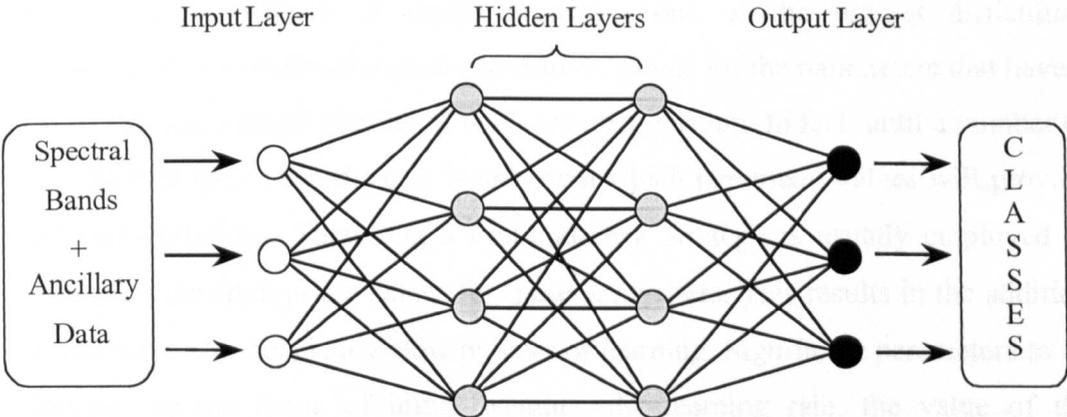


Figure 1.1 A simple four-layer fully connected feed-forward neural network.

multiplications between input node values and weights of the links connected to that node. The output layer includes a set of codes to represent the classes to be recognised. The most popular coding method for feed-forward neural networks is that the value of the output node corresponding to a specific class is assigned to 1, and others to 0. For example, the second output class at a four-node output layer is represented by 0 1 0 0. All inter-node connections have associated weights, which are generally randomised at the beginning of the training process.

Despite their promising prospects, artificial neural networks (ANNs) suffer from several deficiencies, basically related to the problems encountered in their design and implementation. These deficiencies restrict their general acceptability, particularly in the remote sensing community. From the design perspective, the specification of the size of the hidden layer(s) is critical for the network's capabilities of learning and generalisation. However, the sizes of all layers in the network are of importance as components of the ANN structure. As the size of the input layer is often equal to number of features on which the classification is based, and the size of the output layer usually corresponds to the number of output classes, the hidden layer or layers are subject to adjustments in size. Despite the fact that the effects of employing too small or too large network structures are known in a general sense, the exact nature of the impact of the sizes of the hidden layers on network performance has not been fully investigated. Although several heuristics have been proposed, none is universally accepted for estimating the optimal number of hidden layer nodes for a particular problem.

In the implementation of neural networks, one of the biggest difficulties encountered is to define the most appropriate values for the parameters that have a major influence on the success of the learning algorithm. In fact, until a number of experiments have been done, it is unknown which parameter values will provide optimum solutions. Therefore, a trial-and-error strategy is usually employed to determine the appropriate values for these parameters. This results in the addition of more time to the already slow process of learning. Significant parameters to be defined are the range of initial weights, the learning rate, the value of the momentum term, and the number of training phase iterations, all of which are related to the question of when and how to stop the training process. Furthermore,

specific encoding techniques are required for the representation of both input and output information.

An appropriate number of training samples has to be measured to ensure a correct presentation of the classification problem to the network. The number of training samples required is mainly dependent on the network structure and the level of complexity introduced by the problem. In addition to these uncertainties, it is not known exactly how ANNs learn particular problems and apply the extracted rules to new cases, or how conclusions can be drawn from the trained networks. As a consequence, artificial neural networks are generally called 'black-box' methods.

The issues noted above have been pointed out by several researchers. For instance, Paola and Schowengerdt (1995b) provide detailed information about the issues to be carefully considered in the design and use of ANNs, as well as reviewing their use for the classification of remotely sensed data. Wilkinson (1997) lists the open questions in neurocomputing regarding Earth observation and also discusses several neural network issues, including the simplicity of training algorithms used, the 'overfitting' problem and their susceptibility to chaotic behaviour under the heading of "*Problems in Using Neural Networks*". Foody (1999), on the other hand, discusses more specific factors, such as the number of hidden units and layers, the quality of the training data and the training time, under the heading of "*Limitations of the Conventional Neural Network Approach*". He describes them as being factors that the analyst may have control over and that strongly influence network performance, especially in terms of speed and accuracy. He also states that a major limitation associated with artificial neural networks is that they are semantically poor. In other words, while an artificial neural network may be able to perform a certain task it is difficult to explain the results or gain any understanding about how the result was achieved.

Even though particular problems have been identified, a complete study has not yet been carried out to determine the nature of the problems and their effects on network performance. Motivated by the above mentioned studies, the present study was conducted in order to gain some insights into understanding the behaviour of artificial neural networks, thus facilitating the steps of network

design and parameter setting. It is hoped that this research will help to disprove to some extent the statement that artificial neural networks are semantically poor.

### **1.3 Research Aims and Objectives**

The work reported in this thesis focuses on the use of multilayer perceptrons using the backpropagation learning technique. It is noted in section 1.1 that the use of such networks in the processing of remotely sensed data is widespread. However, a number of issues that inhibit the successful use of ANNs in image classification have been identified (Paola and Schowengerdt, 1995b; Wilkinson, 1997; Foody and Arora, 1997; Foody, 1999). The main problems that have been recognised in the literature include:

- Specification of network architecture (number of nodes in the input, output, and hidden layers; number of hidden layers),
- Specification of the values of parameters that relate to the learning process (initial weights, learning rate, momentum term, and number of iterations),
- Determination of the optimum number and nature of samples used in training the network.

Although individual studies have highlighted specific problems, such as the influence of the initial weight configuration on the results produced by the network, or have discussed these problems as a whole, no research study to date has attempted to consider all aspects of network design and use in the context of the classification of remotely sensed images. The primary aim of this research, therefore, is to investigate the nature of the issues reported to have significant influence on the performance of the artificial neural network classifier.

The experiments reported in this thesis were mainly implemented to achieve the following objectives, addressing a variety of issues that are extremely important for successful applications of artificial neural networks:

- 1) To critically evaluate conventional statistical classifiers and the techniques that are used to evaluate classification results. An alternative way of portraying the results is investigated, in particular to present variations in the spatial domain and the level of confidence for pixel class assignments.
- 2) To conduct an extensive review of the theory and implementation of artificial neural networks in remote sensing image classification and to give details of the major problems encountered in their use.
- 3) To investigate the use of scientific visualisation techniques for exploring the internal structure of high-dimensional data, and their suitability for presenting remotely sensed image data employed in neural network processing.
- 4) To make a comprehensive evaluation of the techniques of feature selection that are utilised to reduce the dimensions of the datasets. As the volume of multispectral, multitemporal and multisensor data continues to increase, the most appropriate set of inputs often needs to be selected.
- 5) For the main objective of this study, a large number of experiments were carried out to investigate the nature and the effect of the factors that have significant influence on the network performance. The results are used to construct guidelines for the efficient and effective use of neural networks. These guidelines will provide the new or inexperienced user with clear instructions on the use of ANNs in image classification, and will also summarise the experience of previous users.
- 6) To develop a toolkit to perform the major tasks required in the classification of remote sensing images using neural networks. The toolkit is also used to apply data analysis and scientific visualisation techniques with the aim of understanding each process performed. As there is no software package available to date for this purpose, such a toolkit is extremely useful for performing classification tasks and visualising the data and results.

## 1.4 Main Contribution to Research

The contributions made in this study can be divided into two main categories: practical and theoretical contributions. The practical contribution is a visualisation toolkit prepared in MATLAB (version 5.3). It is a menu-driven program that includes a number of data analysis and visualisation tools. The main aim of generating such a toolkit was, to a certain extent, to fulfil the requirements of artificial neural network users in the remote sensing field, and perform the necessary analyses required in this study. Unfortunately, no complete software package allowing applications of a variety of visualisation techniques and neural network classifiers for the analysis and the classification of land cover data is yet available for remote sensing researchers. The toolkit developed in this study has specific menus devoted to *a priori* analyses (mainly of training data), as well as *a posteriori* analyses, largely related to the evaluation of ANN results.

The theoretical contribution of this research is to examine in detail the factors that influence the performance of ANNs in image classification. It is certainly one of the user's expectations that the use of artificial neural networks should be easy to use by means of understanding each step taken. Therefore, some guidelines with sound foundations are required, describing the possible effects of various configurations of the network structure and the parameter settings. Through experimentation, a number of general guidelines have been established, and these are presented.

## 1.5 Overview of the Thesis

The thesis consists of seven chapters, including this introductory chapter in order to achieve the aims and objectives defined above. The early chapters mainly provide background information about the theory of classification, the fundamentals of artificial neural network models, and an overview of visualisation of high-dimensional data. In other words, they describe the techniques that will be employed in subsequent chapters. The following chapters include discussions and reports of experiments related to the problem of the design and use of artificial neural networks. A brief summary of each chapter is provided below:

- **Chapter II** is concerned with the theory of classification with an emphasis on its philosophy. As well as describing unsupervised and supervised classification techniques, general knowledge on the incorporation of spatial information in classification is presented. In the last part of the chapter, the methodologies used to assess classification accuracy are described, introducing the concept of accuracy maps that provide information about the spatial distribution of classification error.
- **Chapter III** provides a summary of the theory of artificial neural networks. The principles of the backpropagation learning algorithm are presented in detail, as this particular algorithm is employed in all ANN applications performed in this research. A special section is devoted to the problems encountered in the use of artificial neural networks in remote sensing. These problems will be the main concern of later chapters dealing with the analysis of the effects of varying ANN parameters.
- **Chapter IV** presents an overview of the techniques used to visualise high-dimensional data (or multivariate data). After a general introduction to the topic, several simple graphical visualisation techniques are described and examples using real-world data are presented. Projection techniques, which can be either linear or nonlinear, are also discussed and the discussion is supported by simple applications. This chapter is intended to provide a guide to the fundamentals of scientific visualisation methods for high-dimensional data. These methods will be used later to understand the characteristics of training data and the behaviour of artificial neural networks during training.
- **Chapter V** is concerned with the use of feature selection techniques to choose the most appropriate number of inputs for a particular classification problem. In this chapter, two datasets employed in this research are described. The most popular class separability indices, namely the divergence, transformed divergence, the Bhattacharyya distance and the Jeffries-Matusita distance, and statistical tests (Hotelling's  $T^2$  and Wilks'  $\Lambda$ ), used for feature selection are described in detail. Conventional

techniques employed to search for the optimum solution without evaluating all possible solutions are outlined. The effectiveness of the feature selection techniques are tested and evaluated for the two datasets. In the last part of the chapter, the performance of a neural network is evaluated for these datasets including pure pixels as opposed to mixed pixels.

- **Chapter VI** concentrates on the problems encountered in the design and use of artificial neural networks, which is the primary aim of this study. Whilst the design of ANNs is mainly related to the sizes of the network layers, the effective use of ANNs is associated with the selection of appropriate rates for the learning parameters. A good network solution can thus be obtained. After providing an extensive literature review for these issues, discussions and implementations are presented. Critical reviews and comparisons are carried out for the heuristics (or rules of thumb) recommended by researchers. The results are used to set out guidelines for users to be able to apply neural networks effectively, knowing the nature of the parameters to be defined.
- **Chapter VII** presents the conclusions drawn from this research. It specifically comprises a number of guidelines for the use of artificial neural networks extracted from the results produced and experience gained during the research. Finally, this chapter gives suggestions for further research that could be conducted on this specific research topic.

In addition to the chapters outlined above, **Appendix A** describes the visualisation toolkit prepared to perform prior and posterior analyses as well as to run the artificial neural network simulator, SNNS (Stuttgart Neural Network Simulator). It was developed using MATLAB software package to analyse and visualise the data and neural network results using scientific visualisation techniques. It is a menu-driven toolkit facilitating a number of procedures from individual programs written in MATLAB and Turbo C++. Note that since the toolkit is not designed for commercial use, it has limitations in terms of the size and format of the datasets. The appendix is written as a guide to describe the use of the toolkit.

## CHAPTER II

# CLASSIFICATION

### 2.1 Introduction

Classification is a process of identification that is addressed and used in all scientific disciplines as a way of comprehending and ordering a mass of data. It can be viewed as the process of converting raw data to categorised meaningful information. It is, in fact, a fundamental and everyday process carried out by humans. It is, for example, practised when recognising somebody we know, or looking at a group of objects containing any type of pattern. It is also practised in communication where general terms are involved. It is unimaginable to restrict languages to proper names only; that would make communication extremely difficult. Consequently, it can be stated that classification is one of the basic tools we use in dealing with the world around us. As stated by (Harvey, 1969, p. 326), classification is a higher level intellectual activity necessary to our understanding of nature. Overall, classification is a basic process we perform instinctively, and we thus give meaning to a vast amount of information existing around us.

Classification of land cover features from remotely sensed image data has been one of the main applications in the remote sensing field. It is an important and difficult task, since such images are high-dimensional and complex in nature. As the number of categories and the amount of data involved increases, so does the

complexity of the classification problem because it becomes more difficult to determine the characteristics of the categories and allocate a pixel to one of the categories. The highest classification accuracy to be produced is, therefore, usually stated to be around 80%. In order to increase this figure, two key factors must be considered. One is the use of representative datasets and employing more powerful classification techniques, such as artificial neural networks. Another factor with a positive effect on the classification accuracy is to incorporate spatial information, such as texture and context. Incorporating such information may result in a considerable increase in the classification accuracy.

The classification process has two main stages. In the first stage, the number and nature of the categories are determined, whilst in the second stage every unknown element is assigned to one of the categories according to its level of resemblance (or similarity). These stages are often called classification and identification, respectively. In the context of remote sensing, the categories could be land cover features or cloud types, and the assignment to one of the categories is carried out by assigning numerical labels, corresponding to the classes, to individual pixels. Hence, for a researcher working in the remote sensing field, classification basically means determining the class membership of each pixel in an image by comparing the characteristics of that pixel to those of known categories.

## **2.2 Definition**

Classification has been defined by many scientists in different fields of study with a wide diversity of meanings (i.e. with many ambiguities), which sometimes causes confusion. Some of these definitions are given as follows:

Classification is the ordering of organisms (or objects) into groups (or sets) on the basis of their relationships. The term relationship simply indicates the resemblance or overall similarity as judged by the characters of the organism without any indication as to their relationship by ancestry.

(Sneath and Sokal, 1973, p.3)

<sup>1</sup>Classification means a way of grouping objects, on the basis of some relationship between them. <sup>2</sup>The term classification refers to the placing of objects into groups in such a way that the members of the groups bear a closer relationship to other members of the same group than they do to members of other groups.

(Pankhurst, 1991, p.1<sup>1</sup>, p.44<sup>2</sup>)

<sup>1</sup>Classification is the basic procedure by which we impose some order and coherence upon the vast inflow of information from the real world. By sense-perception data into classes or sets we transform a mass of unwidely information so that it may be more easily comprehended and more easily manipulated. <sup>2</sup>Classification is essentially a means to a given end, a filter through which we transform sense-perception data for a given purpose.

(Harvey, 1969, p.326<sup>1</sup>, p.348<sup>2</sup>)

Classification analysis addresses itself to the problem of assigning an object to one of a number of possible groups on the basis of observations made on the object.

(James, 1985, p.3)

The classification process may be considered as a form of pattern recognition, that is, the identification of the pattern associated with each pixel position in an image in terms of the characteristics of the objects or materials that are present at the corresponding point on the Earth's surface.

(Mather, 1999a, p.167)

When the above definitions are considered, it can be seen that some of them are specifically associated with the author's field. For example, Sneath and Sokal (1973) consider classification in the context of biological science, and Mather (1999a) makes the definition specifically related to remote sensing.

According to James (1985, p.1), 'each time a discipline has re-invented the subject of classification, it has introduced its own jargon, its own notation, and its own favourite methods. For example, classification is known as pattern recognition, discriminant analysis, decision theory, and assignment analysis'. Furthermore, the word 'classification' has been used with many different meanings, and to refer to a certain stage of the classification process as well as the whole process. For instance, Sneath and Sokal (1973, p.3) state that classification has been used for the end product of the corresponding process. Thus, the result of classification is a classification.

### **2.3 Philosophy of Classification**

In the literature, terms of taxonomy, identification and recognition are frequently used to refer to classification. *The Cambridge International Dictionary of English* defines taxonomy as a system for naming and organising things, especially plants and animals, into groups which share similar qualities. This definition appears to be unsound (inexact), as it fully defines a classification or identification system. Sneath and Sokal (1973, p.3) differentiate classification from taxonomy in that they precisely define taxonomy as the theoretical study of classification, including its bases, principles, procedures and rules.

They also define identification as the allocation or assignment of additional unidentified objects to the correct class once a classification has been established. This means that identification is a secondary step following the classification process. On the other hand, Pankhurst (1991, p.1) states that verbs of 'identify' and 'recognise' carry the same meaning of 'classify' in a general sense, and therefore can be used interchangeably. He, however, also makes a distinction between classification and identification by remarking that 'for a biologist, identification usually means finding the name of a specimen of animal or plant, and the specimen to be identified is usually assigned to a species'. In this remark the assignment of an unknown specimen to a known species is emphasised. It is also assumed that the exact natures of species are known *a priori*. Therefore, for him, identification is again the second step of the classification process. The

statement also suggests that supervised classification is an identification process, which is also pointed out by Hand (1997, p.4). Pankhurst (1991, p.1) also states that 'whatever sort of object is in question, it cannot be identified unless there is already a classification of like objects with which the new object can be compared'. So here, he suggests that the word 'classification' is being used for the first step of the classification process, which at the same time corresponds to the result of an unsupervised classification exercise. However, when the general definition of classification, which is the assignment of unknown elements to known classes, is considered and strictly applied, it can be argued that unsupervised classification is not a proper classification method because there are no known or pre-specified classes involved in the process.

Despite the remarks of the above mentioned authors regarding the stages of the classification process, Cole and King (1968, p.574) state that a preliminary ordering of many data prior to their analysis can be termed 'empirical classification', and representation (or conclusions) of the analysis results can be called 'genetic classification'. In other words, they call the first stage of the classification process 'empirical classification' and the second stage of the process 'genetic classification'.

James (1985, p.3) notes that in classification analysis the existence and the structure of the groups to which the object is to be assigned are of secondary importance. It is the assignment of new cases that concerns us. He also states that classification analysis is sometimes confused with cluster analysis. If one has a mass of currently undifferentiated data and is curious as to whether it has any natural group structure, the method that should be employed is cluster analysis. Basically, cluster analysis attempts to determine any possible groupings in the data. It is not concerned with the problem of classifying new objects into existing groups, as this is the case in classification. It should be highlighted that classification analysis itself is not concerned with identifying any possible (or inherent) groupings that might be contained within a mass of data.

Another problem is pointed out by Harvey (1969, p.326), is that classifications have been produced without its ever being quite clear what purposes they are

designed for. The geographic literature is replete with complex classifications of towns, land uses, climates, regions, and morphometric features, which appear to have been devised with no particular purpose in mind. It is scarcely surprising that many of these classifications have never been used for anything. Geographers have not been alone in their misconduct, and indeed their misuses appear minor compared with those of sociologists and political scientists.

One of the problems of classification in geography is concerned with objects that are unique. All objects are unique in some respect and cannot be classified on this basis, for by definition, each unique object would require a separate class. The Earth might be considered to be a unique object and in many respects this is indeed true, as it lies at a unique distance from the Sun and is the home of the human race. On the other hand, the Earth is one of a series of planets that move round the Sun and thus it can be classified as a planet. Any unique object is rarely incapable of being subdivided. It is the unique combination of its several elements, which give it its unique character (Cole and King, 1968, p.576).

The generality of the classes used in a classification differs depending upon the geographical scale and the purpose for which the classification is intended. For example, if a classification is performed to identify a forested region, the classification could be based on discriminating several forest types including deciduous and conifer forests; if a classification is carried out to discriminate general land cover features, one might use the general class of forest with other land cover features, such as sugar beet, grass and peas; and if a classification is performed on a global scale, then more general class types, including vegetation, soil, and water, are needed. It can be noted that class subdivisions are totally related to the nature of the data, the classification method and the purpose of the study.

Conceptually, scale represents the window of perception, the filter, or the measuring tool through which a landscape may be viewed or perceived (Levin, 1992). The scale issue is of considerable importance in remote sensing studies in that it refers to spatial resolution of the imaging instrument. As each scene contains different sized objects, the use of a single scale may not be relevant to

identify the objects. The problem of determining the most appropriate scale for a particular study is one of the concerns discussed by a number of scientists in different fields. It should be noted that conclusions drawn at a specific scale are valid only for that scale and should not be used to reach conclusions at other scales. Several articles, including Woodcock and Strahler (1987), Foody and Curran (1994), Marceau (1999) and Marceau and Hay (1999), have discussed the issue of scale in geographical studies.

Another issue pointed out by Harvey (1969) is that in geographic literature the difference between classification and ordination is not usually recognised. However, it can be said that classification can include ordination when used in a broad sense. Ordination techniques, also known as dimensionality reduction techniques, are used to reduce the dimensionality of hyperdimensional data for visualisation purposes. Specifically, they search for a configuration in a low-dimensional Euclidean space in such a way that inter-point relationships, usually distances, are preserved with the minimum error. Dimensions are usually reduced to two or three to display the data, typically on a computer screen, and visually evaluate the internal structure of the data. Ordination techniques, details of which can be found in Chapter Four, are particularly useful to determine the outlying (or atypical) elements, clusters of similar elements and other inherent data structures. The primary purpose of such techniques is to project high-dimensional datasets onto two or three-dimensional space, not to concentrate on locating clusters and allocating pixel memberships.

## **2.4 Taxonomy of Classification Techniques**

Classification techniques may be categorised in terms of four criteria. Firstly, they can be classified as supervised and unsupervised depending on the involvement of a training dataset. Supervised classification techniques require training areas to be defined by the analyst in order to determine the characteristics of each category. Each pixel in the image is, thus, assigned to one of the categories using the extracted discriminating information. Problems of diagnosis, pattern recognition, identification, assignment and allocation are essentially supervised classification

problems since in each case the aim is to classify an object into one of a prespecified set of classes (Hand, 1997). Unsupervised classification, on the other hand, searches for natural groups, called clusters, of pixels present within the data by means of assessing the positions of the pixels in the feature space. They are automated procedures and therefore require minimal user interaction. Details of such procedures are given in following sections.

Another distinction among classification methods can be made by considering the underlying philosophy and assumptions of the techniques. By this, they can be classified into two groups: statistical classification and non-statistical classification. Statistical classification procedures employ purely statistical estimations to derive some rules from the data, which leads to some assumptions. The most common assumption of this kind is that the frequency distribution of the data is in Gaussian (or normal) form. However, non-statistical methods do not make any assumptions about the frequency distribution of the data used, and do not use the statistical estimates. The minimum distance and maximum likelihood classifiers can be given as examples of statistical classification methods, whilst the artificial neural network approach and knowledge-based methods can be given as examples to non-statistical classification methods. Detailed information about major statistical and non-statistical classification methods is also given in following sections.

Researchers also categorise classification techniques as being either 'hard' (or crisp) or 'soft' (or fuzzy). In a 'hard' classification (also called one-pixel-one-class classification) each individual pixel is given a single, unambiguous label. This methodology is ideal for cases such as crop classification where agricultural fields are homogenous in terms of the land cover feature they contain. For this, the fields should be large relative to the instantaneous field of view (IFOV) of the sensor, otherwise, it cannot be assumed that a single pixel comprises only a single land cover type. For example, in the case of AVHRR images that have a spatial resolution of 1 km, pixels are likely to be mixed (including more than a single land cover type). The effect of spatial resolution is all dependent upon the nature and scale of variation of the land cover features within the scene. For example, for areas covered with semi-natural vegetation, a spatial resolution of 20 metres

would not guarantee pure pixels and it is most likely that a pixel could contain several land cover types, including herbs, bare soil, bushes and trees. It should be noted that if there are large number of mixed pixels in the image under analysis, then the scale or the resolution of the image selected is not relevant for the purpose of the study. As stated by Mather (1999a, p.189), the question of scale is one that bedevils all spatial analysis; and also fuzziness and hardness, heterogeneity and homogeneity are properties of the landscape at a particular geographical scale of observation that is related to the aims of the investigator. Major drawbacks of 'hard' classification are low classification accuracy levels and poor extraction of information. In addition, this classification methodology can be an oversimplification of the structure of the dataset; in particular, it may be that there are some objects, which definitely do belong to certain groups, but other objects whose group membership is much less evident (Gordon, 1981, p.58).

If one is interested in determining the memberships of pixels unambiguously, that is, the relative level of presence of different classes is in question, 'hard' classification methods are clearly irrelevant. For this purpose, the idea of 'soft' or 'fuzzy' classification was introduced. In 'soft' classification, instead of assigning a pixel to a certain class, the probability of membership of the pixel (called its membership grade) for each class is estimated. The decision relating to the labelling the pixel is left to the investigator. In addition to the use of fuzzy classifiers, it is possible to soften the output of conventional 'hard' classifiers to derive a fuzzy land cover representation (Foody, 1996). This methodology provides more detailed membership information, which allows greater understanding of the nature of each individual pixel. Although 'soft' techniques are relatively new, they have been favoured and used by many researchers. In fact, compared to 'hard' classification, 'soft' classification methodology is more suitable to the classification problem of land cover features that are generally in a continuous form in nature. The main problem with their use is that it is difficult to perform accuracy assessment on the output. The most commonly used 'fuzzy' classification method in remote sensing is the fuzzy *c*-means algorithm, which can be used either in unsupervised or supervised classification fashion. More information on 'fuzzy' classification can be found in Wang (1990a, 1990b), Foody (1996), Mather (1999a, 1999b).

Another categorisation is made by considering the fundamental unit to be considered as the basic element in the classification. This brings out two classification models: per-pixel and per-field classifications. Per-pixel classification is applied as the general method of classification in which each pixel is considered and classified individually. The result of such a classification can be noisy, with a “salt-and-pepper” appearance, resulting from the fact that some pixels can be atypical, mixed or unknown features, that were not included in the training dataset. Moreover, an increase in spatial resolution increases the internal variation within land parcels. Although it is possible to get detailed classification results using such techniques, the noisiness in the output could be unacceptable in some cases. In per-field classification, each individual field, as opposed to a single pixel, is the basic spatial unit, equivalent to the operational taxonomic unit (or OTU) in taxonomy (Sneath and Sokal, 1973). The term field, in this context, refers to a parcel of land, such as different types of agricultural fields and urban areas. It is expected that, using this approach, the noise in the image can be averaged out, which should lead to improvement in the classification accuracy. It should be noted that in this approach it is assumed that every pixel in a field belongs to the same class, which is sometimes not the case. Also, the problematic pixels within the fields are ignored. Although employing such an approach could be useful to depict the field boundaries and to better analyse the results visually, an accurate interpretation cannot be made to determine the total behaviour of the fields. For instance, per-field classification may give misleading results for crop yield estimation.

It should be always borne in mind that the most suitable classification method for a particular classification problem is totally dependent upon the nature of the data available and the purpose of the classification to be performed.

#### **2.4.1 Unsupervised Classification**

In some cases, ground information concerning the characteristics of individual classes is not available. In such circumstances, an unsupervised classification

technique is used to identify a number of distinct or separable categories. In other words, an unsupervised method is used to determine the number of spectrally-separable groups or clusters in an image for which there is no *a priori* or insufficient ground truth information available. Such unsupervised methods can be viewed as techniques of identifying natural groups, or structures, within multispectral image data. While applying an unsupervised method, the analyst generally specifies only the number of spectral groups to be discriminated, and the method generates the specified number of clusters, in feature space, that corresponding to spectrally-separable land cover features. Determination of the clusters is performed by estimating the distances between the pixels in feature space. These automated classification methods are expected to delineate (or extract) the land cover features that are desired by the analyst. As stated by Mather (1999a), this philosophy is like fishing in the pond of data and hoping to come up with a suitable catch.

After the specified number of groups is determined, they are labelled by allocating pixels to land cover features present in the scene. However, some groups may be inappropriate since they represent either irrelevant features for the purpose of the study or mixed classes. Therefore, the spectral characteristics of the area of interest should be sufficiently well known by the analyst to allow him/her to correctly label the clusters representing actual land cover features. Unsupervised classification techniques generally require user interaction in specifying the number of groups to be recognised and in labelling the correctly identified areas with the individual feature (or class) label. Owing to the minimal amount of user involvement, they are usually considered as automated procedures. Two of the most popular unsupervised classification methods are the Chain and ISODATA methods, which are not discussed here as they are beyond the scope of the study, but can be found in numerous text books including Jensen (1996) and Mather (1999a).

Although the description of these methods as automated procedures sounds complicated and powerful, the results of such methods are generally inferior to those achieved by supervised methods. This is partly because most real-world features exhibit complexity in their nature, and therefore they are not easily

separable in terms of their spectral characteristics. In addition, the assumption, forming the basis of the unsupervised approach, that the pixels belonging to a particular class will have similar spectral values in feature space, and all classes are relatively distinct from each other in feature space is difficult to satisfy in practice. Consequently, the accuracy of the results obtained by unsupervised classification methods is limited.

## **2.4.2 Supervised Classification**

Supervised classification may be defined as the process of identifying unknown objects by using the spectral information derived from the training data provided by the analyst. The result of the identification is the assignment of unknown pixels to pre-defined categories. The main difference between unsupervised and supervised classification approaches is that supervised classification requires training data as input. The training data is used to extract the properties of each individual class within the training data. In remote sensing, the ground reference data for training, is generally derived from fieldwork, aerial photography, or from the study of appropriate maps.

Supervised classification methods may be grouped into two general categories: statistical and neural algorithms. In the statistical supervised approach, the information required from the training data varies from one algorithm to another. For example, the parallelepiped classifier requires only the minimum and maximum spectral values for each class in each band, whereas the maximum likelihood classifier requires the mean vector and variance-covariance matrix for each class. In contrast, supervised neural network models do not use any statistical information to identify unknown pixels present in an image. Instead, they use all the training data available. This is the principal feature that makes supervised neural network models more powerful than their statistical counterparts. As a result, no assumption is made about the frequency distribution of the data in supervised neural network models. However, the effect of any incorrect definition of training pixels is more considerable in the neural network models than in the statistical models. This is due to the fact that neural network models take every

individual training pixel into consideration, whereas statistical models use only the overall properties of the data. For example, in the estimation of the mean, the effect of misidentified pixels is smoothed by averaging.

As mentioned earlier, supervised classification is performed in two stages; those of training and classification. In the training stage, the analyst defines the regions that will be used to extract training data, from which statistical estimates of the data properties are computed. In the classification stage, every unknown pixel in the test image is labelled in terms of its spectral similarity to specified land cover features. If a pixel is not spectrally similar to any of the classes, then it can be allocated to an “unknown” class. As a result, an output image, or thematic map, showing every pixel with a class label, is produced.

The characteristics of the training data selected by the analyst are of considerable importance for the reliability and the performance of a supervised classification process. The training data must be defined by the analyst in such a way that they accurately represent the characteristics of each individual feature used in the analysis. Two features of the training data are of key importance. These are the representativeness (or objectiveness) and the size of the training data. In order to have a representative set of data, the sample selection must be performed by selecting pixels that correctly represent the spectral diversity of each class, so that variations in planting times, seed properties and soil conditions are considered. Therefore, samples should be taken from each of such fields to include all spectral sub-classes. The best sampling strategy is to select training pixels randomly from the whole test image. Unfortunately, this is generally not possible in practice, as the ground data for the whole area is generally not available. On the other hand, one way of testing the representativeness of the training set is to examine the number of pixels left unclassified. If there are large numbers of such pixels then it is likely that the training data are not representative of the whole study area, and further samples should be added to the dataset.

The size of the training dataset is also very important if statistical estimates are to be reasonable. Sample size is mainly related to the number of features whose statistical properties are to be estimated. According to Mather (1999a, p.175), ‘the

size should be at least  $30p$  pixels per class where  $p$  is the number of spectral bands, and preferably more'. On the other hand, Campbell (1987, p.311) states that the operator should assure that several individual training areas for each category provide a total of at least 100 or so pixels for each category.

Although supervised classification methods require more user interaction, especially in the collection of training data, they generally give more accurate results compared to unsupervised classification techniques. Therefore, they are mostly favoured by researchers. It should be noted that the current trend is to employ supervised artificial neural network models rather than statistical ones in the classification of remotely sensed image data.

#### **2.4.2.1 The Parallelepiped Classifier**

The parallelepiped classifier, also known as the box classifier, is the simplest statistical supervised classification method used in remote sensing studies. The decision rule of the classifier is based on constructing a parallelepiped for each class with its dimension defined by the minimum and maximum values for each feature. These extreme values are provided by the user at the beginning of the classification process. Sometimes the dimensions of the parallelepiped are calculated by adding and subtracting a multiple of the standard deviation (usually 2 or 3) from the mean for each class and for each feature. The extreme values are used to construct the boundaries of the parallelepipeds. Pixels lying inside the region defined by a parallelepiped are assigned to the class associated with that parallelepiped. If a pixel (e.g. pixel 4 in Figure 2.1) is outside all of these regions, it is labelled as unknown.

An example illustrating the parallelepiped classification philosophy for a two-dimensional feature space and four land cover features is given in Figure 2.1. The parallelepipeds (or boxes) are drawn in such a way that they cover the area of feature space occupied by pixels belonging to the same class. It is certain that pixel 1 and 2 can be correctly classified as "forest" and "peas", respectively. Pixel 4 is left unclassified since it does not lie within any of the rectangular boxes. However, the parallelepiped classifier assigns pixel 3 to the class linseed as the

method uses rectangular decision boundaries that are defined without considering the covariance of the classes, resulting in a complete failure for this pixel. If the spectral distribution of sugar beet and linseed is assessed, it is clear that pixel 3 should belong to the class “sugar beet”.

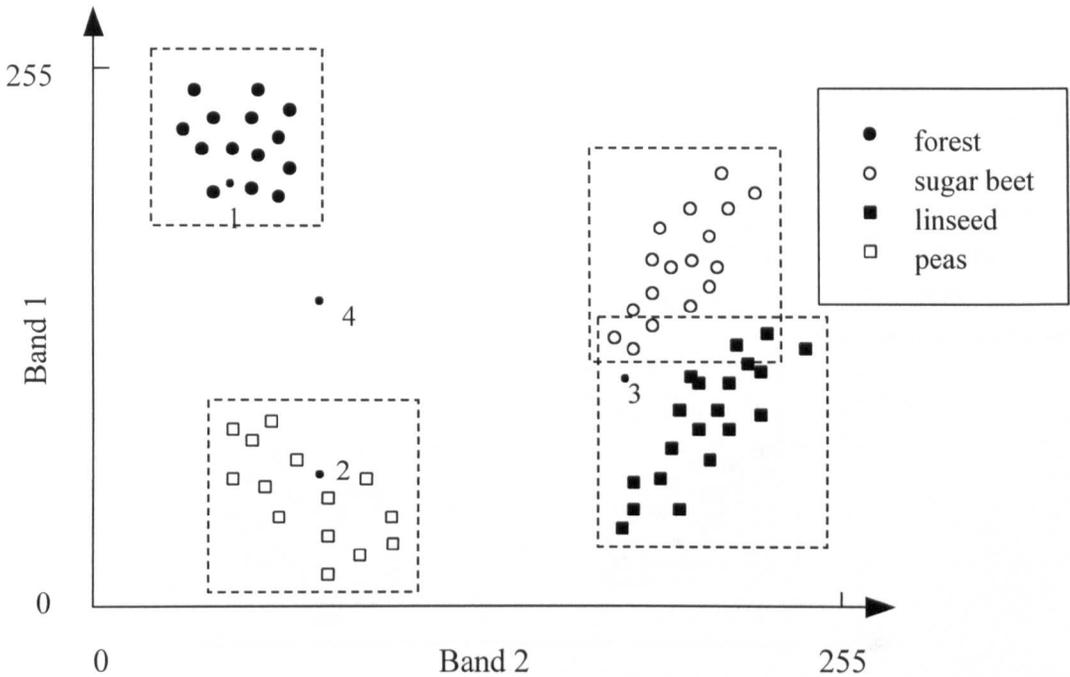


Figure 2.1 Schematic diagram illustrating the principles of the parallelepiped classification method. The boundaries of the parallelepipeds are determined from training datasets.

The main problem with the parallelepiped technique occurs when a pixel lies inside two or more overlapping parallelepipeds, which makes the labelling process difficult. Classifying such pixels correctly is of great importance, as overlapping parallelepipeds are common in situations where remotely sensed image data are used. Several suggestions have been made to overcome this problem. The first and simplest decision rule is to assign the pixel to the first parallelepiped that the pixel was allocated to. The second solution is to employ another, generally more complicated, decision rule to allocate only such multiply-labelled pixels to a specified class. Thus, most of the pixels are classified by the parallelepiped classifier, and only a limited number of pixels are classified by the additional decision rule. The secondary decision rule, used to resolve the problem

of multiple labels, is generally another statistical classifier, such as the maximum likelihood classifier. Another method proposed by Lillesand and Kiefer (1994) is based on modifying the parallelepiped by introducing stepped borders. It is assumed that these borders are more likely to better describe the boundaries of the distribution of pixels in a given class.

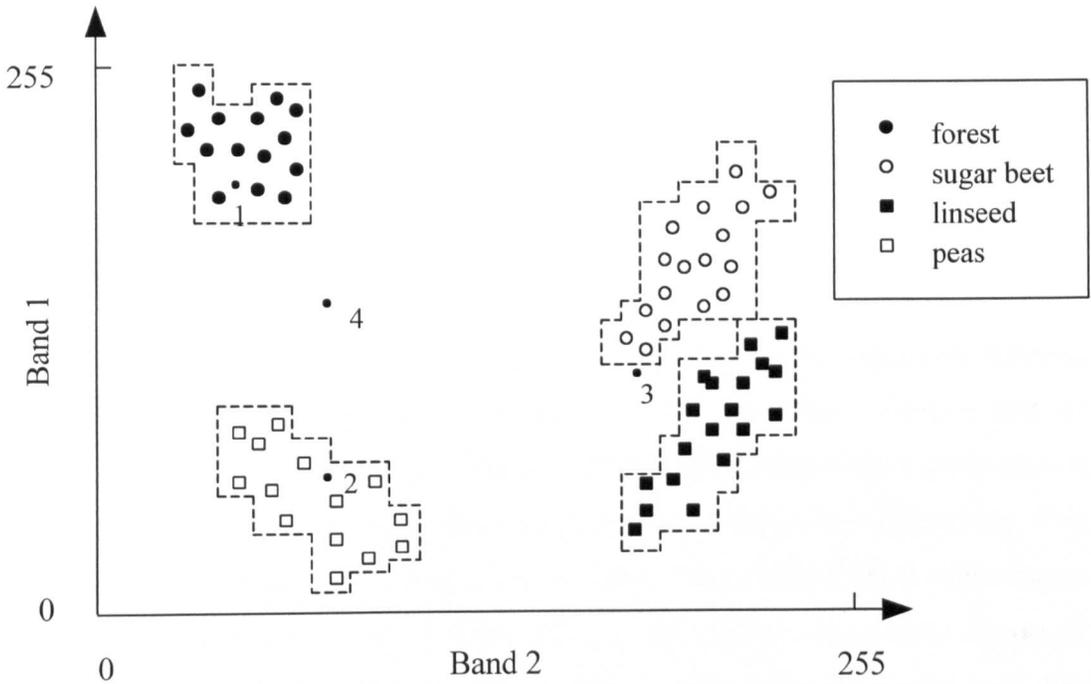


Figure 2.2 More accurate definition of decision regions using stepped borders.

When the idea of stepped boundaries is applied to the problem of Figure 2.1, Figure 2.2 is produced. The stepped boundaries method labels pixel 3 as “unknown” since it is outside all of the boxes, despite the fact that it is very close to the border of the sugar beet class - closer to the boundary of sugar beet compared to that of linseed. This example shows that in the use of the parallelepiped classifier, the representativeness of the training data is of considerable importance. If the training data includes a pixel situated in a similar location to pixel 3, resulting in a larger boundary definition, then the pixel could have been classified as sugar beet.

The parallelepiped classification technique is easy to program and computationally fast because it is based on a very simple concept. However, there are some difficulties relating to the technique. The main problem stems from the

assumption that a particular spectral class can be represented by a rectangular region of feature space. This is not always the case, especially for remotely-sensed data. Moreover, the technique is based on only the minimum and maximum pixel values (or estimates derived from means and standard deviations) computed from the training data to represent the training data. As a result, this simple classification procedure often gives misleading results, that may not be representative of the actual spectral distribution of the land cover classes. It should be also noted that the technique requires a high degree of human interaction.

#### **2.4.2.2 The Minimum Distance Classifier**

A simple and popular classification method is that using the minimum distance classifier (also called minimum-distance-to-means classifier, centroid and k-means classifier). It employs the minimum distance or nearest-centre decision rule to label unknown pixels. As it uses the Euclidean distance in calculations, it is sometimes called the Euclidean distance classifier. This classification method uses the Euclidean distance in multidimensional feature space to measure the degree of dissimilarity between pixels and class centroids computed from training data. The pixel is assigned to the least dissimilar class centroid. Like the parallelepiped classifier, this algorithm does not take all the training data into consideration. It considers the mean (or average) spectral value in each band for each class. The mean centre of each class is estimated from the training dataset, which results in a mean vector. In order to assign a pixel to a specified class, Euclidean distances are calculated for each mean (or centroid) centre, and then the minimum value, i.e. the shortest distance, is determined. As a result, the pixel is allocated to the class that is the closest in terms of the estimated multidimensional Euclidean distance from mean centres.

Figure 2.3 shows the mean centres of four clouds of pixels representing four specific classes. These centres are marked with crosses. In order to illustrate the idea of the minimum distance classification, lines are drawn from pixel 4 to each mean centre to search the closest centre. The shortest distance from pixel 4 is to the mean centre of the forest class, so this pixel can be classified as “forest”.

However, if a threshold distance is employed in the algorithm as described below, then pixel 4 is most likely to be labelled as “unknown” since it lies far away from the boundaries of the forest class. The same philosophy can be applied to determine the membership of other pixels marked on the figure.

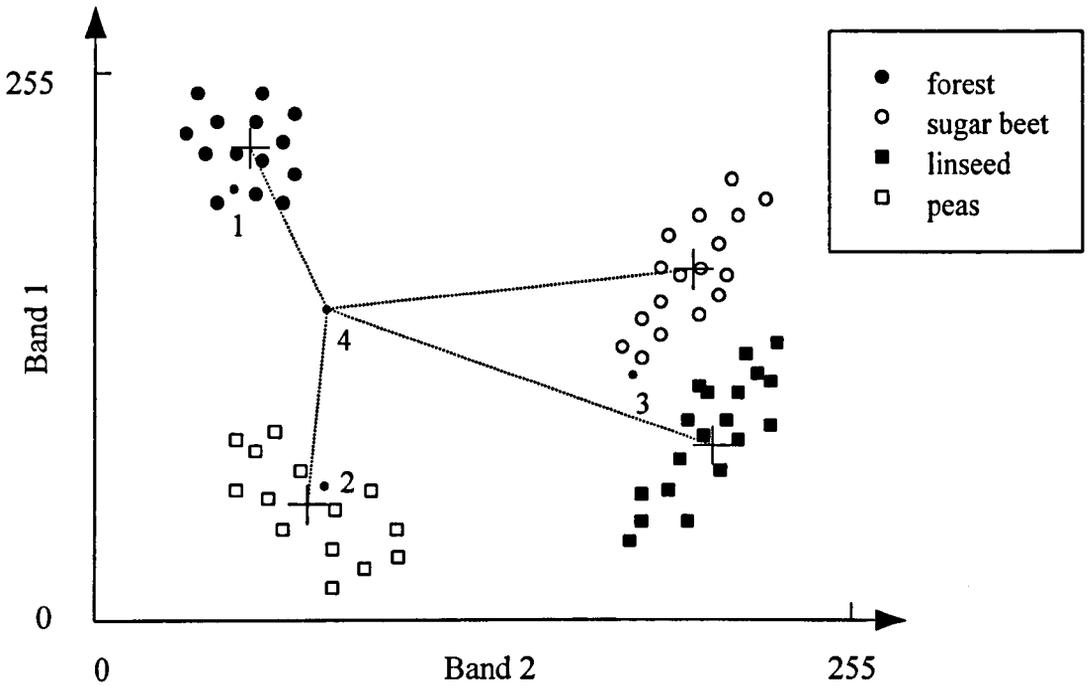


Figure 2.3 Schematic diagram illustrating the classification of pixel 4 using minimum distance classifier. Crosses represent the mean centres of each group.

One problem with the method is that all the pixels are assigned to classes, which may not be realistic or relevant, especially in the presence of outlying and atypical pixels. A modification to the technique uses a threshold distance measure. The threshold distance can either be defined by the analyst at the beginning of the process, or calculated for each class in terms of its standard deviations. Using the standard deviation helps to reflect the nature of the data more effectively. Thus, the condition for a pixel being a member of a class is to have the shortest distance that is at the same time less than the threshold distance estimated for that class. Any pixel lying further than the threshold distance is left unclassified.

The principal limitation of the method stems from the use of a simple distance measure. The Euclidean distance does not take the spectral distribution of the data

in the feature space into consideration. However, an extension of this method that employs the Mahalanobis distance instead of the Euclidean distance, overcomes this limitation by considering the variance-covariance matrices for the classes present in the training data. The Mahalanobis distance classifier is discussed in Chapter Five.

The minimum distance classifier is mathematically simple and easy to program. It can give results that are comparable to more sophisticated methods, such as the Maximum Likelihood Classifier and Artificial Neural Networks in cases where the classes are well-defined in feature space. Efficient use of the technique also requires user involvement.

### **2.4.2.3 The Maximum Likelihood Classifier**

The maximum likelihood classifier, which is the most elaborate and most popular statistical supervised classification method used in remote sensing analysis, is based on the idea that the geometrical shape in feature space of the pattern of pixels belonging to a given class can be represented by an ellipsoid. The locations, shapes and sizes of these ellipsoids are derived from the mean vectors and variance-covariance matrices of the individual classes. While the mean vector is used to determine the position of the centre of an ellipsoid in multidimensional feature space, the variance-covariance matrix, representing the variability of brightness value within a particular class, defines the shape and the size of the ellipsoid. Specifically, the shape of the ellipsoid is defined by the relative dimensions of the axes of the ellipsoid as well as its orientation. The maximum likelihood classification method can be thought of as an extension of the Mahalanobis distance classifier because it is also based on the estimation of Mahalanobis distances between the positions of pixels and mean centres.

A series of concentric ellipses centred on the mean vector of a given class is used to evaluate pixels to be classified in terms of likelihood probabilities. These concentric ellipses represent the probability of membership of a class with contours in such a way that the probability declines away from the mean centre. Basically, the maximum likelihood function describes ellipsoidal 'equi-probability

contours', which can be viewed as probability zones. Unlike the minimum distance classifier, distance from the centre is not the only criterion to judge the membership of a pixel, as the shape and the size of the ellipsoids are important in determining the probabilities of membership. As noted earlier, the size of the axes of the ellipse is related to the variance of the training set, while the orientation of the axes is related to covariance.

As the maximum likelihood classification method represents a cloud of pixels forming a class as a multidimensional ellipsoid, the centre, size and shape of which are derived entirely from the training data, it is expected that the method should yield better results than either the parallelepiped or the minimum distance classifiers. As more information is extracted from the training data and is used to identify new data, one might expect more accurate classification results.

In order to classify an unknown pixel, the membership probabilities for the specified classes are estimated through the probability density functions and the label of most likely class (i.e. having the highest probability value) is assigned to the pixel. If the highest probability value of a pixel is lower than a threshold set by the analyst, then the pixel is left unclassified.

The mathematical theory underlying the maximum likelihood classification technique is outside the scope of this study, but can be found in Thomas *et al.* (1987a), Jensen (1996) and Mather (1999a).

The classification problem presented in Figure 2.2 and 2.3 is also used to illustrate the maximum likelihood classifier. Figure 2.4 shows equi-probability contours drawn around the mean centre of each class. In other words, the locations and shapes of the ellipses are determined entirely by the training data. Pixel 1 and 2 can be again easily classified as forest and peas, respectively, and pixel 4 is left unclassified. On the other hand, pixel 3 seems to be a member of sugar beet class as it is situated very close to the largest ellipse around the sugar beet class showing the lowest probability. Surely, this pixel is assigned to sugar beet class if equi-probability contours are slightly enlarged. This should be, in fact,

implemented in practice since the training data cannot always depict the exact boundaries of the classes, especially when only pure pixels are involved.

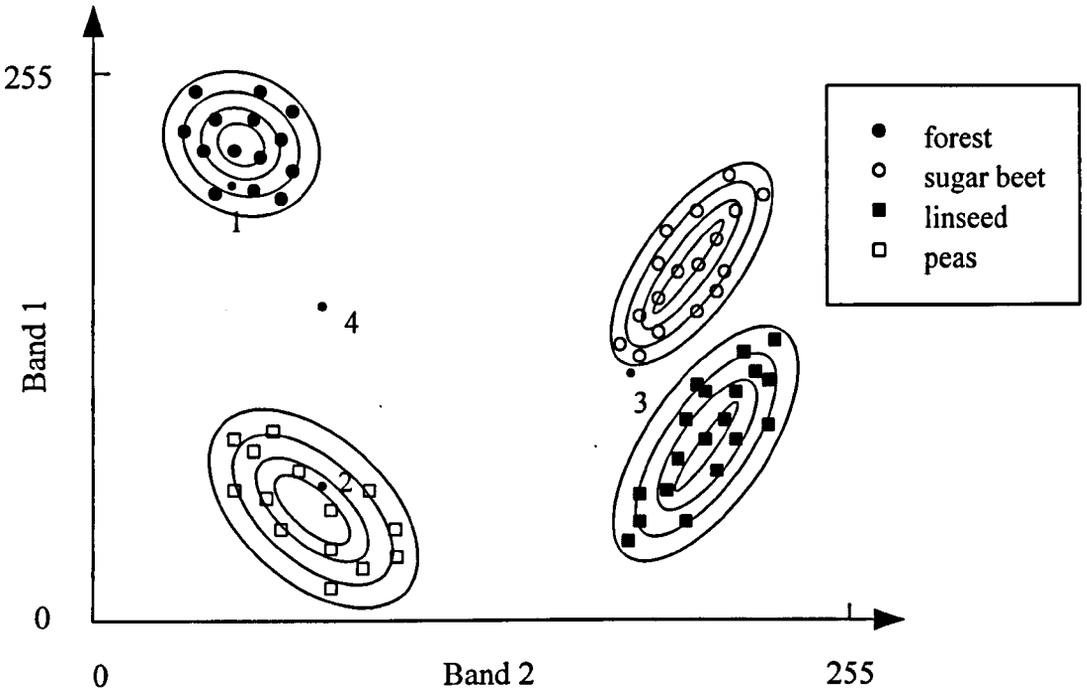


Figure 2.4 Schematic diagram illustrating the idea of maximum likelihood classifier. Concentric ellipses represent equi-probability contours.

In the maximum likelihood method, it is assumed that frequency distribution of a cloud of pixels representing a class is Gaussian, that is, multivariate normally distributed. This assumption of normality is generally reasonable for common spectral response distributions (Lillesand and Kiefer, 1994) and the method can give good results even in situations where slight departures from the Gaussian assumption are encountered. The maximum likelihood classifier gives good results if the assumption of multivariate normal distribution for the training data is fulfilled. The reliability of the results declines when the distribution of the data departs from the normality, especially when the distribution is bimodal. For extreme cases, the multivariate normal assumption does not properly describe the data distribution in feature space and results are misleading. Prior to the use of the maximum likelihood classifier, the histograms of the training data can be analysed to verify that they follow a normal (or Gaussian) shape, a bell-shaped curve.

The maximum likelihood classifier normally assumes equal weights for each class. An extension of the maximum likelihood classifier is the Bayesian classifier that applies two weighting factors to the probability estimated. The first weighting factor is the '*a priori* probability' specified by the analyst for each class, and the second weight relates to the 'cost of misclassification', for each class. The Bayesian classifier is preferred when the necessary information for the two weights is available.

The main drawback of the method is the computational cost required to classify each pixel. This issue is particularly important in circumstances where data to be classified are in a large number of spectral bands, or include many spectral classes to be discriminated. It should be also noted that the maximum likelihood classification method is much slower than the previous techniques described above. The use of categorical data is not feasible as the procedure assumes that the data forming each class are normally distributed.

The maximum likelihood classification method is available in almost all remote sensing and image processing software packages, and it is generally used as a standard supervised classification method. Therefore, there exist, in the literature, many papers comparing the results of maximum likelihood method to those of others. Some of these papers are Belward and De Hoyos (1987), Benediktsson *et al.* (1990), Wilson (1992), Paola and Schowengerdt (1994), and Alpaydin and Gurgen (1998).

## **2.5 Artificial Neural Networks**

One of the most significant recent developments in the theory of classification of remotely sensed images has been the introduction of artificial neural networks (ANNs). The principal theory of ANNs is originated from the desire to create processing systems that behave like the human brain. The brain is extremely complex, consisting of billions of neurons and inter-neuronal connections. It is not exactly known how the brain works, but it is thought that information is processed using a complex network of neurons that work in a parallel manner. ANNs are

designed to mimic such a structure and processing philosophy in a computer environment. In the implementation of ANNs, parallel computation and high computing power are required to perform a particular learning task. Perhaps one of the reasons for the recent popularity of ANNs is the development of new generation computers with increased computing power. Such development is particularly important to reduce the time required by an ANN to learn the characteristics of sample data, which is one of the biggest difficulties in the use of artificial neural networks.

ANNs have been found to be effective in identifying patterns and other underlying data structures in multidimensional data, such as the remotely sensed data. They have some unique advantages, such as their non-parametric nature, arbitrary decision boundary capabilities, and ability to generalise from training data. In addition, unlike traditional statistical methods, such as the maximum likelihood classifier, ANNs permit the use of a range of data types, including categorical data. It has also been reported that artificial neural networks can classify small training datasets better than conventional statistical classifiers. Although many studies have been carried out for a number of years using artificial neural networks in several fields including speech and image recognition, the application of such techniques to remotely sensed image data is quite recent.

Numbers of neural network models have been introduced, along with their learning strategies, which define the methodology of updating the inter-neuron weights associated with interconnections between the neurons so as to improve the performance of the network. In the field of remote sensing, the most popular ANN model has been the multilayer perceptron (MLP), also known as feed-forward neural network (Figure 2.5). It should be noted that the self-organising map (SOM) and learning vector quantization (LVQ) are also widely used in research investigations.

The robustness of artificial neural networks rests on their unique structural representation in that processing units are connected in such a way that every input is locally processed among neighbouring units. Therefore, if any “damage” is experienced by a few of these elements, the effect is compensated by changes in

the neighbouring units. The effect of “damage” is thus minimised, and does not affect the overall performance of the network. ANNs are therefore thought to be tolerant to noise present in the data. On the other hand, this unique structure provides a degree of robustness by taking advantage of slight variations in the data to establish better boundaries between the features.

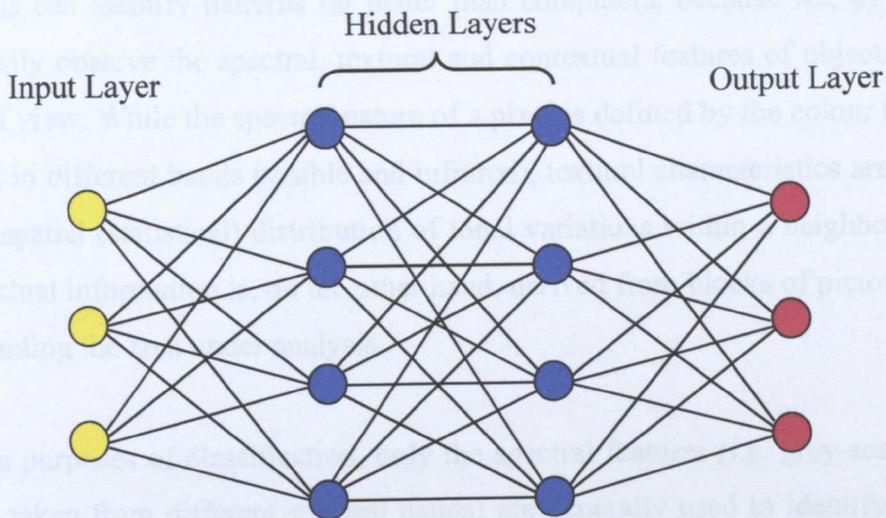


Figure 2.5 A four-layer feed-forward neural network containing three input nodes, four hidden nodes in each hidden layer and three output nodes.

The characteristics of artificial neural networks are discussed in detail by a number of authors, including Pao (1989), Paola (1994), Bishop (1995), Paola and Schowengerdt (1995b), Ripley (1996) and Atkinson and Tatnall (1997). Chapter Three of this thesis provides a detailed review of artificial neural networks and their uses in remote sensing image classification.

## 2.6 Incorporation of Spatial Information

Although spectral information provides useful information to determine the characteristics of land cover features, the addition of a different kind of information may assist in the discrimination of classes that are not easily distinguished using spectral data alone. Spatial information, including texture and context, can be used to provide such additional information, which can be

extracted from image data, digital elevation models (DEMs) and also from thematic maps, such as soil and geology maps.

### **2.6.1 Texture and Context**

Humans can identify patterns far better than computers, because we, by nature, can easily observe the spectral, textural and contextual features of objects in our field of view. While the spectral nature of a pixel is defined by the colour (hue) of a pixel in different bands (visible and infrared), textural characteristics are related to the spatial (statistical) distribution of tonal variations within a neighbourhood. Contextual information is, on the other hand, derived from blocks of pictorial data surrounding the area under analysis.

For the purposes of classification, only the spectral features (i.e. grey-scale pixel values taken from different spectral bands) are generally used to identify ground cover features. A ground cover feature generally covers a region of spectrally similar pixels, with a range of variation in grey levels. Considering the entire region, as opposed to a single pixel, could certainly provide better definition of features, which may result in an improved classification. Thus, the spectral variation within a specific region could be taken into consideration along with spectral values in order to classify each pixel in the image. Since textural properties of images appear to carry useful information for discrimination purposes, it is important to incorporate such information in the classification processes. In remote sensing applications, 3 by 3 windows have mainly been used to characterise texture, by considering the spectral (grey level) values of the eight neighbouring pixels. Taking larger neighbourhoods (as large as 64 by 64) into account may give a better description of the texture.

Despite its importance and ubiquity in image data, a formal approach or precise definition of texture does not exist. Texture discrimination techniques are, for the most part, ad-hoc (Haralick, 1982). However, texture can be described as the variation in grey-level tone within a neighbourhood, representing the pattern of spatial relationships among adjacent pixels. Texture features are usually described

as being fine, coarse, or smooth. As noted by Mather (1999a), the observation of texture depends on two factors. One is the scale of the variation that we are willing to call “texture” – it might be local or regional. The second is the scale of observation.

Many textural measures have been proposed, including the grey-level co-occurrence matrix, autocorrelation functions, optical transforms, digital transforms, textural edgeness, grey-tone run lengths and auto-regressive models. More recently, more sophisticated texture models have been developed with the benefit of recent improvements in computer technology. The characteristics of these texture measures and their theoretical comparisons can be found in several review articles, including Haralick *et al.* (1973), Haralick (1982) and Augusteijn *et al.* (1995).

A considerable amount of research has been carried out to investigate the effectiveness of texture features for the classification of remotely sensed images. For example, Weszka *et al.* (1976) perform a comparative study of texture measures including the Fourier power spectrum, second-order grey-level statistics and the first-order statistics of grey-level differences in a study aimed at identifying three geological terrain types. Augusteijn *et al.* (1995) evaluate the performance of texture measures, including co-occurrence matrices, grey-level differences, texture-tone analysis, features derived from the Fourier spectrum and Gabor filters, using a Landsat TM satellite image for the delineation of a variety of vegetation types. Paola and Schowengerdt (1997) employ texture features in the classification of land-use categories in a neural network based classification using the grey-scale values of the eight neighbours of the pixel to be classified (i.e. using a 3 by 3 window). Mather *et al.* (1998) investigate the effectiveness of spectral and textural information in the identification of surface rock type in an arid region using Landsat TM and SIR-C SAR image data.

Gurney and Townshend (1983) suggest that one disadvantage of textural measures is that there is an effective reduction in the spatial resolution of the final classified image because an area has to be defined within which the measurements of texture are made. According to Mather (1999a), ‘with few exceptions, texture measures

have not been found to be cost-effective in terms of the improvement in classification accuracy resulting from their use. Two reasons could be proposed to account for this: (i) the difficulty of establishing the relationship between land surface texture and scale in terms of the textural feature on the ground relative to pixel size, and (ii) the cost of calculating the texture feature’.

The context of a pixel is derived from the spatial relationships between that pixel and the others in the image. Contextual information can be used either to classify the raw image, or to manipulate the classified image. Contextual information is normally used to modify the classified image. Thus, not only might classification error be reduced by the use of contextual information, but also additional classes could be recognised by separating pixels with the same spectral properties into additional classes according to their context (Gurney and Townshend, 1983). Contextual models can be grouped into four categories in terms of the type of spatial relationship involved; namely, distance, direction, connectivity and containment. Many procedures have been developed to extract contextual information, but a simple way of incorporating contextual information into the classification is to use a form of majority filter window. One of the popular procedures proposed recently is to use geostatistical methods to determine the contextual characteristics of a pixel. Sharma and Sarkar (1998) group the approaches used to incorporate context in the classification of remotely sensed data as follows:

- Methods based on the classification of homogenous objects,
- Techniques based on probabilistic relaxation,
- Methods derived using compound decision theory and sequential compound decision theory, and
- Methods derived based on a stochastic model for the distribution of classes in the scene.

Even though incorporating contextual information has not been as popular as incorporating texture information in the classification, a number of research results have been reported in the literature. For example, Wilson (1992) compares the effect of using pure and mixed pixels in the classification of simulated datasets

employing minimum distance, maximum likelihood and penalised maximum likelihood classifiers. He comes to the conclusion that incorporating contextual information, using a smoothing filter and adding a penalty to the likelihood function, produced much improved solutions with an increase in classification accuracy of as much as 9 percent; Sharma and Sarkar (1998) propose a method to incorporate contextual information using high resolution (e.g. a few metres) or low resolution (e.g. a few hundred metres) data depending on the ratio of region size to pixel size for each class in the classification of a large number of land cover features from three different subscenes. They conclude that the contextual model is superior to other methods in two out of the three examples considered.

Despite the fact that contextual information has been found to be effective in improving the classification accuracy, the selection of the most suitable contextual procedure is the key to produce improved classification results. This selection is totally dependent on the characteristics of the data used. The computational power required for the estimations is also important for the applicability of such techniques.

### **2.6.2 Using Ancillary Data**

As each additional source of information contributes to the characterisation of the objects under analysis, the use of ancillary data is of significant importance. Such data are generally map-based, using themes such as topography, geology, soils and vegetation. Two groups of ancillary information exist: continuous and categorical. Slope and aspect maps extracted from digital elevation models are continuous forms of ancillary information, whereas soil, vegetation and geology maps are in categorical form. One problem in the use of such sources of information is that some of them, such as the vegetation and land use maps, are produced much earlier than the acquisition date of the image. Therefore, they may not represent the reality, causing matching problems between the sources. Another problem is that they are generally in paper form and need to be digitised, which requires considerable amount of time.

The incorporation of such data in the classification can be performed in three main stages: before, during or after the classification. These stages are also known as stratification, classifier operations and postclassification sorting respectively, and are comprehensively described by Hutchinson (1982). He also concludes that using ancillary data can improve the accuracy of the classification when used during any of the stages.

## **2.7 Classification Accuracy Assessment**

Results produced by any classification process applied to remotely sensed data must be quantitatively assessed in order to determine their degree of reliability or accuracy. For this purpose, accuracy assessment is carried out to determine the degree of error in the end-product, which is typically a thematic map or image in remote sensing studies. It is with these accuracy measures that such maps gain meaning. For example, Lillesand and Kiefer (1994) state that a classification is not complete until its accuracy is assessed.

A common way of describing the classification accuracy is by a single percentage value (e.g. 80%) that is calculated by comparing the areas covered by each category in classification map and ground reference data. Such a non-site specific description disregards locational accuracy. In other words, nothing is known about the agreement or disagreement between the ground truth and classification results in any specific location. Therefore, the use of non-site specific accuracy assessment can be misleading.

A comparison between the map generated by a classification process and the ground reference data is necessary. It should be noted that the ground reference data do contain some degree of error, yet it is assumed that the “ground reference” map is correct. This comparison is usually carried out using a confusion matrix, also known as an error matrix or contingency matrix. The term “confusion” comes from the fact that such matrices show the confusion between categories through misclassifications.

Accuracy assessment of classification results using confusion matrices has become a convention, and is therefore used by most researchers. A confusion matrix is a square matrix containing the number of pixels assigned to each class by the classifier being employed. The number of rows and columns in the confusion matrix is equal to the number of categories. Such matrices thus consist of both the ground reference and classification data, with ground reference data represented by the columns of the matrix, and the classification results are represented by the rows. Hence, correctly classified pixels for each class are located along the principal diagonal of the confusion matrix.

A confusion matrix is a very effective way of representing accuracy in that the accuracies of each class are described along with both the errors of inclusion (commission error) and errors of exclusion (omission error) present in the classification. A commission error, represented by off-diagonal row elements of the confusion matrix, occurs when a pixel is included in a category to which it does not belong. On the other hand, an omission error, represented by off-diagonal column elements of the confusion matrix, is the error that a pixel is excluded from the category that the pixel belongs to. Every error is an omission from the correct category and a commission to a wrong category (Congalton and Green, 1999). As well as showing the errors of commission and omission, confusion matrices can be used to compute a number of descriptive and analytical statistics, such as overall accuracy, producer's accuracy and user's accuracy.

### **2.7.1 Overall Accuracy**

The overall accuracy can be computed by dividing the total number of correctly classified pixels (i.e. the sum of the diagonal elements of the confusion matrix) into the total number of pixels in the training dataset (not the total number of pixels classified as there may be some pixels left unclassified). The overall accuracy can be viewed as an average of individual class accuracies. It can only show the overall effectiveness of a classification over the entire scene, not the effect of the classification on individual classes. As emphasised by Story and Congalton, (1986), the categories could, and frequently do, exhibit drastically

differing accuracies, and yet combine to produce equivalent or similar overall accuracies. Therefore, individual class accuracies should be computed and presented together with the overall accuracy. However, the proper way of representing the results of a classification is to present the error matrix so that other accuracy measures can be calculated when needed.

Individual land cover class accuracies can be calculated by dividing total number of correctly classified pixels for each class by the corresponding column or row totals (marginals). As a result, two accuracy measures, producer's accuracy and user's accuracy, can be calculated.

Producer's accuracy is computed by dividing the number of correctly classified pixels in each category by the number of pixels in the training set for that category, which corresponds to a column total. This measure of accuracy shows the classification performance for the pixels of a particular class in the training set. User's accuracy is estimated by dividing the number of correctly classified pixels by the number of pixels that were classified in that class, corresponding to a row total. User's accuracy gives the probability that a pixel allocated to a particular class actually belongs to that class on the ground.

It should be remembered that such procedures only indicate how well the statistics extracted from these areas can be used to categorise the same areas. If the results are good, it means nothing more than that the training areas are homogenous, the classes are spectrally separable, and the classification strategy being employed works well in the training areas. This aids in the training set refinement process, but it indicates little about how the classifier performs elsewhere in a scene. One should expect training area accuracies to be overly optimistic, especially if they are derived from limited datasets (Lillesand and Kiefer, 1994, p. 613).

The accurate interpretation of accuracy measures derived from confusion matrices is of great importance to understand the efficiencies and deficiencies of a classification being carried out.

Table 2.1 Example confusion matrix (from Congalton and Green, 1999).

	Deciduous	Conifer	Agriculture	Shrub	Row Total
Deciduous	65	4	22	24	115
Conifer	6	81	5	8	100
Agriculture	0	11	85	19	115
Shrub	4	7	3	90	104
Column Total	75	103	115	141	434

$$\text{Overall Accuracy} = (65+81+85+90)/434 = 321/434 = 74\%$$

Producer's Accuracy

$$D = 65/75 = 87\%$$

$$C = 81/103 = 79\%$$

$$AG = 85/115 = 74\%$$

$$SB = 90/141 = 64\%$$

User's Accuracy

$$D = 65/115 = 57\%$$

$$C = 81/100 = 81\%$$

$$AG = 85/115 = 74\%$$

$$SB = 90/104 = 87\%$$

Considering the confusion matrix shown in Table 2.1, there exist considerable differences between the user's and producer's accuracies for corresponding classes. These values also show a significant variation from the overall accuracy (74%). If the overall accuracy is solely taken into account, it can be concluded that the classifier has an average accuracy of 74%, without giving the effectiveness of the classification on a particular class, which could be misleading (or inexact). If the overall accuracy and one of the individual class accuracy measures are considered, the analyst could again reach to some misleading conclusions. For example, a producer's accuracy of 87% is achieved for the deciduous class, which is quite high when compared to the overall accuracy. The analyst can conclude at this stage that, although the overall accuracy is average, the deciduous class can be classified with higher accuracy (87%). Drawing such a conclusion could be a serious mistake because the user's accuracy of the deciduous class is only 57%. This means that although 87% of the deciduous areas have been correctly identified as deciduous, only 57% of the areas called deciduous on the classification map are actually deciduous on the ground. A problem can be clearly noticed from the row corresponding to deciduous class in that there is a confusion between the deciduous class and the agriculture and shrub

classes. It can be concluded from the above statements that careful analysis of the confusion matrix is always necessary to present the results and conclusions in a meaningful way.

### 2.7.2 Kappa Coefficient

A statistical measure of accuracy that can be computed from the confusion matrix is the kappa coefficient ( $\kappa$ ). It is a measure of difference between the actual agreement and chance agreement, in that actual agreement is evaluated between ground reference data and classification results, whereas the chance agreement is assessed between the ground reference data and the results of a random classifier. Due to numerous papers using and recommending the kappa coefficient as an accurate measure of accuracy, it has become the conventional way of analysing the confusion matrices. The kappa coefficient is defined by:

$$\kappa = \frac{\text{observed accuracy} - \text{chance agreement}}{1 - \text{chance agreement}} \quad (2.1)$$

and can be computed from the formula:

$$\kappa = \frac{N \sum_{i=1}^r x_{ii} - \sum_{i=1}^r (x_{i+} \cdot x_{+i})}{N^2 - \sum_{i=1}^r (x_{i+} \cdot x_{+i})} \quad (2.2)$$

The  $x_{ii}$  are the diagonal elements of the confusion matrix,  $x_{i+}$  and  $x_{+i}$  are the sums of row  $i$  and column  $i$  respectively.  $N$  is the number of pixels in the confusion matrix, and  $r$  is the number of rows, which is equal to number of columns.

The kappa coefficient assumes that the data are randomly sampled from a multinomial distribution. Confidence intervals around the kappa value can be estimated using the approximate large sample variance, which is computed using

the Delta method, described by Congalton and Green (1999). In the estimation of the variance it is assumed that the kappa statistic is asymptotically normally distributed. The value of  $\kappa$  generally ranges from 0 to 1. However, in some extreme cases it can have a negative value. A kappa value of zero indicates that the classification performed is no better than a random classification of pixels, whilst a kappa value of 1.0 shows perfect agreement between the classification results and the ground reference data. On the other hand, a kappa value of 0.72 can be interpreted as an indication that the classification performed is 72 percent better than one resulting from chance (random assignment of pixels to categories).

Although overall accuracy only incorporates the major diagonal elements of the confusion matrix and excludes omission and commission errors, the kappa coefficient indirectly incorporates the off-diagonal elements using row and column marginals. Therefore, the kappa coefficient can be viewed as an adjusted overall accuracy in that the estimated contribution of chance agreement is subtracted. Kappa is a powerful measure because it can not only be used to assess a single confusion matrix, but can also be used to statistically compare matrices. It can therefore be stated that the kappa coefficient is conceptually a more robust accuracy measure than the overall accuracy. For testing the significance of a single error matrix or two independent error matrices resulting from different dates of images and classification techniques, a Z test statistic is used. Mathematical details of significance tests can be found in Congalton and Green (1999).

Unfortunately, there is no agreement among researchers about which accuracy measure should be preferred in any particular condition. As highlighted by Congalton (1991), each accuracy measure incorporates different information about the error matrix and therefore must be examined as different computations attempting to explain the error.

The estimation of the kappa coefficient estimation can be carried out for the confusion matrix listed in Table 2.1 as follows:

$$\sum_{i=1}^r x_{ii} = 65 + 81 + 85 + 90 = 321$$

$$\sum_{i=1}^r (x_{i+} \cdot x_{+i}) = (115 \cdot 75) + (100 \cdot 103) + (115 \cdot 115) + (104 \cdot 141) = 46,814$$

$$\kappa = \frac{434 \cdot 321 - 46,814}{434^2 - 46,814} = \frac{92,500}{141,542} = 0.65$$

Individual class accuracies can also be computed using the underlying philosophy of the kappa coefficient. Such an accuracy assessment can be performed using the conditional kappa coefficient. For each category (the  $i^{\text{th}}$ ), the conditional kappa can be estimated from:

$$\kappa_i = \frac{N \cdot x_{ii} - x_{i+} x_{+i}}{N \cdot x_{i+} - x_{i+} x_{+i}} \quad (2.3)$$

Similar statistical tests to those described for the kappa coefficient are also available for conditional kappa estimates computed for each class.

Considering the confusion matrix given in Table 2.1, the conditional kappa for the deciduous class, which is the first class, can be computed as:

$$\kappa_1 = \frac{434 \cdot 65 - 115 \cdot 75}{434 \cdot 115 - 115 \cdot 75} = \frac{19,585}{41,285} = 0.47$$

When the formula is applied to other classes (conifer, agriculture, and shrub) in the confusion matrix, conditional kappa values of 0.75, 0.64 and 0.80 are found, respectively.

To standardise reporting procedures for static thematic maps, the confusion matrix must be presented in addition to percent commission error by category, percent omission error by category, the overall accuracy, number of points sampled, map

accuracy (at a specified confidence interval), and the kappa coefficient (Lunetta *et al.*, 1991).

### 2.7.3 Accuracy Maps

The accuracy measures described in section 2.7.2 are derived from confusion matrices and consider the pixels belonging to a certain class as a whole. The error is computed for individual classes and for the overall performance of the classifier. Clearly, they disregard the spatial distribution of classified pixels and the error attached to them. They can be, therefore, viewed as methods that take the generalised error into consideration. These methods cannot give any measure of error to represent the variations in the accuracy of pixels because all the pixels having membership rates over a threshold value defined by the analyst are assigned to discrete classes as a result of a hard classification. However, there is a need to present the class membership levels for each classified pixel to the final user of the remote sensing data products so as to provide an indication of spatial distribution of the accuracy. Such information could be extremely useful, since different regions of thematic maps have varying accuracies, due to the nature and complexity of remotely sensed image data.

Two methodologies presented here are proposed to produce special (unique) output images to display the spatial distribution of the accuracy (reliability of each pixel). Both methodologies portray the spatial pattern of commission and omission errors. The first approach shows the membership probabilities associated with each pixel in terms of using tones of a specific colour attached to a particular class. In other words, the results of a classification are displayed using several colour tones for each class, with tone depending on the membership probabilities, where each class was represented by a distinct colour. Such representation clearly provides a better way of representing the results of a classification in that both class allocations and levels of reliability are given. A sample thematic image from an ANN classification is shown in Figure 2.6. The classification involved seven land cover classes that are presented in four colour tones depending on the output activation value.

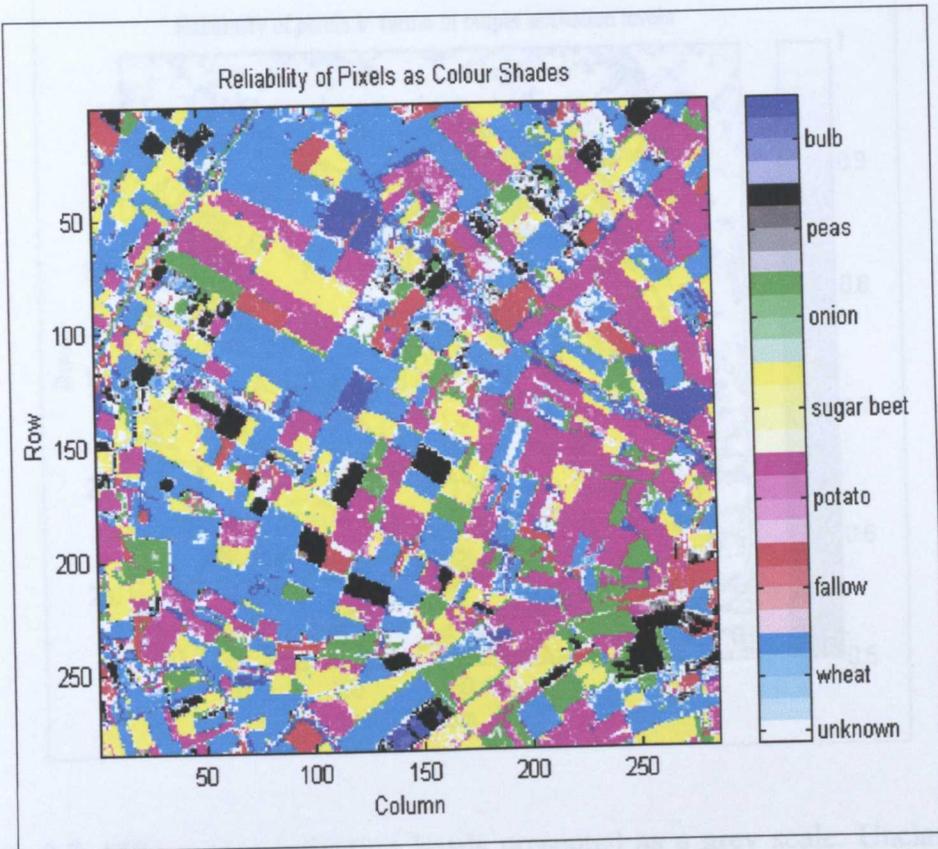


Figure 2.6 ANN output activation levels presented with colour tones for each class. High activation values are represented by darker colour tones, while low activations are shown by lighter colour tones. Unclassified pixels are displayed in white.

The second methodology is based on displaying all the pixels on a grey scale depending upon the membership probabilities. As the artificial neural network classification method is mainly employed in this study, membership probabilities of pixels are the output node activation levels. Output activations lower than 0.5 are set to black, whilst an output activation of 1.0 is set to white. Activation values in between 0.5 and 1.0 are displayed in grey tones. Thus, the areas that were not recognised by the ANN classifier and the effect of spectral variations can be easily recognised. It is also possible to outline the boundaries of the fields, because mixed pixels (mostly appearing in the borders) are classified with low probability of membership; they are, therefore, darker than those pixels within the field. The result of this process is shown in Figure 2.7.

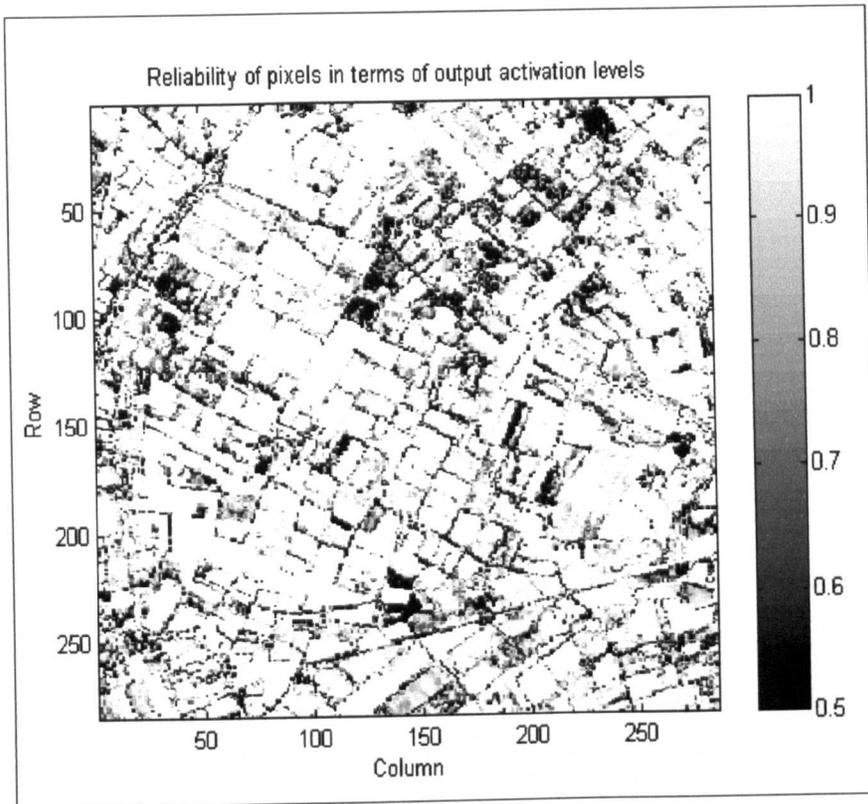


Figure 2.7 ANN output activation levels presented as a grey scale. Unclassified pixels are shown in black. Boundary pixels are clearly evident.

It is possible to display the results given in Figure 2.7 in a colour spectrum since there are a variety of colourmaps available in MATLAB. In addition to the grey-scale colourmap, a colourmap described as 'hot' can be used to make the figure clearer with hues of red and yellow. The result of this process is shown in Figure 2.8. It is also possible to present solely problematic areas, which are the areas left unclassified, in the output image. This process can be also performed using the visualisation toolkit written for this study.

Having the two types of thematic images described above, one can evaluate each pixel and observe the effect of spectral variation in individual fields. It should be noted that spectral variations are basically due to the variations in soil type, the amount of fertilisers used within a field, soil characteristics, different planting dates and different seed properties. Both methodologies are implemented using programs written in MATLAB, and the programs can be run via the visualisation toolkit produced for this study and described in Appendix A.

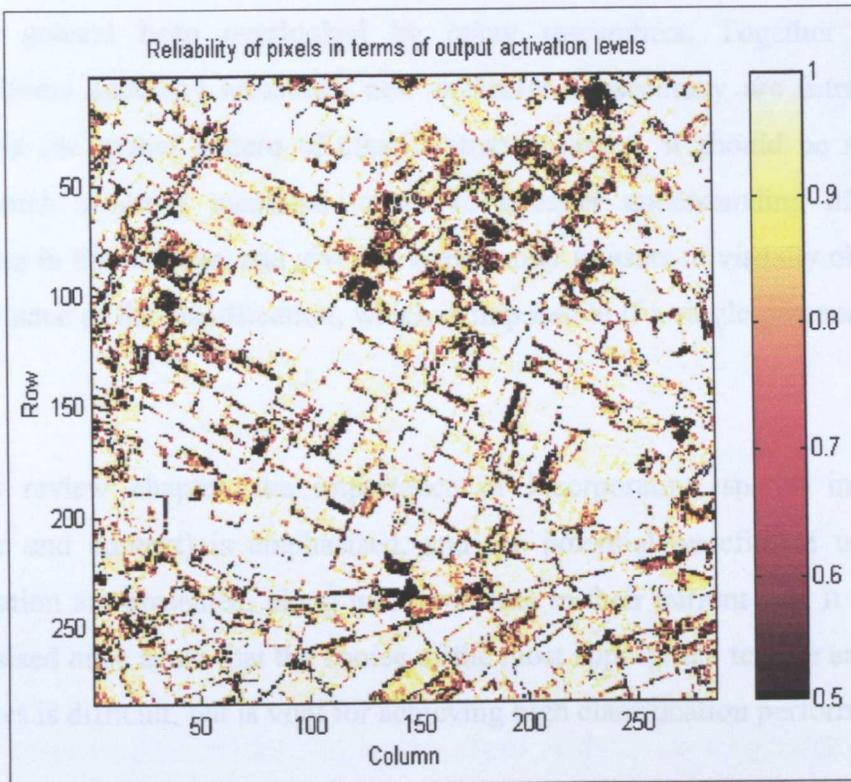


Figure 2.8 Output activation levels presented using another colour spectrum (hot).

As emphasised by Vieira and Mather (1999), the final product of a classification process, typically a thematic map, should be provided together with confusion matrix, statistical error measures and accuracy (reliability) maps, which give the final user a better understanding of the potential error sources associated with remote sensing data products.

## 2.8 Summary

The concept of classification is discussed in this chapter. The philosophy underlying classification is discussed extensively, assessing different views about the issue. Classification techniques, categorised using four criteria, are discussed with an emphasis on supervised classification techniques. The advantages and disadvantages of the techniques are given in detail. The most appropriate classification technique is dependent partly upon the characteristics of the data used and partly upon the nature of the classifier to be employed, in particular on

its underlying assumptions. The issue of accuracy assessment is discussed; this has in general been overlooked by many researchers. Together with the conventional accuracy measures, new measures of accuracy are introduced to represent the spatial pattern of classification accuracy. It should be noted that using such accuracy measures results in a better understanding of possible problems in the datasets, and gives an opportunity to users to visually observe the performance of the classification, which is impossible if a single percentage value is used.

In this review chapter, the importance of incorporating spatial information (texture and context) is emphasised, and the potential benefits of using such information are presented, along with problems in their current use. It should be emphasised once again that the choice of the most appropriate texture and context measures is difficult, but is vital for achieving high classification performance.

# CHAPTER III

## ARTIFICIAL NEURAL NETWORKS

### 3.1 Introduction

A new mathematical model that has emerged recently, and which has made a great impact in the scientific community is the artificial neural networks (ANNs). ANN has attracted increasing attention from researchers in many fields during the last decade, resulting in studies aiming to solve a wide range of problems. ANN has been proved to be more robust compared to conventional statistical classifiers in recognising patterns from noisy and complex data and in estimating their nonlinear relationships. In short, it is known to be good at learning the internal representation of data in any form.

Artificial neural networks are heuristic algorithms, in that they can learn from experience via samples and can subsequently be applied to recognise new data. These systems are intended, in an extremely simple way, to imitate the behaviour of the network of neurons in the human brain. The primary aim of the ANNs is to improve the performance of computer recognition processes by simulating the superior characteristics of the human brain. According to Civco and Waug (1994), 'the powerful capabilities for knowledge acquisition, recall, synthesis, and problem solving of the human brain have inspired scientists from different disciplines to attempt to model its operations. Based on the biological theory of the human brain, artificial neural networks are models that attempt to parallel and simulate the functionality and decision making processes of the human brain'. ANN methods

have been developed and used in a wide range of fields including geography (Hewitson and Crane, 1994), medicine (Dickson *et al.*, 1997), finance (Swingler, 1996b; Giles *et al.*, 1997), manufacturing (Sutton, 1992; Monostori and Barschdorff, 1992), and speech recognition (Hennebert *et al.*, 1994; Altun, 1998).

The power of artificial neural network techniques rests in their unique advantages that may be listed as follows:

- ♦ they are non-parametric,
- ♦ they have arbitrary decision boundary capabilities,
- ♦ it is easy to incorporate different types of data and input structures,
- ♦ they yield fuzzy output values that can enhance classification,
- ♦ they can generalise better, especially in the use of multiple images,
- ♦ they are tolerant to noise.

Of the advantages of ANN techniques, the most important one may be their non-parametric nature. In other words, there is no underlying assumption about the frequency distribution of the data. They learn the characteristics of the training data (or the internal structure of these data), typically in an iterative way, so they may be called data-dependent techniques. It is also worth noting that artificial neural networks can give considerably better results for small training datasets compared to conventional statistical classifiers (Hepner *et al.*, 1990; Blamire, 1994; Paola, 1994 and Foody, 1995). A survey of neural network research and applications can be found in Kemsley *et al.* (1992).

Although artificial neural network classification methods are more robust than conventional statistical approaches, they have a number of drawbacks, related in particular to the long training time requirement, determining the most efficient network structure for a particular problem, and inconsistent results due to the use of random initial inter-node weights. Most importantly, the structure of the network has a direct effect on training time and classification accuracy. There are also problems stemming from the nature of steepest-descent based learning algorithms. All of these problems, which can be encountered in the use of artificial neural networks, are discussed in detail in section 3.9.

Many kinds of neural network model and learning algorithms have been developed as a result of different interconnection strategies. The exact number is uncertain. It is possible to categorise neural network models in terms of two criteria. The first one is based on whether the model employs a supervised or an unsupervised learning strategy. While in supervised models input and output information is provided to adjust the weights in such a way that the network can produce the given outputs from the inputs, only input information is provided in unsupervised models to find out possible classes in the dataset. Major unsupervised neural network models are Kohonen Self-Organising Map (SOM), Adaptive Resonance Theory Networks (ART), Hopfield networks, and Grossberg networks, whilst most common supervised models are the Perceptron, Multilayer Perceptron (MLP), Radial Basis Function Network (RBF), Recurrent Networks, and Learning Vector Quantization (LVQ). The second criterion relates to the directionality of the learning method associated with the network model. If the information advances from input layer to output layer, the learning method is called “feed-forward”. Conversely, if the information proceeds from output layer to input layer, the network is termed “feed-back”.

The most common neural network model is the multilayer perceptron (MLP), an extension of the original Perceptron model that included only an input and output layer. MLP-type networks work in a feed-forward direction where information progresses from an input layer to an output layer in the learning phase. Such networks contain an extra layer or layers termed the hidden layer(s) to overcome the problems of the Perceptron. Due to the involvement of one or more extra layers and the use of nonlinear rather than linear transfer functions, the MLP can approximate and map any kind of problem. Bostock (1994) emphasises that the major reason for the popularity of MLP models is that whilst some problems are more efficiently modelled by other more specialised networks, such as radial basis function networks or binary tree structures, the multilayer perceptron is a good general learning tool for a wide range of applications.

In this study, the MLP that has been the most popular network model for remote sensing studies is employed to accomplish the research objectives. In the remote sensing field, ANNs have been used for various classification problems including

land-cover classification, rainfall estimation, sea ice classification, geological mapping, multisource data classification and cloud-cover classification.

### 3.2 History of Artificial Neural Networks

Warren McCulloch and Walter Pitts conducted a pioneering study of simple logic circuits composed of interconnected neuron-like elements. They also made the first formal definition of the ANN approach in 1943. Nonetheless, it was several years before suitable network architectures and learning algorithms were developed and ANNs gained a great deal of attention.

Donald Hebb, a psychologist, correctly postulated that it was the connections between neurones in the human brain that stored memories. He published his pioneering work 'The Organisation of Behaviour' in 1949. Although subsequent research has shown that memory formation as a result of learning is more complex than Hebb had initially postulated, his concept is still the starting point for most modern artificial neural network theories. The statement was very important and had a strong influence on researchers, especially on two of Hebb's high-school classmates: Frank Rosenblatt and Marvin Minsky. Minsky embarked on the problem of building synthetic networks that functioned like the brain, and he managed to build a complex hardware simulator with tubes and mechanical servos in 1951. However, Rosenblatt studied the mathematical side of Hebb's ideas, and produced a technique that he called the Perceptron. He summarised his work in 'Principles of Neurodynamics', published in 1962. Perceptron is a very simple model including only input and output layers and their interconnections, and trained in a supervised manner. A learning algorithm systematically modifies the weights (links or interconnections) between the neurones until the output converges to a minimum. Perceptron also employed a 'threshold' concept, originally suggested in 1943 by Warren McCulloch and Walter Pitts.

In 1969, Marvin Minsky and Seymour Papert published their book 'Perceptrons', charging that simple two layer networks had strict limitations and could not solve even some simple logical problems, specifically the Exclusive-OR problem. As a

result of this criticism, many researchers gave up working on neural networks, and Minsky and Papert have been blamed for the cut-off in neural network research. Despite their criticism, they proposed the solution of adding an extra layer that contains nonlinear functions. However, the problem was that no learning algorithm existed at that time to train such networks.

Paul Werbos, a PhD student at Harvard University, demonstrated the feasibility of the backpropagation of errors technique in his PhD dissertation in 1980, but his findings were unnoticed until independently redeveloped in 1982 by David Parker at Stanford University.

Parker's work came to the attention of David Rumelhart at the University of California and James McClelland at Carnegie-Mellon University. The two have worked together, along with their 'Parallel Distributed Processing Research Group', to improve the technique, and introduced the 'backpropagation learning algorithm', which is currently the most popular learning algorithm for multilayer perceptrons. The back-propagation algorithm uses a learning rule that is mainly derived from the Widrow-Hoff rule, also known as the 'delta rule'. Hence, the backpropagation learning algorithm is also called the 'generalised delta rule' as it is an extension of the delta rule. The backpropagation algorithm estimates the output values from a set of input values associated with input nodes and a set of randomly determined weights associated with interconnections in the network. These output values are compared to the actual outputs and the error is propagated backward from output layer to the internal layer(s), and then to the input layer. The performance of networks trained using the backpropagation learning algorithm has shown that the limitations of the original perceptron introduced by Rosenblatt were exaggerated in the 1960s.

Two researchers, Teuvo Kohonen and Stephen Grossberg, who are among the few researchers to have continued their studies on neural networks after the publication of Minsky and Papert's criticisms, have had great influence in the development and recent popularity of neural network research by developing several neural network structures and learning algorithms. Kohonen proposed the Self-Organising Map (SOM) that became the most popular neural network

architecture using unsupervised learning. Stephen Grossberg together with Gail Carpenter introduced and developed a network structure known as adaptive resonance theory (ART).

More information about the development of artificial neural networks can be found in Pollack (1989), and Eberhart and Dobbins (1990), in which the history of neural network development is divided into four segments: the Age of Camelot, the Dark Age, the Renaissance and the Age of Neoconnectionism.

### 3.3 Network Structure

The basic element of an artificial neural network is the processing node (Figure 3.1) that corresponds conceptually to the neuron of the human brain. Each processing node receives and sums a set of input values, and passes this sum through an activation function providing the output value of the node, which in turn forms one of the inputs to a processing node in the next layer of the ANN.

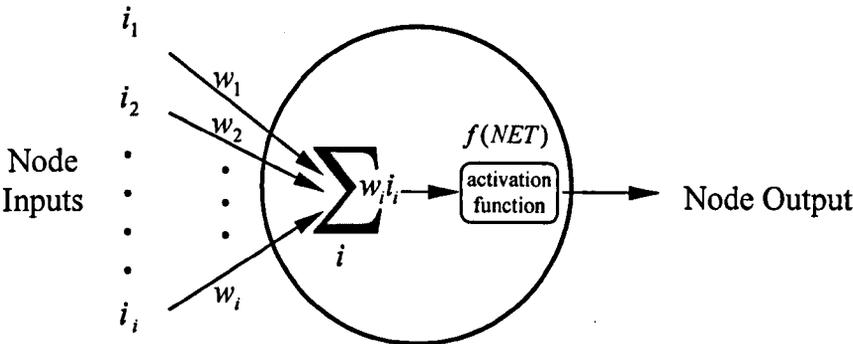


Figure 3.1 A neural network processing node.

Processing nodes make up a set of fully interconnected layers, except that there are no interconnections between nodes within the same layer in the standard feed-forward backpropagation neural networks. The structure of a feed-forward artificial neural network includes three types of layers: input layer, output layer and hidden layer (Figure 3.2). The input layer introduces the distribution of the data for each class to the network. The output layer is the final processing layer that has a set of values (or codes) to represent the classes to be recognised. The

layers between the input and output layer are called hidden layers. These hidden layers, of which there may be only one, perform the basic calculations. It is through these layers that the internal representations of the input patterns can be produced. A typical neural network consists of one input layer, one or two hidden layers and one output layer. The structure of a typical three layer neural network is given in Figure 3.2. Some researchers prefer to refer to three-layered networks, including one input, one hidden and one output node, as two-layered networks excluding the input layer since there is no processing carried out on this layer. Unfortunately, there is no universal agreement on this matter.

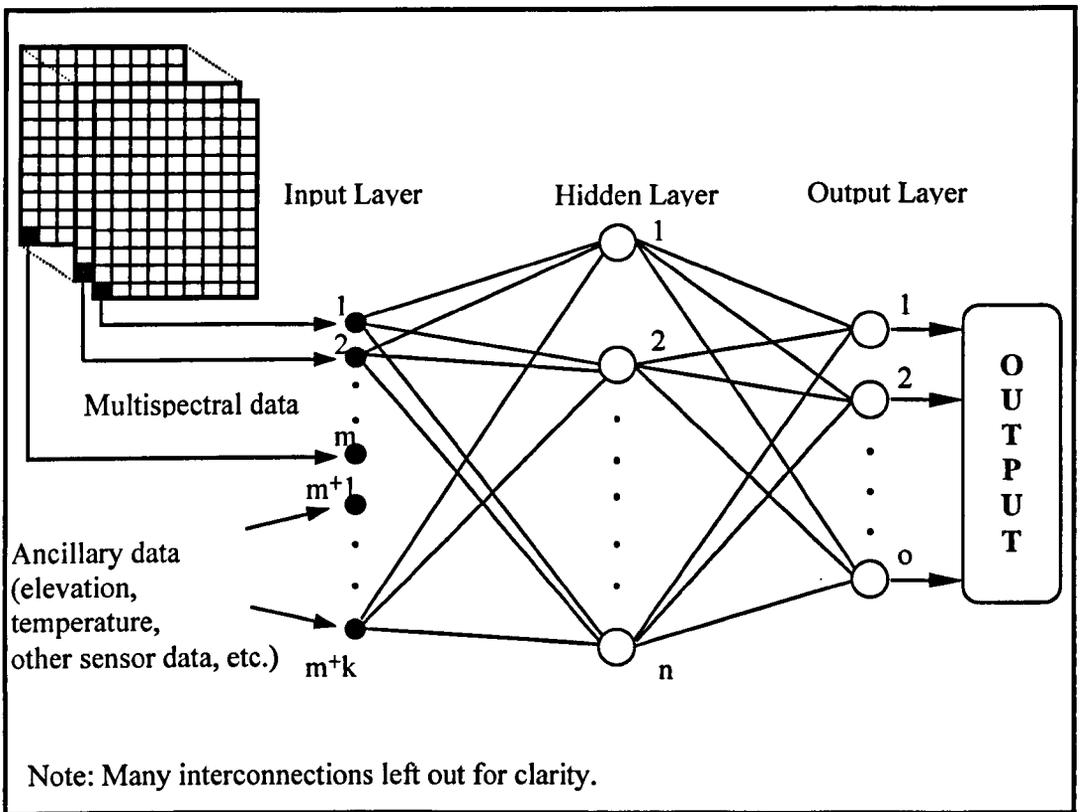


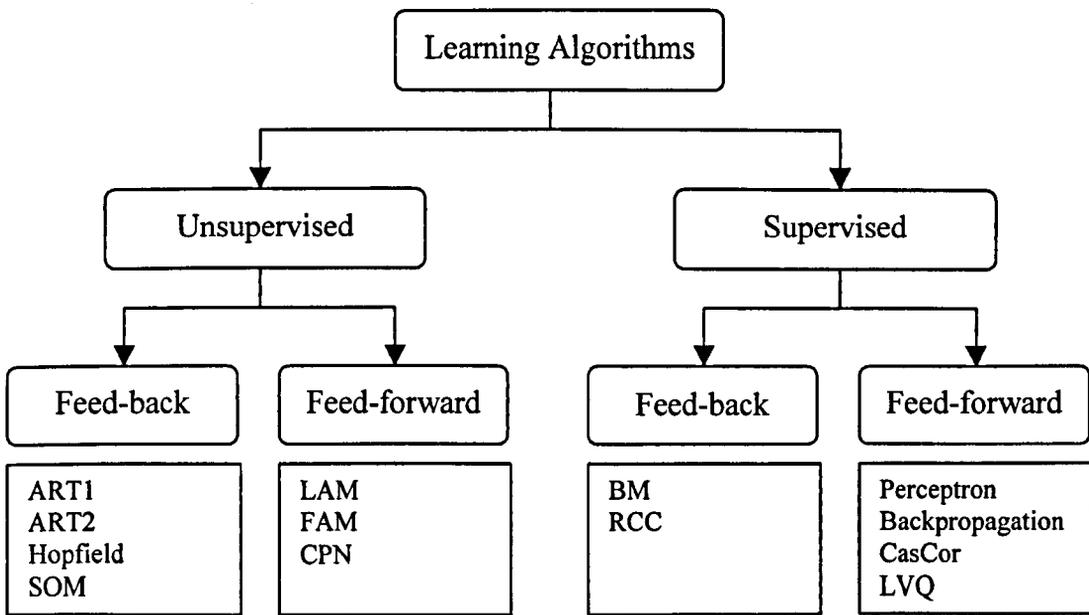
Figure 3.2 A simple three layer feed-forward neural network structure (Paola and Schowengerdt, 1995b).

Each neuron in the input layer represents one of the input features, such as SPOT HRV Band 1, while each neuron in the final layer corresponds to one of the output classes. All inter-node connections have associated weights, which are usually initially randomised. When a value passes through an inter-connection, it is multiplied by the weight associated with that inter-connection. The weights in the network determine class boundaries in the feature space. However, there is

evidence that the initial values of the weights may influence the final classification accuracy significantly (Blamire, 1996; Ardö *et al.*, 1997 and Skidmore *et al.*, 1997).

### 3.4 Learning Algorithms

A learning algorithm is the core of an artificial neural network (ANN) application as it is necessary to make the network neurons and weights capable of performing a useful task by understanding the internal structure of the data. There are many learning strategies developed for different neural network models and the major ones are given in Figure 3.3. However, for training feed-forward neural networks the most popular technique is the backpropagation algorithm introduced by Rumelhart *et al.* (1986). According to Werbos (1995), it has been used in about 70% of ANN applications. He defines backpropagation as a procedure for efficiently calculating the derivatives of some output quantity of a nonlinear system, with respect to all inputs and parameters of that system, through calculations proceeding backwards from outputs to inputs.



ART1-2: Adaptive Resonance Theory 1-2, SOM: Self-organising Maps, LAM: Linear Associative Memory, FAM: Fuzzy Associative Memory, CPN: Counterpropagation, BM: Boltzmann Machine, RCC: Recurrent Cascade Correlation, CasCor: Cascade Correlation, LVQ: Learning Vector Quantisation.

Figure 3.3 Major neural network learning algorithms.

The backpropagation algorithm, also called the generalised delta rule, is an iterative gradient descent training procedure. It is carried out in two stages. In the first stage, after all the network weights have been randomly initialised, the input data are presented to the network and propagated forward to estimate the output value for each pattern set. In the second stage, the difference (error) between known and estimated output is fed backward through the network and the weights are changed in such a way that the difference is minimised. The whole process is repeated with weights being recalculated at every iteration until the error is minimal, or else lower than a given threshold value.

A processing node sums the inputs multiplied by the weights of interconnections and then estimates the output of the node using the activation function:

$$net_{pj} = \sum_i w_{ji} i_{pi} \quad (3.1)$$

$$o_{pj} = f(net_{pj}) \quad (3.2)$$

where  $net_{pj}$  is the sum of the inputs,  $w_{ji}$  is the weight vector,  $i_{pi}$  is the value of the  $i$ th element of the input pattern,  $o_{pj}$  is the output of the node  $j$  for pattern  $p$ , and  $f(\cdot)$  is the activation function, which is usually a nonlinear function. The most common activation function used is the sigmoid function.

The algorithm minimises the error that is the sum of the differences between the actual and calculated output values. The error for pattern  $p$  is estimated from:

$$E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2 \quad (3.3)$$

where  $t_{pj}$  is the target input for  $j$ th component of the output pattern for pattern  $p$ ,  $o_{pj}$  is the  $j$ th element of the actual (calculated) pattern produced by the presentation of input pattern  $p$ . The total error of the network can then be estimated from:

$$E = \sum E_p \quad (3.4)$$

New weights are estimated by updating the weights with  $\Delta w_{ji}$  :

$$w_{ji}^t = w_{ji} + \Delta w_{ji} \quad (3.5)$$

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \quad (3.6)$$

where  $\eta$  is a term called the learning rate that must be initially set by the user. It is used to control the degree of the change in the weights in response to errors in the output during each cycle.

The mathematical theory underlying the backpropagation algorithm is presented only briefly above, as the details are beyond the scope of this study, but can be found in numerous sources, such as Rumelhart *et al.* (1986), Pao (1989), Paola (1994), Bishop (1995) and Ripley (1996).

Training a feed-forward neural network using the backpropagation algorithm involves setting several initial parameters including network structure, learning rate, momentum term and activation function. Of these parameters, two (network structure and activation function) are discussed in later sections. The value of the learning rate has a great impact on the success of ANN applications. If the learning rate is set too high, the learning algorithm may not reach the global minimum, and an increase in error can be observed. If the learning rate is too small, then the process of searching the minimum error will be slow, resulting in long computation times.

The momentum term is added to new (adapted) weights as a fraction of the weight change calculated in the previous iteration. It is used to speed up the process of learning, leading to faster convergence towards the global minimum, and preventing the network from getting stuck into a local minimum by pushing the network away from that point. It also prevents networks oscillating between two

points by forcing the weights to change in the same down-hill direction. Unfortunately, it is difficult to set the optimum values for learning rate and momentum term, as these optimum rates may change during training; therefore, they are sometimes altered during the learning process. Such strategies are called adaptive learning strategies. The effect of learning rate and momentum term is presented in Figure 3.4.

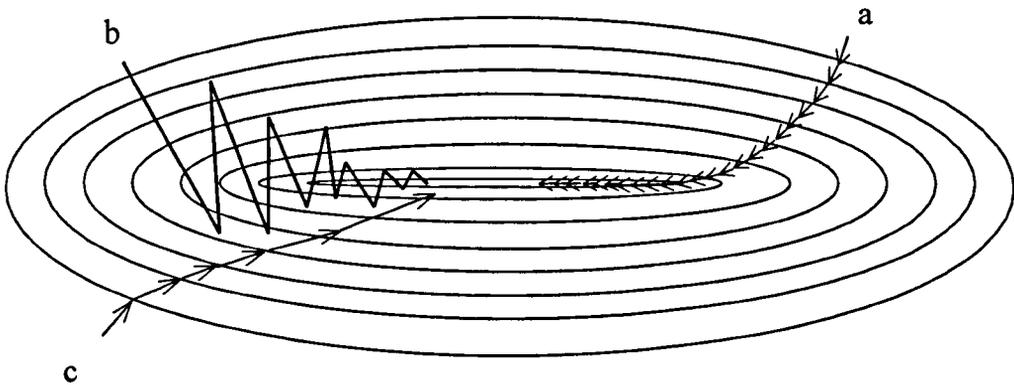


Figure 3.4 The descent in weight space. a) for small learning rate; b) for large learning rate, and c) with large learning rate and momentum term added (Kröse and Van Der Smagt, 1996).

Another important issue is to define a stopping criterion for the learning process, as it is unusual for real-world problems to train a network until the training error is zero. A convergence criterion must be defined to prevent overtraining. This can be considered as a threshold value. When the network reaches this value, training is stopped and the trained network is tested for its performance. There are two methods that have been suggested to find out the best time to terminate the learning process in terms of best generalisation performance. The first method involves employing a validation set for testing the performance of the trained networks during the learning process. Learning is stopped when the error on the validation set starts to rise. According to Ripley (1996, p.154), ‘this is dangerous as it is often encountered examples in which, after an initial drop, the error on the validation set rises slowly for a large number of iterations, then falls dramatically to a small fraction of its previous minimum’. Another problem of using a validation set occurs in cases that there are a limited number of data available that are only enough to form the training and test sets.

The second solution, which has lately been discussed at length in the literature, is early stopping (Wang, 1994a; Wang *et al.*, 1994 and Sarle, 1995). Wang *et al.* (1994) state that a network has better generalisation performance when learning is stopped at a certain time before the global minimum of the empirical error is reached. In addition, for a fixed number of learning examples, the larger the ratio  $d/n$ , where  $d$  is the number of weights (or nodes) and  $n$  is the number of samples, the larger is the improvement in generalisation error if the algorithm is stopped before the global minimum is reached. In this philosophy, it is assumed that when the learning reaches the global minimum, the network loses its generalisation capabilities as it becomes too specific.

Despite its simplicity, it has been reported by researchers that the backpropagation algorithm gives reasonably good results for many problems including those involving complex and noisy data. It is also easy to implement computationally, compared to others. The main drawback of the backpropagation learning algorithm is that there is no guarantee of convergence to minimum error. It is also likely to become trapped into a local minimum, as illustrated in Figure 3.5.

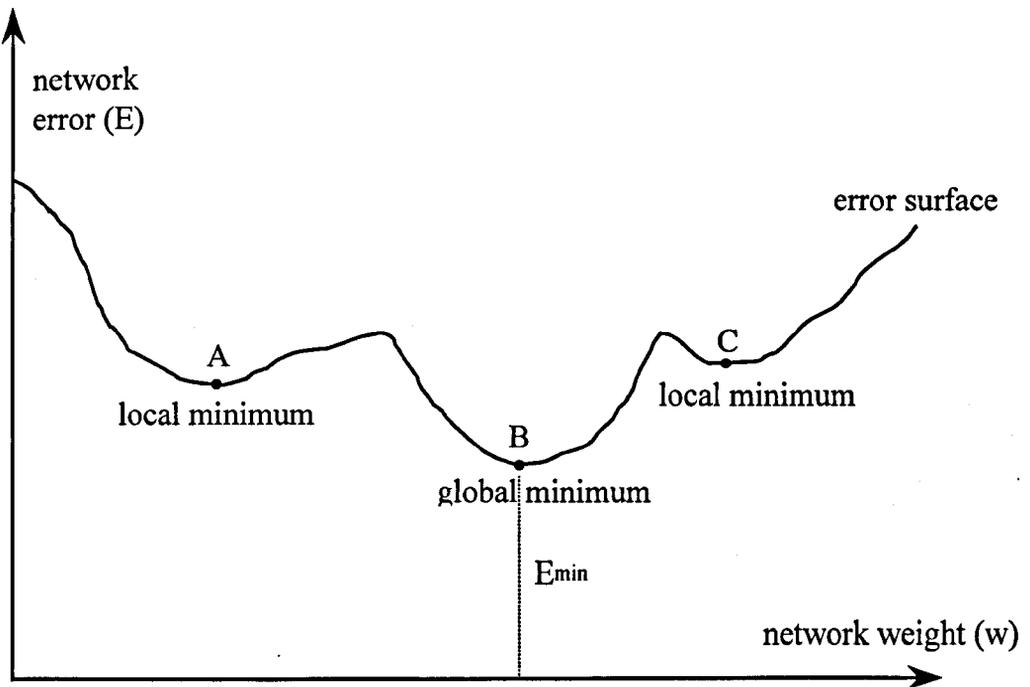


Figure 3.5 Typical error surface with local minima (one dimensional weight space). A and C are the local minima, B is the global minimum.

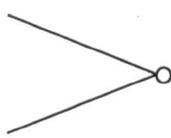
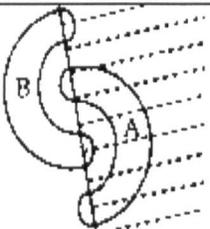
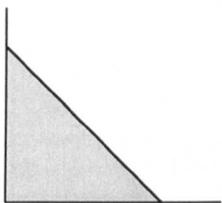
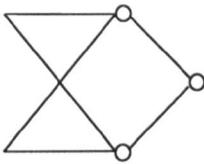
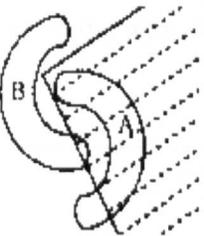
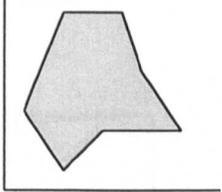
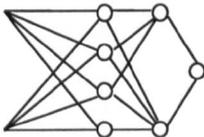
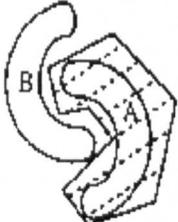
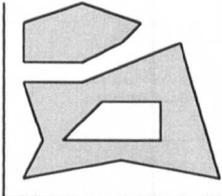
Another problem is that it is possible for the backpropagation process to oscillate between two points. This behaviour is generally observed when the learning algorithm reaches a flat region in the error surface. In addition, because of its gradient descent nature, it is a very slow technique. Details of the inefficiency of backpropagation algorithm and their possible solutions suggested in the literature are discussed in section 3.9.

At the end of the training process the decision boundaries defining the classes are formed in the feature space. Determination of the decision boundaries is dependent on some factors, one of which is the number of hidden layers. An excellent discussion about the decision region capabilities related to the number of hidden layers (Table 3.1) is given by Lippmann (1987). He shows that a Multilayer Perceptron (MLP) with one hidden layer can implement arbitrary convex decision boundaries. Cybenko (1989) has also pointed out that a network with one hidden layer can form an arbitrarily close approximation to any continuous non-linear mapping, assuming only that the transfer function computed by a neurone is nonconstant, bounded, continuous and monotone increasing. However, these conclusions do not suggest that there is no benefit having more than one hidden layer. For some problems a small two hidden layer network can be used where a single hidden layer network would require large number of nodes.

Chester (1990) underlines the fact that the problem with a single hidden layer is that the neurons therein interact with each other globally, making it difficult to improve an approximation at one point without worsening it elsewhere. However, with two hidden layers this problem is overcome. According to Hand (1997), 'a network with two hidden layers allows convex regions to be combined, producing nonconvex, even disconnected regions. Thus, in principle, two hidden layers are sufficient for any problem. However, in practice, it may be advantageous to use more than two layers as increasing the complexity of the nodes can have dramatic advantages'. In a recent study based on land cover classification, Kavzoglu (1999) draws some important conclusions about the effects of the network size on the learning and the performance of the classifier. Some of these conclusions are: large networks learn tasks more quickly but not necessarily better, large networks

do not always improve the accuracy of the classification, and a network that is large enough to learn the characteristics of the data is usually sufficient.

Table 3.1 Types of decision regions that can be formed in the input data space by two, three, and four layer neural networks with hard limiting activation functions and one output node. Regions for networks with sigmoid activation functions and multiple outputs will be more smooth but have similar properties (modified from Paola and Schowengerdt, 1995b).

Network Structure	Type of Decision Regions	Classes with Meshed Regions	Most General Region Shapes
Two layer 	half plane bounded by hyperplane		
Three layer 	convex open or closed regions		
Four layer 	arbitrary (complexity limited by number of nodes)		

### 3.5 Activation (Transfer) Functions

Any differentiable nonlinear function can be used as an activation function, the role of which is to activate the training process. Therefore, there are many possible functions to choose from. For the Multilayer Perceptron (MLP) the activation function must be a nonlinear one; otherwise, it can only discriminate linearly separable objects (classes) like the Perceptron that includes only input and output layers. As they transfer their input values to another value, they are also

called transfer functions. An activation function must be chosen at the beginning of a learning process. The activation function used for hidden nodes may often be different from those used for the output nodes as they have different roles in the learning. The use of an activation function on the output nodes provides such output values that can be used as *a posterior* probabilities. According to Civco and Waug (1994), ‘an activation function is required to avoid saturation of a processing node, caused by extremely large positive or negative internal summations’. Whilst activation functions are employed in order to decrease the number of iterations, they introduce non-linearity into the network, and thus improve the performance. A sigmoid function, also called a logistic function, is generally used for this purpose. The sigmoid function (Figure 3.6) is formulated as:

$$f(NET) = \frac{1}{1 + e^{-NET}} \tag{3.7}$$

where NET is the sum of weighted input values to the processing node.

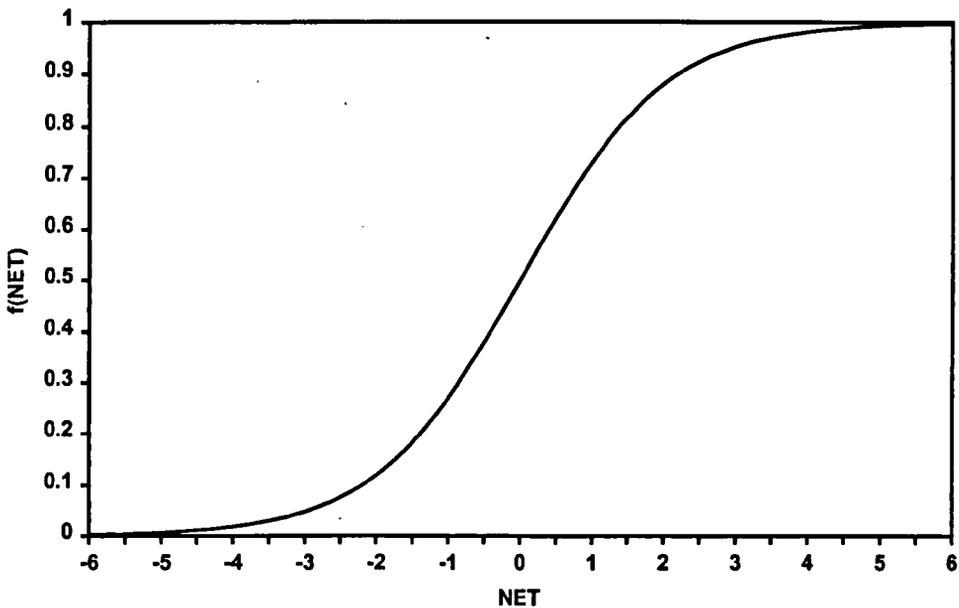


Figure 3.6 The sigmoid activation function.

The sigmoid function has some characteristics that are very important for network performance. As can be seen from the Figure 3.6, output values of zero and one are only possible for input values of  $\pm\infty$ . Instead of the minimum and maximum

values, values 0.1 and 0.9 are generally used. The activation function has a nearly linear input/output relation in between these two extreme values. But as the outputs of a node approach these values, the derivation of the activation function decreases, and since the change in weights is proportional to the derivative value, only very small changes will occur in the weights. The derivative has a maximum value when the output is 0.5. Since the change in weights is proportional to the derivative value, the weights will change rapidly in this case and help influence the node to commit to a high or low value. This feature probably contributes to the stability of the learning stage (Paola and Showengerdt, 1995b).

The second most widely-used activation function is ‘tanh’ function (Figure 3.7). Although the sigmoid and the ‘tanh’ functions are similar, it is often found that using the ‘tanh’ activation function gives rise to faster convergence of the training algorithms than the sigmoid function (Bishop, 1995). The ‘tanh’ function is in the form given below:

$$f(NE\text{T}) = \tanh(NE\text{T}) = \frac{e^{NE\text{T}} - e^{-NE\text{T}}}{e^{NE\text{T}} + e^{-NE\text{T}}} \quad (3.8)$$

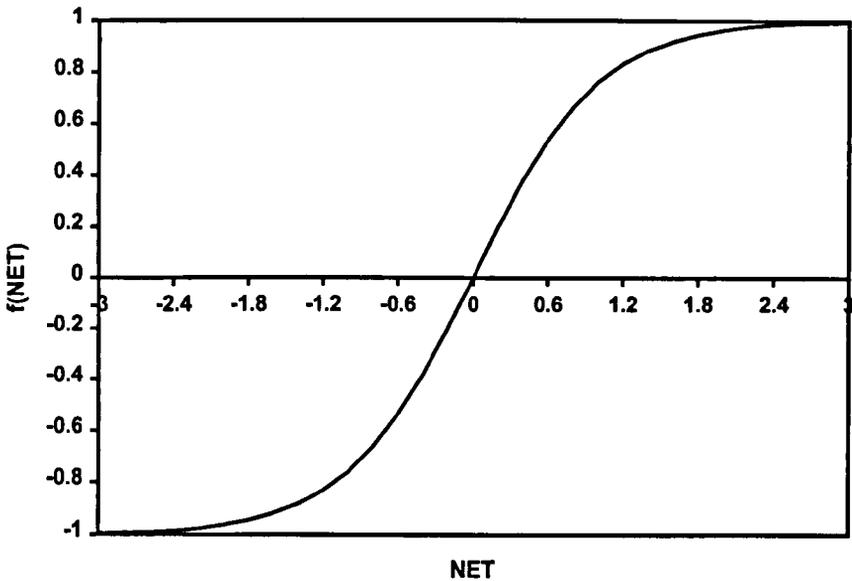


Figure 3.7 The ‘tanh’ activation function. NET is the weighted sum of the inputs to the processing node.

The main difference in the use of these activation functions is that whilst for sigmoid input data and output classes are coded in the [0 1] range, for the ‘tanh’ function they are given in a [-1 1] range. As can be noticed, the ‘tanh’ function represents the data in a broader range, which may have positive effect in the performance of the network.

### 3.6 Encoding

One of the most important issues with neural networks is encoding. This is the representation of real data values in the network as inputs and outputs. Encoding techniques may be divided into two groups; input encoding and output encoding. There are several input data encoding approaches comprising coarse coding (Bischof *et al.*, 1992), Gray coding (Benediktsson *et al.*, 1990) and binary coding (Benediktsson *et al.*, 1990; Heermann and Khazenie, 1992). Coarse coding is a type of interpolation method in which an arbitrary number of input nodes is used to represent the input data. Basically, input data are converted to floating point values using a Gaussian response function. The technique has the advantage of employing continuous-coded values instead of discrete-coded values. On the other hand, whilst in binary coding each output code is converted to the binary codes, such as 0=00, 1=01, 2=11, where for 8-bit band values 8 inputs are required, the Gray code representation, which is a modified version of binary coding, can be derived from the binary code representations as follows:

Assuming that  $b_1, b_2, \dots, b_n$  is an  $n$ -digit binary code, the corresponding Gray codes  $g_1, g_2, \dots, g_n$  can be obtained from:

$$g_1 = b_1$$

$$g_k = b_k \oplus b_{k-1}, \quad k \geq 2 \tag{3.9}$$

where  $\oplus$  is modulo-two addition (Benediktsson *et al.*, 1990).

Although it has been claimed that binary representation of the input data help the network to detect the small differences between the pixels, it requires many inputs; for instance, four band data require 32 input nodes, which is the main disadvantage of the method. However, the most widely used technique used in this study is to scale data to the range from 0 to 1 for the sigmoid and  $-1$  to 1 for the ‘tanh’ activation function. According to Paola and Schowengerdt (1995b), ‘although this representation is not a mathematical requirement, it avoids the use of a scale or shift factor every time the sigmoid activation function is evaluated, thus reducing floating point computations’. To prepare remotely sensed data for the network the values of the pixels in each band are simply scaled to this range by setting the minimum value to 0 (or  $-1$ ) and the maximum value to 1 for each band. Each band is represented by a node in the input layer with the scaled values. The simplest form of data input is to use one pixel to represent each band, which is called per-pixel based classification. This technique has been used in most applications such as Paola and Schowengerdt (1995a), Dreyer (1993), Bischof *et al.* (1992) and Benediktsson *et al.* (1990).

An extension of the per-pixel approach is to use a window (generally 3 by 3) of pixel data from each band of the image as input. This helps the network use textural information to better learn and classify the data by using neighbourhood information. Researchers have attempted to use texture information to improve their ANN classification performances. For example, Hepner *et al.* (1990) reported that using 3 by 3 windows of input pixels allowed the network to assimilate data relating to spatially adjacent pixels in both the training and classification operations. Paola and Schowengerdt (1994) also highlight the fact that using texture in the network significantly reduced the number of iterations to train the network. Unlike conventional statistical techniques, texture information can easily be incorporated into ANN for classification tasks. However, there are two major problems in the use of such information in ANN classifications. Firstly, in order to incorporate texture into feed-forward networks trained with backpropagation algorithm, a large dataset relative to network size is required to estimate the texture accurately. Using larger window sizes would further increase this requirement (Blamire and Mineter, 1995). Secondly, more processing time is

required to train such networks as the size of the network increases (more weights will be adjusted).

One output node is generally used for each ground-cover class to encode the output classes. One of the most widely used approaches is that desired values are assigned to be 1 at only one node and 0 at other output nodes in the output layer to represent output classes. For example, the first output class for a six-node output layer is represented by 1 0 0 0 0 0, as the second one is represented by 0 1 0 0 0 0. One of the advantages of this approach is that although the sum of the output activations is rarely equal to 1, they are interpreted as *a posterior* probabilities of the pixels being a member of each class in practice. The higher the output value, the greater the confidence that a pixel is a member of that class. The opposite also holds true. Binary coding has also been used for output encoding, which results in extremely slow convergence. Another type of output coding suggested by Benediktsson *et al.* (1990) is temperature coding. In this coding scheme the representation for  $n$  has 1 in its first  $n$  digits and  $-1$  in the rest (e.g., 4 = 1 1 1 1 -1 -1).

Benediktsson *et al.* (1990) conclude in their comparative study that using Gray-coded inputs and temperature-coded outputs gave higher accuracies and required fewer learning cycles than using binary-coded inputs and outputs.

The continuous output values resulting from the classification process can be interpreted in different ways, including a measure of classification confidence, class mixing and *a posterior* probabilities. According to Bischof *et al.* (1992) and Paola (1994), results of artificial neural networks are *a posterior* probabilities but the probabilities are different from the ones produced by the maximum likelihood classifier. Foody (1996, 1997, 1999) also states that the activation level of an output unit indicates the strength of membership of a pixel to the class associated with the output unit. This feature of ANNs is very important particularly for fuzzy land cover classifications from remotely sensed data.

The simplest way of assigning a pixel to an output class is to choose the class of the output node with the highest probability of membership. For example, if the

output of the network for a given pixel is 0.25 0.01 0.01 0.70 0.02 for a five output-class problem, then this pixel is assigned to class four as it has the highest output layer activation value in the fourth output node. In some cases, this assignment would be misleading as the probability of being a member of a class may be lower than 0.5, which could be the highest output value. Therefore, a modified version of this scheme including a threshold parameter in decision making is generally applied. In such cases, in order to be assigned to an output class the maximum probability of membership for a pixel must be higher than the user defined threshold value. This scheme is employed in the present study.

### **3.7 Generalisation**

The power of the network depends on how well it describes new data after completion of the training process. This is the main criterion for judging the performance of a network. Generalisation may be defined as the ability of a neural network to interpolate and extrapolate to data that it has not seen before (Atkinson and Tatnall, 1997). There are three factors affecting the generalisation capabilities of a neural network. These factors are the size of the training data, training time, and the architecture (structure) of the network.

Approaches developed to discover the proper size of the neural networks are called dynamic network design strategies, and they can be divided into three main groups. The first group starts with a small network and iteratively increases the number of nodes in the hidden layer(s) until satisfactory learning occurs. This is known as the constructive approach. The most widely used constructive method is the cascade correlation algorithm developed by Fahlman and Lebiere (1990). The cascade correlation algorithm starts the training process with no hidden layer. Only the input and output units are fully connected to each other. The network is trained for a user defined number of times, and then a hidden layer with a single node using a sigmoid activation function is added to the network. The new hidden node has connections to all the input and output nodes. The values of the weights of the hidden units are determined before adding the hidden layer. These weights are calculated so as to maximise the correlation between the output of the unit and

the residual error of the network outputs. The network is then trained with the new hidden layer to reduce the error. Each time a hidden layer with a single node is added and retrained. Training is performed in a way that only the new weights are trained, with the value of all the previous weights left unchanged. It is assumed that each node thus learns the characteristics of a particular feature or class, acting like a specialised feature detector, and training the network only for these new links preserves this special relationship. It has been claimed that the cascade correlation algorithm can provide a small network to solve a variety of problems.

Since such techniques employ a number of small networks that are more sensitive to initial circumstances and learning parameters than larger networks, they are more likely to become trapped in local minima ending in failure of training. Furthermore, a number of networks must be trained to find the optimum network structure, resulting in long processing time.

The second approach is to begin with a larger network and make it smaller by iteratively eliminating nodes in the hidden layer(s) or interconnections between nodes. These types of algorithms are called pruning algorithms and will be discussed in detail in the next section.

There also exist some techniques that employ both constructive and pruning strategies. These techniques couple the pruning and constructive techniques in a way that the size of a small network is increased during training until a reasonable solution is reached and then the size of the network is reduced using pruning methods to make a smaller and faster network that also has higher generalisation capabilities. Such an algorithm is introduced by Hirose *et al.* (1991). In their algorithm, training starts with a network including only a single hidden node, and a new node is added to the hidden layer whenever the network is trapped in a local minimum, which is detected by checking the change in the training error after every hundred iterations. If the change (improvement) in the error is less than one percent of the previous error, and the error is higher than a user defined value, then it is assumed that the network is trapped into a local minimum. Therefore, a new node is added to hidden layer. When the error is below the user defined value, the network is pruned by eliminating the last inserted hidden node. After

each node pruning, the network is retrained to restore the loss of the node. Once the final network solution does not provide enough power to generalise, the previous network structure is adopted, which is thought to be the optimum network structure for the problem under consideration. The main problem in the use of such techniques is the long training time requirement.

An extensive survey study of the techniques used to determine the optimum network structure is carried out by Alpaydin (1991), who also introduced a learning algorithm called GAL (Grow and Learn) that involves a pruning method. GAL learns an association at one-shot due to being incremental and using a local representation. Details of the dynamic network design strategies can also be found in Bostock (1994).

### **3.8 Pruning Algorithms**

Pruning is the name given to the process of examining a network, determining which units are not necessary to the solution and removing those units (Sietsma and Dow, 1988). For example, Figure 3.8(b) shows a link-pruned version and Figure 3.8(c) shows a node-pruned version of Figure 3.8(a), which is a fully connected network structure. Several nodes and a number of connections between the nodes have been removed. The use of pruning algorithms is relatively recent, and experience of their use has not been widely reported in the remote sensing literature (Kavzoglu and Mather, 1999).

After a network is trained to a desired solution with the training data, hidden layer nodes or inter-connections are analysed to determine their level of participation in the solution. There are several ways to determine the identity of non-contributing units. A widely used approach uses a form of sensitivity analysis to locate non-essential neurons, involving the setting of the value of a specific neuron to zero for all training set inputs and noting the effect on the network output. Thus, neurons that have only a minor effect on the performance of the network can be identified and removed (Reed, 1993 and Karnin, 1990). An alternative approach,

introduced by Sietsma and Dow (1991) and known as Noncontributing Units, is an interactive two-stage method in which the analyst examines a trained network

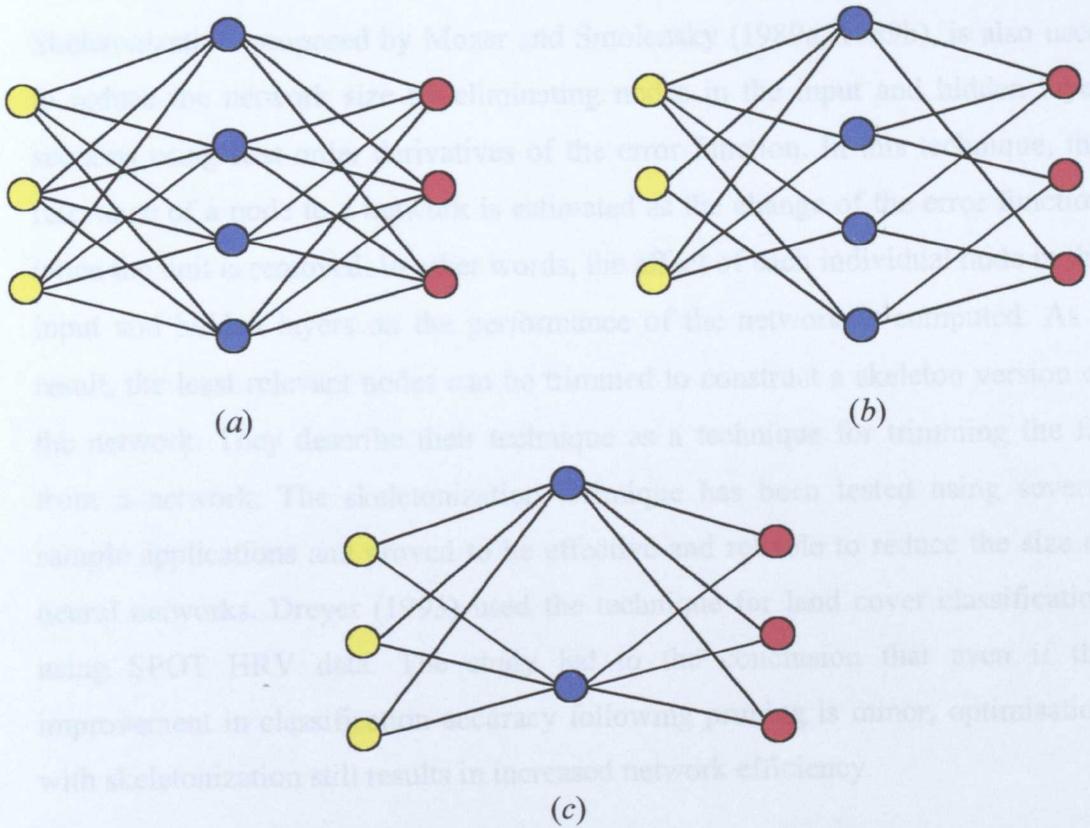


Figure 3.8 (a) A fully connected artificial neural network. (b) Interconnection pruning (seven links removed). (c) Node pruning (two nodes removed).

and decides which neurons are to be removed. Criteria used to identify unnecessary nodes in the first stage are as follows:

- 1) If a neuron has a constant output over all the training patterns then it is not contributing to the solution and can be removed,
- 2) If a number of neurons have highly correlated responses (e.g. identical or opposite) over all patterns then they are redundant and can be combined into a single unit. All their output weights should be added together so the combined unit has the same effect on following units.

In the second stage, nodes that are linearly independent from the other nodes at the same layer, which are not strictly necessary, are removed.

Skeletonization, proposed by Mozer and Smolensky (1989a, 1989b), is also used to reduce the network size by eliminating nodes in the input and hidden layer sections using first-order derivatives of the error function. In this technique, the relevance of a node to a network is estimated as the change of the error function when the unit is removed. In other words, the effect of each individual node in the input and hidden layers on the performance of the network is computed. As a result, the least relevant nodes can be trimmed to construct a skeleton version of the network. They describe their technique as a technique for trimming the fat from a network. The skeletonization technique has been tested using several sample applications and proved to be effective and reliable to reduce the size of neural networks. Dreyer (1993) used the technique for land cover classification using SPOT HRV data. The study led to the conclusion that even if the improvement in classification accuracy following pruning is minor, optimisation with skeletonization still results in increased network efficiency.

Castellano *et al.* (1997) introduced a pruning method to reduce the size of trained feed-forward neural networks by iteratively removing hidden layer neurons and then adjusting the remaining weights in a way that preserves overall network behaviour. This method is formulated in terms of a system of linear equations, and an efficient conjugate gradient algorithm is used to solve the system in the least-squares sense.

There are three major pruning methods used to remove the least effective interconnections (or links) in neural networks, namely, magnitude-based pruning, optimum brain damage, and optimum brain surgeon. Magnitude-based pruning, the simplest pruning algorithm, is based on deleting inter-connections having small magnitudes. It is assumed that the interconnections whose magnitude is small will have a minor effect on the performance of the network. Hassibi and Stork (1993) report that this simple and naively plausible idea may lead to the elimination of the wrong weight, and they point out that some small weights may be necessary for low error.

The optimum brain damage (OBD) pruning algorithm, introduced by Le Cun, Denker and Solla in 1990, is based on the second order derivatives of the error function. According to Le Cun *et al.* (1990), ‘the basic idea of the OBD is that it is possible to take a perfectly reasonable network, delete half (or more) of the weights (interconnections) and wind up with a network that works just as well or better’. The aim is to delete, in an iterative fashion, the weights associated with inter-node connections whose removal will result in the least increase of network error ( $E$ ). The method requires the calculation of the Hessian matrix, which can become very large and thus increase the computational cost of the procedure; consequently, several simplifications have been proposed. The main simplification (or assumption) is that the Hessian matrix is a diagonal matrix, in that there are values only in the diagonal section of the matrix. However, Hassibi and Stork (1993) report that the Hessian matrices for problems that they have considered are strongly non-diagonal, and this may lead the OBD algorithm to eliminate the wrong weights.

The optimum brain surgeon (OBS) pruning algorithm, introduced by Hassibi and Stork (1993), can be thought of as the extension of or a slightly more complex form of optimum brain damage (OBD). Although the OBS and OBD methods are based on the same theoretical approach, the OBS technique does not make any assumption about the form of Hessian matrix. Therefore, the OBS method may be expected to be both more complex and more robust than the OBD. It is claimed by its proponents that the OBS is significantly better than either the magnitude-based and OBD techniques, and that the OBS approach permits the pruning of more weights than other methods (for the same error on the training set), and thus yields better generalisation on test data. The drawback of the method is that the inverse of the Hessian matrix has to be computed to judge saliency and weight change for every link. Therefore, the method is quite slow and takes much more computer memory than the other methods discussed.

As the main objective of pruning techniques is to improve the generalisation capabilities of the network, instead of defining the saliency as training error, Pedersen *et al.* (1995) proposed to use the generalisation error as the weight

saliency. This idea resulted in the extension of two most widely used pruning techniques, OBD and OBS, to  $\gamma$ OBD and  $\gamma$ OBS. The only difference between the counterparts is the use of saliency measure as different error criteria.

### **3.9 Problems in the Use of Artificial Neural Networks**

Although the Multilayer Perceptron (MLP) trained with the standard backpropagation learning algorithm is a good general learning tool, and used intensively in research, it has some inherent limitations that may have a great impact on the performance of the classifier. Perhaps the most important one is that the MLP is not guaranteed to converge to the optimum solution, the global minimum, even when one exists. Furthermore, the MLP is computationally demanding and slow. In general, the use of ANNs requires some critical decisions on the part of the user, specifically a remote sensing researcher, which may affect the accuracy of the resulting classification. In terms of the factors involved, these decisions may be divided into two main groups: external factors and internal factors. External factors include the image resolution (spatial and radiometric resolution) and sample choice. However, internal factors are the choices of an appropriate network size (structure), initial weights, number of iterations, transfer function, and learning rate. While internal factors result from the limitations of the MLP and the backpropagation learning algorithm, external factors given here are specifically caused by the issues associated only with remote sensing related studies. These parameters need to be understood and adequately resolved in order to produce good results using ANNs.

The user of ANNs should, first of all, consider all the external factors which may affect the success of subsequent processes before beginning to prepare training and test data for neural network classification. The choice of the number of classes is related to the resolution (spatial and radiometric) of the data, as well as to the overall aim of the project. Spatial and radiometric resolutions are the two major characteristics describing an imaging remote sensing instrument. In addition, the scale of the study and the accuracy required from the classification process should also be taken into consideration. If the aim is to produce a general classification map of the USA or Europe using four major categories (water,

vegetation, urban area, and ice), then there would be no point in using high resolution images such as 20m spatial resolution SPOT HRV data, but AVHRR data would be relevant for such an application. On the other hand, if the aim of a project is to produce detailed information about a specific agricultural crop, even SPOT HRV data would not be appropriate, as a spatial resolution of a few metres would be necessary. As summarised by Woodcock and Strahler (1987), 'the choice of an appropriate scale, or spatial resolution, for a particular problem depends on several factors. These include the information desired about the ground scene, the analysis methods to be used to extract the information, and the spatial structure of the scene itself.' As emphasised, the factor of scale is very important in remote sensing applications and it must be investigated very carefully due to the cost and efficiency of the application.

Radiometric resolution is also important in the sense that high radiometric resolution sensors give more detailed information than low resolution, which is very important to distinguish features. It has considerable effect on the performance of the classification. A study carried out by Tucker (1979) showed that a few percent overall accuracy improvement occurs in the classification performance when using 256 (8 bit) rather than 64 (6 bit) level imagery.

In order to get reliable and accurate classification results, a representative set of samples is necessary. If the training data are not representative then the network may fail to classify new data that are dissimilar to all of the training data. How best to estimate the volume of training data that is required in order to achieve a required level of generalisation is a crucial question that is discussed by Foody (1995) and Hepner *et al.* (1990).

Sample size for each class and sampling methodology used are the two key issues which have been investigated. A small sample size is not enough for a neural network to recognise all classes and to determine the class boundaries in the feature space precisely, whereas a large number of sample patterns can make the network overspecific and requires more computation time for training. In the learning process, the same number of patterns are usually employed for each class in order to avoid any bias. However, the number of samples for each class should

be appropriately defined to reflect the class complexity. Therefore, in the training process, a larger number of samples should be used for broadly defined classes compared to more tightly defined classes. In such cases, analysing the variation of pixels in the feature space using two and three dimensional scatter plots is of great importance. It has been pointed out by Blamire (1994), Blamire and Mineter (1995) that the relative sample size has a considerable effect on the performance of a neural network solution. The effect of training sample size on error rate of learning (training) and testing sets is given in Figure 3.9.

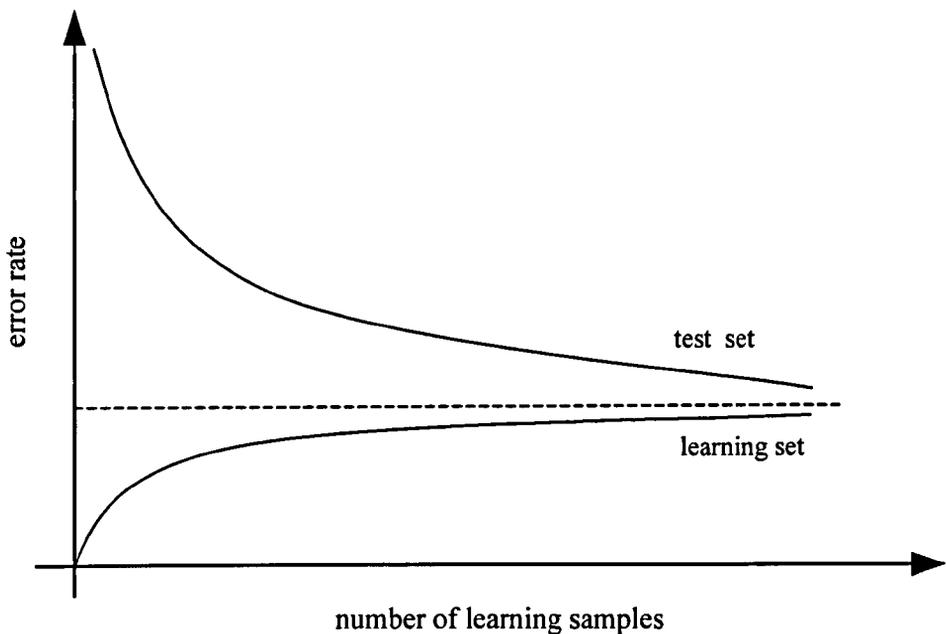


Figure 3.9 Effect of the learning set size on the error rate. The average error rate and the average test error rate as a function of the number of learning samples (Kröse and Van Der Smagt, 1996).

The literature contains a number of discussions of sampling methods and optimum sample size. Different methods have been tested and various conclusions have been drawn, but there is no general agreement on these issues. However, the general tendency is to use random sampling techniques (simple and stratified random sampling methods) and to choose a minimum of 50 or 100 elements per sample depending on the complexity and heterogeneity of each class. It has been reported that the selection of the training data is more important than the size of the training dataset.

Samples are generally chosen via random  $n$  by  $n$  windows rather than by the choice of random pixels. This causes spatial correlation problems due to the correlation between nearby points. It is reported that autocorrelation is related to the interaction between adjacent pixels, the pixel dimensions and the effects of data preprocessing. It should be borne in mind that the validity of a classification is dependent upon the sample size and the representativeness of the sample. As stated by Mather (1999a), 'it is very easy to use an image processing system to pull out "training samples" from an image but it is a lot more difficult to ensure that these training samples are not contaminated either by spatial correlation effects or by the inclusion in the training sample of pixels which are not "pure" but "mixed" and therefore atypical of the class which they are supposed to represent'.

Another issue in preparing sample data is to detect and eliminate atypical pixels to get pure data for classification. There are two methods suggested by Mather (1999a), which are estimation of Mahalanobis distances and using hierarchical cluster analysis method on each training sample. Visualisation of the corresponding band values of each pixel can also help to overcome this problem by projecting the multi-dimensional data to two or three dimensions.

While sampling the input images, one should consider the internal variation of the classes by looking at the spectral characteristics of the area under analysis. The variation in fields results from the effects of factors such as variations in soil moisture, seed characteristics, topographic position, and different planting dates. This variation should be reflected in training samples to get reliable results from artificial neural network classification. If an area of interest contains only one spectrally uniform field for a class, then a subset region would be adequate to train the network. If, however, the same area includes spectrally different regions or the image includes several spectrally distinct regions, then it would not be appropriate to use only one sample area. The most acceptable solution might be choosing samples from spectrally extreme areas.

As stated earlier, there are five main internal factors affecting the accuracy of an ANN classification. These are the network size, choice of initial weights, number

of iterations, type of transfer function and learning rate. Understanding these factors and choosing their appropriate values are key issues for a successful ANN classification. Firstly, in the case of layered neural network architectures like MLP, network size is not only related to the number of layers but is also related to the number of nodes for each layer and the number of connections between these nodes. For a given dataset there may be an infinite number of network structures capable of learning the characteristics of the data. The question is: what size of network is optimum for a specific dataset. Unfortunately, it is not easy to answer this question. In the light of current knowledge, the neural network architecture that gives the best result for a particular problem can only be determined experimentally (Paola and Schowengerdt, 1995b; Kanellopoulos *et al.*, 1997). The quality of the solution found by a neural network is strongly dependent on the network size used. In general, the network size affects network complexity, and learning time, but most importantly, it affects the generalisation capabilities of the network and, as a consequence, the classification accuracy.

As the size of input layer is generally equal to the number of image bands and that of output layer is equated to the number of output classes, the adjustable part in the neural networks is the middle section, the hidden layer(s). The input layer can be expanded by simply adding new data sources as additional neurons, but this increases the computation time by the order of  $N^2$  (Heermann and Khazenie, 1992). In other words, if the size of input data is doubled, the time required to train the network would increase by a factor of four. New datasets should be added only if they contribute to an improved classification.

It is sometimes necessary to use more than one hidden layer to train a network properly, whereas in some cases it is a luxury to use extra hidden layers which can make the network too specific and use more training time. While the use of multiple hidden layers provides some potential benefits, it does not solve the problem of determining the appropriate number of hidden nodes. It simply extends the problem from one to multiple layers. However, it is known that more hidden layer nodes make the neural networks more powerful in determining the location of complex decision boundaries in feature space. The network can thus learn the characteristics of more complex data. However, such networks tend to

'memorise' the patterns in the training set, become overspecific to the data, and hence give poor performance for patterns that are not included in the training data. The effect of number of hidden units on the error rate of learning set and testing set is presented in Figure 3.10.

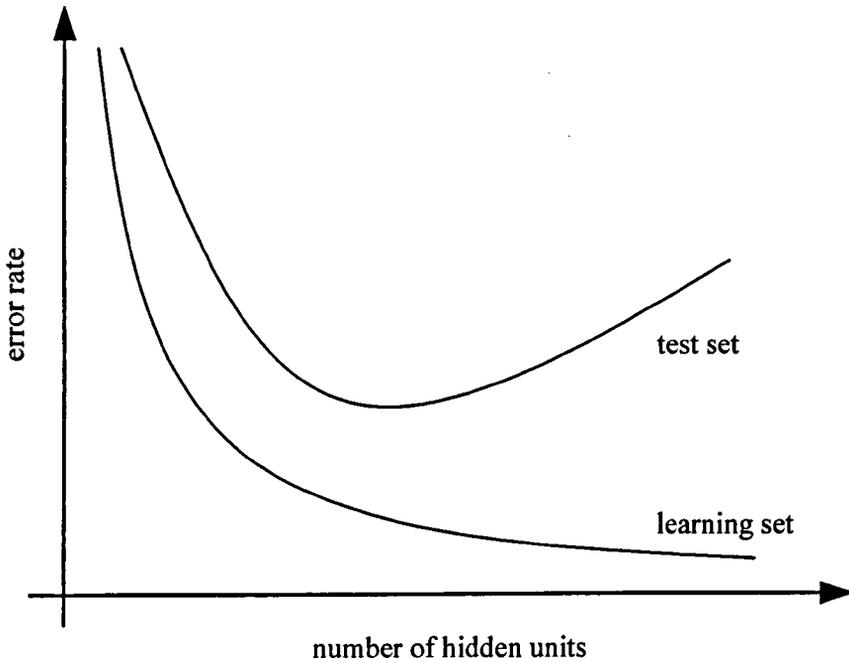


Figure 3.10 The average learning set and the average test set error rate as a function of the number of hidden units (Kröse and Van Der Smagt, 1996).

On the other hand, since there is an almost linear correlation between the number of samples required for the training process and the number of hidden units, large networks generally require more training samples than small networks to achieve good generalisation performance. In many applications, a limited number of samples is available. Consequently, using large networks for such datasets may lead the network to produce unsatisfactory results.

If too few hidden units are used then the network will fail to achieve a satisfactory performance since it cannot learn the underlying data structure. It should be noted that a smaller network is more likely to generalise well, since it extracts the essential and significant characteristics of the training data. The advantages of using small networks are summarised by Karnin (1990) as follows:

- The cost of computation, measured by the number of arithmetic operations, grows (almost) linearly with the number of synaptic connections. Hence, a smaller network is more efficient in both forward computations and learning.
- Neural network learning is usually based on a finite (often small) set of training patterns. A network that is too large will tend to memorise the training patterns and thus have poor generalisation ability. This phenomenon is also known in classification theory as ‘tuning to the noise’, and it occurs whenever the number of free parameters of the classifier is large relative to the training data.
- There is always the hope that a smaller network will exhibit a behaviour that can be described by a simple set of rules.

However, there are problems for small networks being sensitive to initial conditions and learning parameters. These networks are also more likely to become trapped in a local minimum as the error surface of a smaller network is more complicated and includes more local minima compared to the error surface of a larger network (Bebis and Georgiopoulos, 1994). Therefore, the best generalisation performance is obtained by trading training error against network complexity (Le Cun *et al.*, 1990).

Paolo (1994) states that the choice of the number of hidden nodes is not a significant problem as in the experiments hidden layer sizes greater than three produced adequate classification results after a similar number of iterations. Furthermore, a hidden layer equal in size to the output layer was adequate for all the classifications attempted. This was true even for those using a few hundred input nodes. Ardö *et al.* (1997) also state that no significant difference was found between networks with different numbers of hidden nodes, or between networks with different numbers of hidden layers. However, there are some reported studies, such as Kröse and Van Der Smagt (1996), Lawrence *et al.* (1996), and Bebis and Georgiopoulos (1994) that contradict these optimistic statements. The effect of hidden layer and hidden layer nodes therefore needs further investigation. For current studies, a general rule, as underlined by Wang (1994a)

and Kavzoglu (1999), could be that as long as the network size is large enough to learn the characteristics of the data, the size of the network does not have a significant effect on network performance.

The nature of the input data is also related to the size of hidden layers in the sense that the separability of the classes present in the data determines the degree of difficulty of the problem, as the neural network establishes hyperplanes in feature space to distinguish the classes based on the data characteristics. Another factor is the number of output classes. A large output layer makes the problem more complex since the network will be determining more complex class boundaries in the feature space. It is thus important to choose the number of output classes appropriate to the scale and nature of the study region to avoid unnecessary training, as noted above.

Initial weight values, which are defined by the user at the beginning of a learning process, also affect the solution found by the learning algorithm in that they define the starting point of the search for global minimum. As no learning happens when all the weights are set to zero, a uniform range is defined for the weight values, generally plus and minus 0.5 or less. It is reported by Blamire (1996), Ardö *et al.* (1997) and Skidmore *et al.* (1997) that the initial values of the weights affect the accuracy of the classification significantly. However, there is currently no solution other than trial and error to prevent an ANN from becoming trapped in a local minimum by changing the initial weights, which is the rule of thumb. The reason to begin the search for global minimum each time from a set of randomly-determined weights is to start from different parts of multi-dimensional error surface, the dimension of which is defined by the number of weights in the network. Due to the effect of random initial weights, required training times and the resulting classification accuracies could be different. This problem has been investigated by some researchers, but there is currently no universal solution for this problem.

Another crucial question is the number of iterations required for a particular problem. On the one hand, fewer iterations than the required number lead to a network that cannot learn data well and thus produce accurate classification

results. On the other hand, more iterations than the required number tends to make the network overspecific and thus the network loses its generalisation capabilities, in that it cannot classify data outside the range of the training data with high accuracy. The solution for determining the optimum number of iterations is to employ a convergence (or stopping) criterion, such as using a validation set or employing an early stopping rule, as discussed earlier. Another solution to this problem could be to systematically save networks during the training stage. Thus, it will be possible to test the performance of these networks and decide the most appropriate one.

Another choice that has to be made at the beginning of an application is the nature of the transfer functions for the hidden and output layers. There are many functions available for this purpose, but none of them is found to be superior to the rest. Although the sigmoid function has been mainly preferred and used in the literature, recent studies show that using 'tanh' gives better and faster convergence. The impact of transfer functions on network learning is not exactly known, and also needs further examination.

The learning rate is another internal parameter affecting the performance of the network. It plays a major role in determining the magnitude of the alterations made to the weights in the network at each iteration. There are no clear descriptions in the literature of the exact nature of the learning rate. If the learning rate is not suitable for an application, it may increase the time for the network to learn from the training data and result in failure to learn the characteristics of the data.

There is no doubt that the biggest disadvantage of artificial neural networks (ANNs) is the computation time necessary for training the network. The reason for being computationally demanding and slow is that iterative gradient descent algorithms, such as backpropagation, are employed. The backpropagation method gets much slower if input units are highly dependent on each other (correlated) so that adapting one disturbs the other. Another reason for a long training process is using non-optimum learning rate and momentum values. To speed up backpropagation three remedies are recommend in the literature: using a 'tanh'

activation function instead of a sigmoid, employing an adaptive learning rate instead of constant one, and rescaling the input variables. Three most successful variants of backpropagation learning are QUICKPROP (Fahlman, 1988), RPROP (Riedmiller and Braun, 1993) and CEN-BP (Joost and Schiffmann, 1998). In order to reduce the long training time for multilayer perceptrons, some new learning techniques have also been introduced, for example fast learning by Dawson *et al.* (1994), genetic learning by Zhou and Civco (1996), and dynamic learning by Chen *et al.* (1995a, 1995b).

In order to prevent neural networks from taking an excessively long time to train due to the presence of some atypical pixels, a low-pass filter, such as a 3 by 3 mean filter, can be used to eliminate the negative effects of those pixels. This filter also introduces texture information to the network. Using texture information can also help to improve the accuracy of the classification. For example, Paola (1994) found that using a median filter increased the accuracy of the maximum likelihood classifier from 89.5 to 92.4%, and of the neural network classifier from 96.2 to 98.5%. Also adding texture resulted in fewer iterations and faster convergence times. Similar results are reported by Bruzzone *et al.* (1997). However, despite speeding up the convergence and reducing noise effects, the network will be more powerful if it is trained with raw data including some noisy pixels as recommended by Sietsma and Dow (1991). Moreover, adding texture information increases the number of computations significantly; for example, if a 3 by 3 window is used, instead of one node, nine input nodes will be employed in the input layer.

### 3.10 Summary

The fundamentals of artificial neural networks (ANNs), particularly the multilayer perceptron (MLP), are discussed in this chapter. The main components of an ANN, such as learning algorithms and transfer functions, are described in detail. It is emphasised that, although the ANN approach can give considerably better results than conventional statistical classifiers, it has some handicaps (or problems) that should be taken into consideration. These handicaps are described

in detail so as to provide clear definitions of the problems that users can face. This may certainly help to get some insight into the behaviour of artificial neural networks. Since the major drawback of artificial neural networks for new users is to determine the optimum network structure for a particular problem, emphasis has been placed to network design strategies, constructive and pruning techniques. The information given in this chapter is important in order to understand the concepts to be discussed in following chapters.

As the artificial neural network approach is relatively new and many characteristics of them are still unknown or under investigation, there are some open questions with regard to utilising them efficiently. These questions are summarised by Wilkinson (1997) as follows:

- Do neural networks really offer significant advantages compared to other pattern recognition and data transformation algorithms ?
- Classification has been the main application for neural networks in Earth observation but has research on classification reached an impasse imposed by extraneous factors such as quality of ground data, or lack of possibility of precise class definitions ?
- Is it necessary or even possible to construct a very large modular neural network (VLNN) to encode landscape characteristics of the whole of Europe (i.e. to create a 'pan-European classifier' which can describe local conditions and avoid the generalisation problem) ?
- Is special purpose hardware really needed to exploit neural networks in a realistic way in remote sensing in an operational context ?
- Should new or less common neural network models and architectures be explored for use in remote sensing or can the existing commonly-used models such as MLP offer as much functionality as is likely to be required for most practical purposes ?
- Are there any novel applications of neural networks in remote sensing that have so far not been considered ?

Some authors, such as Duguay and Peddle (1996) and Ardö *et al.* (1997) claim that ANNs do not give significantly better results than other classification techniques, whereas most authors have reported that ANNs give better results. This brings the question of whether ANNs are best for all classification tasks or not. This question should be answered, especially by comparing ANN-based methods to new, powerful, techniques such as genetic algorithms and evidential reasoning (Davis, 1987) as well as the decision trees. It is obvious that ANNs will be saved from their black-box definition when all the above questions are answered. In the next chapters of this thesis their behaviour in terms of the effect of network structure and the learning parameters, which are set at the beginning of the learning process, are thoroughly investigated. In order to perform these investigations a number of visualisation techniques are put into practice through programs written in MATLAB software package. In other words, an attempt is made in this study to answer some of the important questions by carrying out many analyses on networks using scientific visualisation techniques.

## CHAPTER IV

# VISUALISATION OF HIGH-DIMENSIONAL DATA

### 4.1 Introduction

Many applications require data that are inherently multi-dimensional in nature. Specifically, in the area of remote sensing, the analysis of remotely sensed image data requires processing of multi-band and multi-temporal data, which are described by a large number of features. The representation and processing of such data demand large memory and processing time in a computer. On the other hand, visual analysis of such data can greatly help analysts understand the internal structure of the data by exploring patterns, identifying trends, and comparing complex information.

Understanding the structure of a dataset is often a difficult task, especially when the data represent complex phenomena, characterised by many variables. If such a dataset is to be explored, one of the first steps is to visualise it on a plane (typically on a computer screen) either in a two or a three dimensional representation in order to gain some insight into the data structures and to understand the relations present in the data. In fact, by mapping the data onto a low-dimensional space it is usually possible to recognise some important relations, such as finding clusters of related data points and detecting outlying or atypical points.

Visualisation and analysis of three (or higher) dimensional data is more difficult to implement and interpret than the case of two-dimensional data. In the case of three (or higher) dimensional data, a viewer is asked to construct a three (or higher) dimensional mental image of the space containing the data points, which are mapped onto a two-dimensional screen. Since we live in a three-dimensional environment, it is not possible for us to imagine and to visualise the geometric relations in higher dimensions.

Two groups of techniques are described in the literature to visualise multi-dimensional data. The first group is designed to map the data with their original values onto a plane. These techniques are called graphical analysis techniques as they are generally based on some sorts of graphical representation. Chernoff faces, parallel coordinate plots, and Andrews' plots are the major methods that are used to depict multi-dimensional data. By using these techniques it is possible to exactly represent each pattern as a picture with  $n$  degrees of freedom. The second group of techniques is based on the reconstruction of high-dimensional data in a lower dimensional subspace by reducing the number of dimensions to two or three while minimising some error function. These techniques are called projection techniques, or mapping algorithms. They aim to determine a new configuration of points in a lower dimensional subspace that represents the structure of the original data as faithfully as possible. Both groups of visualisation techniques are discussed in the following section.

## **4.2 Graphical Visualisation Techniques**

### **4.2.1 Chernoff Faces**

Chernoff (1973) brought up the idea of using cartoon faces as symbols to represent data values coded into the facial characteristics such as curvature of smile, angle of eyebrow, shape of the face, location of eyes and mouth. In other words, each dimension of the data determines the size, location and shape of some component of a cartoon face. According to Chernoff (1973), 'the purpose is to

allow viewers to draw conclusions on their vast experience by interpreting facial expressions at glance'. The representation of high-dimensional data by using faces may be more useful than others, since people are used to studying and reacting to faces, and they are able to ignore insignificant characteristics and focus on the potentially important features.

One major advantage of using faces is their inherent meaningfulness. For example, in an economic data analysis the curvature of the mouth may be used to represent the richness of a country, and so one could easily recognise rich countries as being represented by faces with a broad smile, whereas poor countries will be depicted by sad faces. Comparative studies have shown that the faces are more easily memorised. It has been also pointed out that faces form more memorable stimuli in a paired-associate learning task than polygons or arrays of numbers.

Discussions of the use of Chernoff faces for representing multi-dimensional data indicate that, firstly, it is a fact that some features of the face may be more informative than others. For example, certain observers may concentrate on the eyes, while others focus on the chin. Secondly, different observers may use different features of the faces to judge their similarity. Moreover, certain features are more readily seen than others are and so these will obviously be more informative. These issues indicate the subjectiveness of the observer's judgement. A possible procedure recommended to overcome such problems is to produce several sets of faces for the data. Detailed information and applications can be found in Chernoff (1973), Everitt (1978), Everitt and Dunn (1991) and Everitt and Nicholls (1975), in which Chernoff faces, Andrews' plots and Sammon's Nonlinear Mapping algorithm are compared.

The data in Table 4.1, adopted from Hartigan (1975), shows the number of crimes of different types per 100,000 population in some US cities. A representation of these data as cartoon faces is illustrated in Figure 4.1. As can be seen, representation using Chernoff faces gives greater insights and allows the analyst to effectively interpret the dataset.

Table 4.1 Crime data of US cities (from Hartigan, 1975).

City	Murder	Rape	Robber y	Assault	Burglary	Larceny	Auto theft
Atlanta	16.5	24.8	106	147	1112	905	494
Boston	4.2	13.3	122	90	983	669	954
Chicago	11.6	24.7	340	242	808	609	645
Dallas	18.1	34.2	184	293	1668	901	602
Denver	6.9	41.5	173	191	1534	1368	780
Detroit	13.0	35.7	477	220	1566	1183	788
Hartford	2.5	8.8	68	103	1017	724	468
Houston	16.8	26.6	289	186	1509	787	697
Kansas C.	10.8	43.2	255	226	1494	955	765
LA	9.7	51.8	286	355	1902	1386	862
New Orleans	10.3	39.7	266	283	1056	1036	776
New York	9.4	19.4	522	267	1674	1392	848
Portland	5.0	23.0	157	144	1530	1281	488
Tucson	5.1	22.9	85	148	1206	756	483
Washington	12.5	27.6	524	217	1496	1003	739

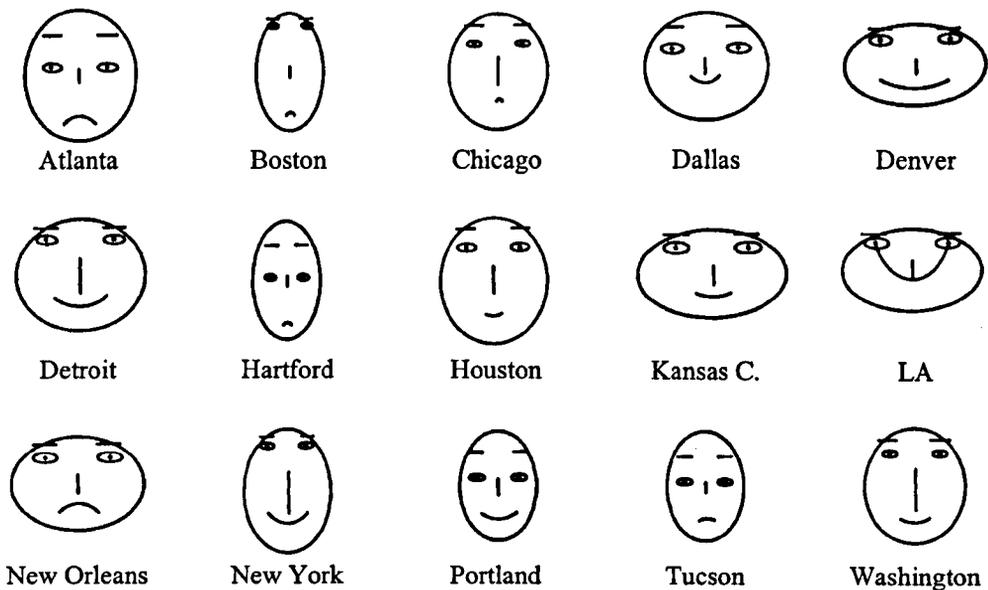


Figure 4.1 Chernoff faces for US city crime data.

The correspondence between face features and each variable, crime type, is as follows:

- 1) Murder: area of face,
- 2) Rape: shape of face,
- 3) Robbery: length of nose,

- 4) Assault: location of mouth,
- 5) Burglary: curve of smile,
- 6) Larceny: width of mouth,
- 7) Auto theft: separation of eyes (length between eyes and eyebrows).

### 4.2.2 Parallel Coordinate Plots

The parallel coordinate display can be thought of an extension, or generalisation of a two-dimensional Cartesian plot. The idea is to set all the axes as parallel to each other in a two-dimensional Cartesian plot such that the whole dataset is displayed. Thus, a planar diagram, in which  $n$ -dimensional data points are represented in a unique way, is obtained. Wegman (1990) discusses the details of the technique together with its extensions.

In parallel coordinate plots, a vector  $(x_1, x_2, \dots, x_n)$  is created by plotting each  $x$  value on a different axis. Then, these points are joined together by a line. In other words, every multi-dimensional data point is represented by a line in terms of its value for each dimension. Figure 4.2 illustrates two points plotted in parallel coordinate diagram. The main advantage of the method is representing each multi-dimensional data point in the same planar system. This considerably helps the observer to examine the characteristics of the data.

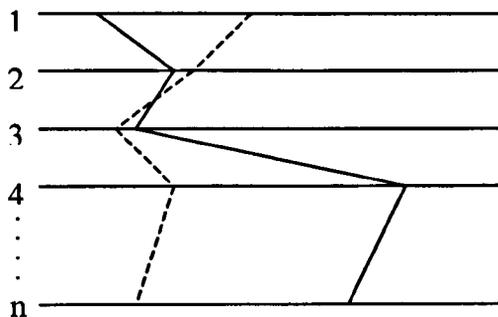


Figure 4.2 Parallel coordinate representation of two  $n$ -dimensional points.

It is easy to determine uncorrelated data points in a parallel coordinate representation, since they are displayed as distinct lines compared to overall trends

of inherent clusters. One-dimensional projections of the data are obtained by the individual parallel coordinate axes. Therefore, separation on any axis portrays a view of the data that allows the detection of clustering. Owing to the high reliability of the multidimensional parallel coordinate plot, it is generally easy to see whether the clustering propagates through other dimensions.

Some of the data analysis features of the parallel coordinate representation include the ability to diagnose one-dimensional features such as marginal densities, two-dimensional features, such as correlations, and nonlinear structures, and multi-dimensional features such as clustering, hyperplanes, and the modes (Wegman and Luo, 1997). These issues and some extensions of parallel coordinate plots are discussed in detail in Wegman (1990).

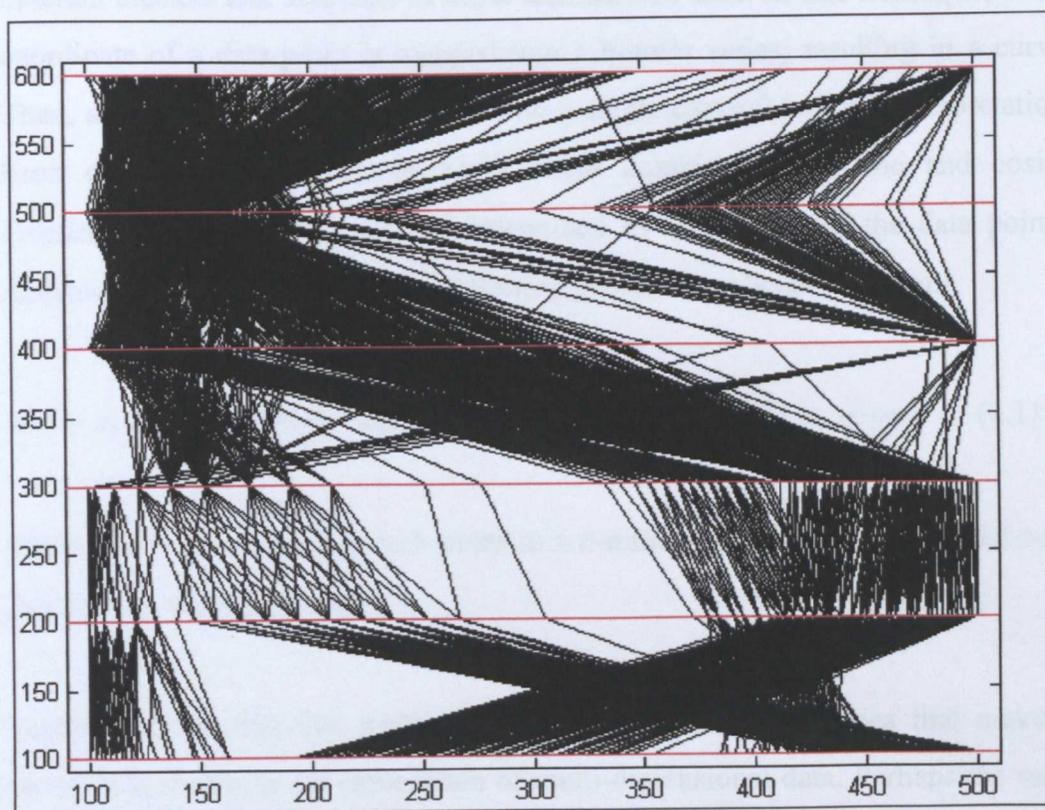


Figure 4.3 Parallel coordinate representation of Elveden dataset.

The main drawback of the technique appears when a large number of dataset including large number of clusters is to be visualised. In such a case, it is hard to distinguish clusters, as in Figure 4.3, which is the parallel coordinate

representation of the Elveden dataset containing 700 pixel values in six dimensions, and seven distinct classes. To overcome this problem, one solution discussed by Wegman (1990), Miller and Wegman (1991), and Wegman and Luo (1997) is the use of parallel coordinate density plots where the parallel coordinate plot is replaced with its density, estimated using average shifted histograms.

### 4.2.3 Andrews' Plots

A very simple technique, introduced by Andrews in 1972, has been used to obtain a visual representation of multivariate data in which each multi-dimensional data sample is mapped into a function that is in an orthogonal sinusoidal form. This technique has been used for many applications and found useful for identifying inherent clusters and atypicals in multi-dimensional data. In this technique, every coordinate of a data point is mapped into a Fourier series, resulting in a curve. Then, all the curves are superimposed on a single curve for visual interpretation. Each observation is presented by a linear combination of sine and cosine functions, whose coefficients are determined by the values of the data points. Andrews (1972) defines his simple plotting procedure as below function,

$$x(t) = x_1 / \sqrt{2} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \dots \quad (4.1)$$

where  $x' = [x_1, x_2, \dots, x_n]$  each point in a  $n$ -dimensional space. This function is plotted over the range  $-\pi \leq t \leq \pi$ .

Andrews shows that this particular function has many properties that make it particularly useful in the exploration of multi-dimensional data. Perhaps the most important of these is that this representation preserves Euclidean distances. As a consequence, points that lie close together in the original  $n$ -dimensional space will be presented by lines on the plot that are close to each other, whilst distant points will be represented by lines that remain apart for at least some value of  $t$ . This property enables the plots to be used for the possible identification of clusters, atypical points, or other peculiarities of the data.

A problem that arises when using this technique is that only a fairly limited number of data may be plotted on the same diagram before it becomes too confusing. Various procedures might be adopted in order to overcome this problem. For example, first, a plot of all the data could be produced to assess the general characteristics of the data. This could be followed by separate plots of each set of subset plots of 10-20 points; then, these plots can be examined and compared to the whole. Alternatively, selected quantities or percentage points of the distribution of the  $n$  values could be plotted along with the curves of selected individual data points. Gnanadesikan (1977) suggests such an idea, based on using only selected quantiles or percentage points (e.g. median, upper, and lower quartiles) and calls it a *quantile contour plot*.

When the form of the function involved in Andrews' plots is examined, it can be easily observed that the original variables are not equally weighted. Some are associated with cyclic components having a high frequency, others with components having a low frequency. Since in these plots low-frequency components are more informative than those with high frequencies, it may be useful to associate  $x_1$  with the variable considered, in some sense, to be the most important,  $x_2$  with the second most important, and so on. In the absence of any firm ideas as to such an order of variables, it may be useful to apply Andrews's technique not to the raw data but to the transformed variables obtained for example from principal components analysis, since these will automatically be in order of decreasing importance in a particular sense (Everitt, 1978).

Another problem with Andrews's technique is that, due to the composite structure of each point's function, it is not possible to observe the effects of variables separately. While Figure 4.4 shows five points in eight dimensional space, Figure 4.5 displays the result of Andrews's plot for Elveden dataset. It is very easy to interpret the characteristics of the data from Figure 4.4, whereas Figure 4.5 is hard to understand due to the complex structure of the curves, and the difficulty of recognising individual curves within the general pattern.

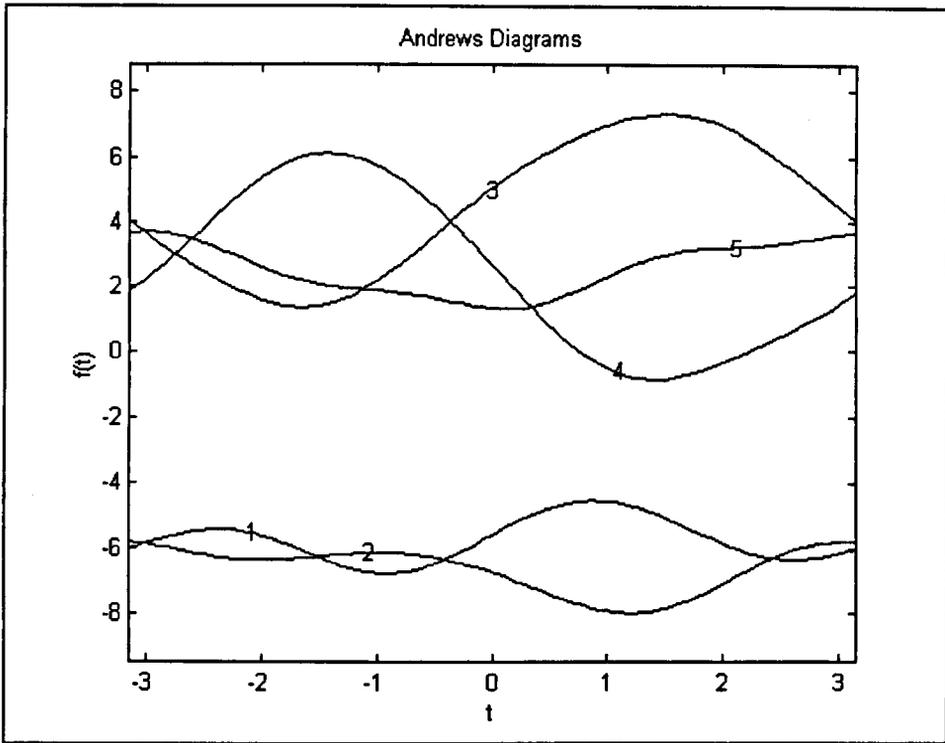


Figure 4.4 Andrews' diagram of five eight-dimensional data vectors.

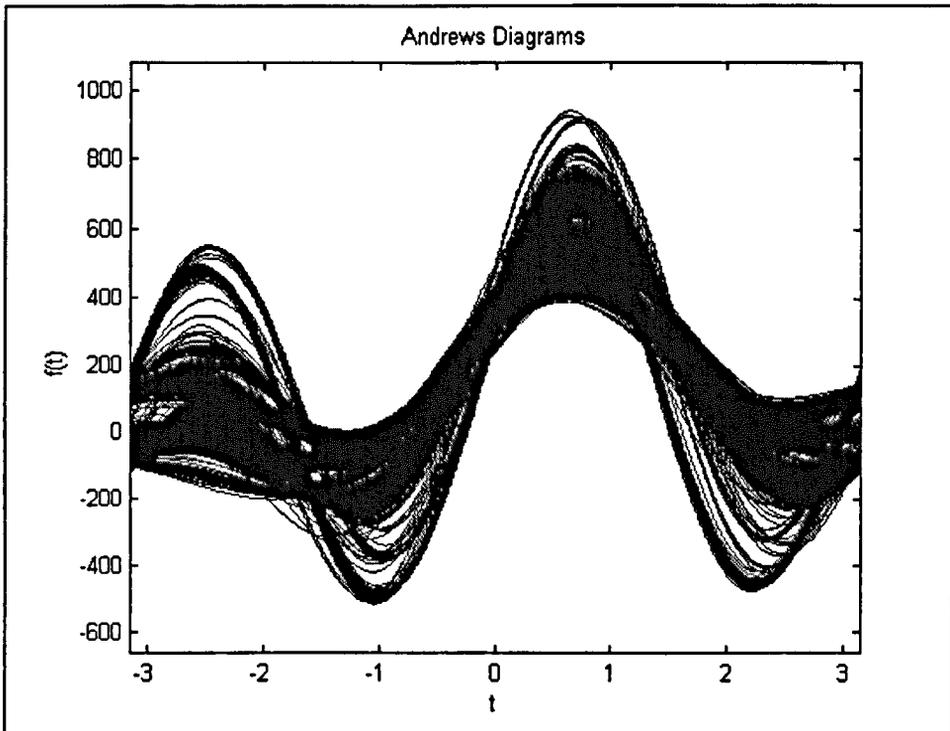


Figure 4.5 Andrews' diagram for Elveden dataset.

The main disadvantage of all graphical multivariate data representation techniques is that they are limited by the size of the datasets. They can work well with

relatively small datasets. They can be, however, confusing (not useful) when they are applied to large amount of data. It is difficult to imagine a large image having multi-band and multi-temporal data being adequately represented by Chernoff faces. Using subsets of the datasets would be the natural solution for the problem. The methods could, however, be quite useful in gaining some overall view of the data.

### **4.3 Projection Methods**

The methods that are used to map multi-dimensional data onto a lower dimensional subspace are called projection methods. Such methods can be considered as dimensionality reduction methods as well as mapping algorithms. They help to visualise any underlying structure present in the data, and examine the characteristics of the dataset. The main objective of the projection methods is to preserve the geometric relationships among the patterns in the original space as much as possible.

Projection techniques can be divided into two groups, termed the linear and nonlinear projection techniques. The main difference between these techniques is that, while linear methods search a linear subspace such as a line or a plane for projection, nonlinear methods try to find a nonlinear subspace. In addition, nonlinear methods are based on some kinds of preservation criteria such as preservation of all the distances between points. Figure 4.6, adapted from Pekalska (1998), illustrates the difference between linear and nonlinear projection methods for two-dimensional case.

As can be seen from Figure 4.6, points are at equal distances from each other, but the linear method projects them onto a line and does not preserve distances between points. Conversely, in the case of nonlinear projection, a nonlinear one-dimensional curve is used and as a result, nearest neighbour distances are preserved.

Linear techniques are generally used because of their simplicity, generality and speed. Furthermore, these methods are mathematically well defined. On the other hand, the mathematical basis of nonlinear techniques is generally more complicated and their implementation uses mostly heuristic algorithms. These techniques are used when linear methods are unable to preserve inherent complex data structures. Perhaps the most important characteristic of these techniques is that they are data-dependent and they do not make any assumptions, specifically about the frequency distribution of the data. On the other hand, they have some drawbacks, the most important of which is that new data cannot be placed into the low-dimensional subspace without recomputing all of the pattern coordinates.

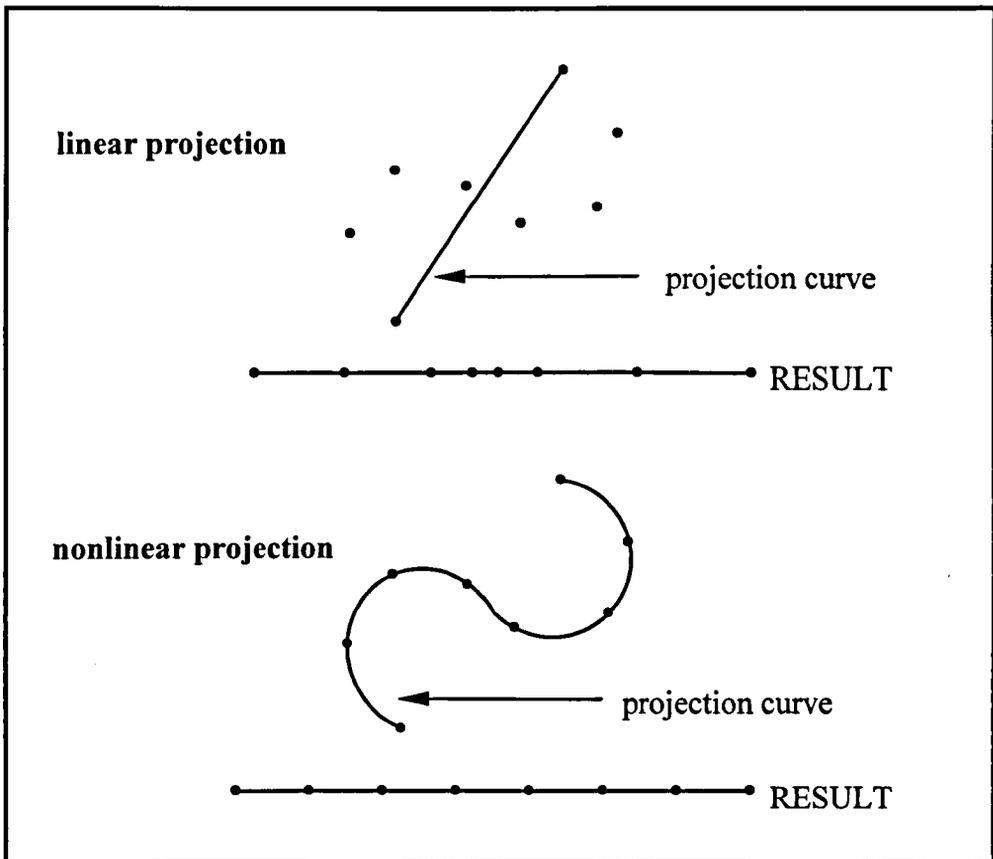


Figure 4.6 The difference between a linear and nonlinear projection method.

The main linear projection methods that are discussed below are: Principal Components Analysis and Factor Analysis, while the major nonlinear techniques discussed are Multidimensional Scaling, Sammon's Nonlinear Mapping, Self-Organizing Map, and Auto-associative Feed-forward Artificial Neural Networks.

### 4.3.1 Linear Projection Techniques

#### 4.3.1.1 Principal Components Analysis

Principal Components Analysis (PCA) is one of the oldest and perhaps the most commonly used technique for analysing multi-dimensional and multivariate data. It has been implemented mainly for feature extraction, data compression, and multivariate data projection. It has been used as a standard tool in many areas, such as communication, signal and image processing, pattern recognition and data analysis (Everitt and Dunn, 1991; Azimi-Sadjadi *et al.*, 1993; Bateson and Curtiss, 1996). Specifically, PCA is a widely used technique in the analysis of remotely sensed images. A principal components transformation of a multi-spectral image is performed to remove or reduce the amount of redundant information resulting from the correlation between the spectral bands (Mather, 1999a; Lillesand and Kiefer, 1994). PCA is also used for change detection and land cover characterisation of multi-spectral images (Fung and LeDrew, 1987; Hirosawa *et al.*, 1996; Picchiotti *et al.*, 1997).

PCA is a linear orthogonal transformation that projects an  $N$ -dimensional input space to a  $d$ -dimensional space, where  $d \leq N$ . The coordinate vectors produced by PCA for the  $d$ -dimensional space are uncorrelated since coordinates axes are set to be orthogonal. By using PCA, it is possible to represent large amount of variance of the original data in a smaller number of dimensions. The most important advantage of the technique is that original values can be reduced to new components, which are fewer in number, with the least possible loss of information.

In applications of PCA, a set of data is redescribed to get a smaller number of components, which can be thought of as composite variables. The estimated principal components are normally ranked in decreasing order of importance. In other words, the first component shows the most important dimension of variation in the dataset, whilst the second component describes the most important

dimension of variation in the data after the effects of the first principal component have been removed. Components can be interpreted by examining the component loadings, which identify the relative positions of the variables along the new component axes. The 'descriptive power' or 'strength' of a component is defined by the corresponding eigenvalue, which can be described as the percentage of variability in the dataset that can be accounted for by the component.

In PCA applications, the user has a choice of basing the analysis on the correlation matrix or the covariance matrix. It is very important to understand the difference between these alternatives. If the correlation matrix is chosen, as the basis for a PCA, then the variables are standardised to zero mean and unit standard deviation. Thus, the same weight, regardless of their actual variability, is assigned to all variables. If the covariance matrix is selected, different weights are assigned to the variables with respect to their variances. If the variables are measured in different units, or scales, the only choice is to use the correlation matrix for principal components analysis. If all the variables are measured in the same units, then the user can choose to base the PCA either on the correlation or the covariance matrix. However, care must be exercised as the fundamental aim of PCA is to partition the variance of a dataset, and the use of the covariance matrix implies that the total variance around the mean of each variable is included.

The identification of principal components is to some extent arbitrary in that completely different results may be produced by analysing the covariance matrix rather than correlation matrix. This indicates that it could be misleading to try to allocate too much meaning to components in many situations. Therefore, it may be more helpful to use PCA for reducing the dimensionality of the data in order to provide a starting point for further investigations.

If the first two or three components account for a large proportion of the total variance, then these components can be projected onto a plane to produce two or three-dimensional representations of the data. These plots may reflect the main structure of the original data. An important question is how many components are

needed to provide an adequate summary of a given dataset. The general tendency is to select components having corresponding eigenvalues greater than 1.0 if the correlation matrix is used. The underlying idea of this choice is that a component with a eigenvalue of 1.0 contains as much information (as useful as) any one of the original standardised variables. Therefore, there is no point in selecting components that contain less information than a single original variable. Another idea is that it is enough to use components having 70-90 percent of the total variation. However, the best way to find out the appropriate number of principal components to represent the inherent structure of the data could be using a 'scree diagram', which is an eigenvalue plot. Starting with the first component, the line connecting the eigenvalues of the covariance or correlation matrix is initially a steeply downward one, then slowly becomes an approximately horizontal line. The point where the curve first begins to straighten out is considered to indicate the number of useful components. Figure 4.7 shows the scree plot of the eigenvalues of the components for Elveden dataset.

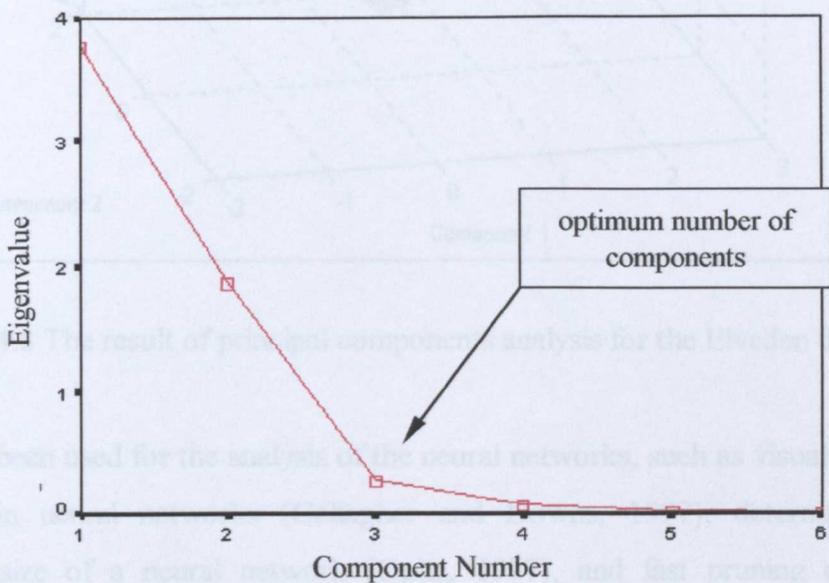


Figure 4.7 Scree diagram of principal components for Elveden dataset.

As emphasised by Bailey and Gatrell (1995), there is no guarantee that the directions which maximally separate observations in attribute space (as identified by the principal components) will necessarily be those that correspond to the configuration of observations in geographical space. Indeed, in general, this is

unlikely. Therefore, PCA will not necessarily be of use if the objective of the analysis is to determine which combinations of attributes demonstrate the most significant spatial pattern.

Figure 4.8 illustrates the results of PCA applied to the Elveden dataset, which consists of 700 pixel values in six dimensions representing seven land cover classes.

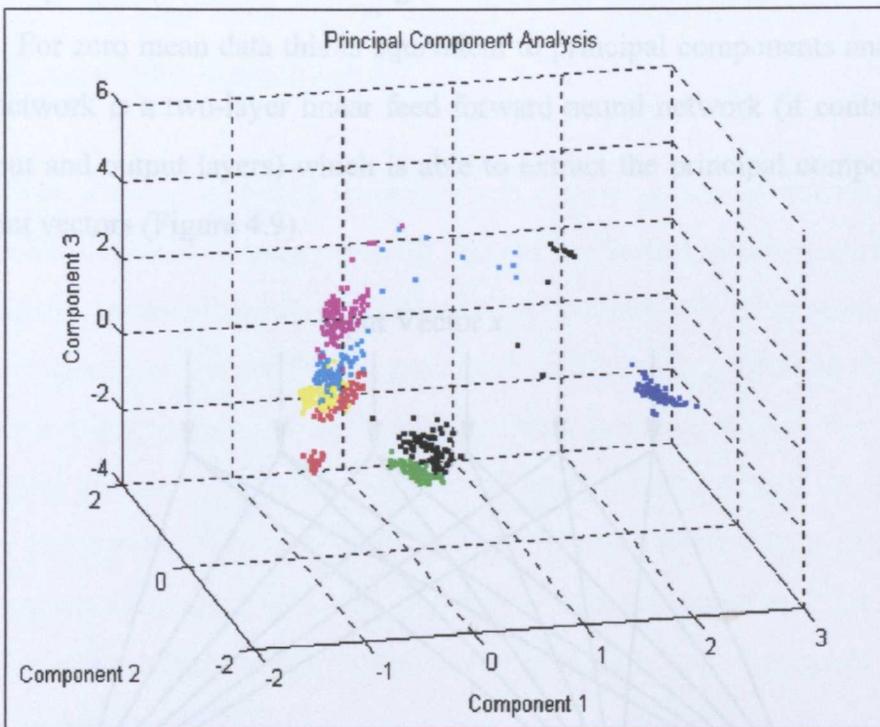


Figure 4.8 The result of principal components analysis for the Elveden dataset.

PCA has been used for the analysis of the neural networks, such as visualisation of learning in neural networks (Gallagher and Downs, 1997), determining the effective size of a neural network (Opitz, 1997), and fast pruning of neural networks (Levin *et al.*, 1994). There is an increasing interest in extending unsupervised neural network learning algorithms to implement Principal Components Analysis (PCA). These types of networks are called Principal Components Analysis Networks, and a number of them are discussed in the literature. Some of them are based on the “Oja rule”, whereas the others are based on an auto-associative bottleneck neural network.

The idea of using Hebbian unsupervised learning algorithm for PCA networks was first proposed by Oja and discussed in his several papers (Oja, 1995a, 1995b; Oja and Karhunen, 1995; Oja *et al.*, 1995). Because of Oja's primary work, the main learning algorithm for such networks is called Oja's learning rule. It is simply a procedure for Hebbian learning with constrained weight vector growth. This procedure adds a weight decay proportional to the squared value of the output. Oja's rule finds a unit weight vector that maximises the mean square output. For zero mean data this is equivalent to principal components analysis. A PCA network is a two-layer linear feed forward neural network (it contains only the input and output layers) which is able to extract the principal components of the input vectors (Figure 4.9).

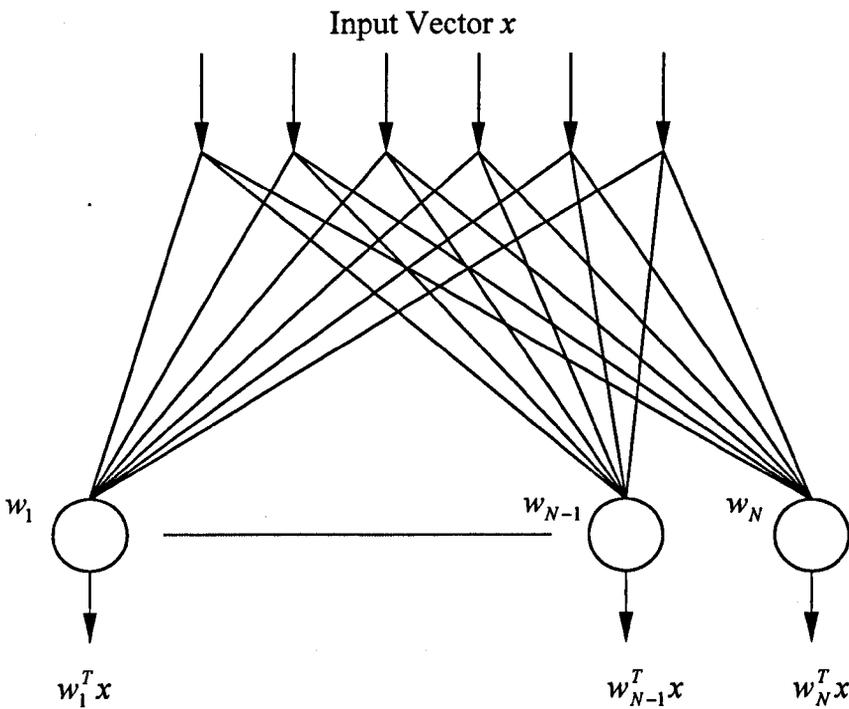


Figure 4.9 The basic structure of a linear PCA network (from Oja, 1995a)

The PCA network has the ability to handle slowly varying statistics in the input data, maintaining its optimality when the statistical properties of the inputs do not stay constant. Such a network using a Hebbian learning algorithm is potentially useful for signal characterisation, feature extraction and data compression. Several

different versions and extensions of the PCA network exist, each with different learning strategies. These strategies are, however, all based on a form of Hebbian learning, which is the basis of many unsupervised learning algorithms. The general form for Hebbian learning is:

$$W_{n+1} = W_n + \eta_n x_n y_n^T \quad (4.2)$$

where  $\eta_n$  is a small update factor,  $y$  represents the node values of the output layer,  $x$  represents the node values of the input layer,  $W$  represents the weights in the network, and  $n$  is the iteration number.

In the field of neural networks, there has been a growing interest in extending the unsupervised Hebbian learning rules in PCA to nonlinear Hebbian learning rules. Such techniques are often called nonlinear PCA methods. The main reason for this interest is that, even though PCA is optimal in approximating the input data in the mean-square error sense, the representation that it provides is often not the most meaningful in describing some fundamental properties of the data. In PCA, the data are transformed to an orthogonal basis that is determined only by the second-order statistics (covariances) of the input data. Developments of PCA methods take into account higher-order statistics and thus may better represent the data (Oja and Karhunen, 1995).

Another type of neural network is the auto-associative neural network (AANN), which learns a task using a back-propagation learning algorithm in unsupervised auto-associative mode. By introducing nonlinearity to the process, such networks are used for nonlinear principal components analysis, and they are also, as a result, called nonlinear principal components analysis neural networks. The AANN architecture consists of an input layer, three hidden layers, and an output layer (Figure 4.10). These networks have the same number of nodes in input and output layers, and are trained to reproduce the input values at the output nodes. The first of the hidden layers is the *mapping layer* with a dimension (number of neurons) greater than the number of input/outputs. The second hidden layer is called the *bottleneck layer* and the dimension of this layer must be smaller than the number

of inputs (and outputs), otherwise, the network would simply copy the inputs to the outputs. The smaller hidden layer forces the neural network to learn any relationships within the input data, and compresses the input data to a number of parameters equal to the size of the middle layer. Therefore, of the three hidden layers, the bottleneck layer plays the key role in identifying the mapping. The third hidden layer is called the *demapping layer* and has the same dimension as the first hidden layer. Kramer (1991) states that the five-layer neural network structure is necessary to model non-linear processes. The mapping layer maps from input data space to the non-linear principal component space (bottleneck layer), and the demapping layer map from the non-linear principal component space to the data space, which is the network output.

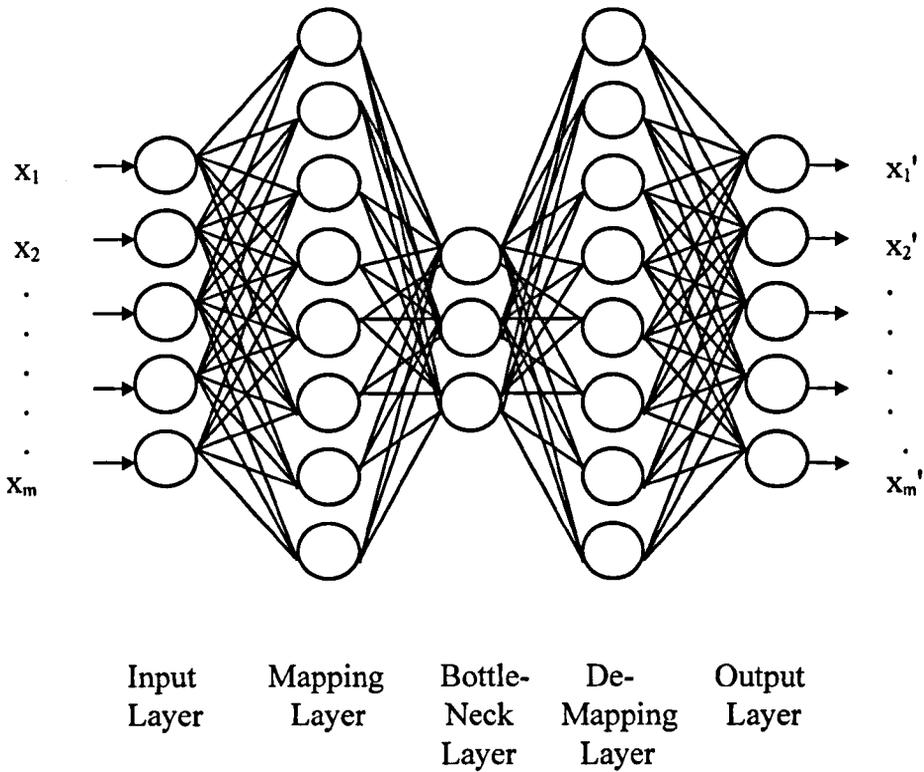


Figure 4.10 Architecture of a five-layer bottlenecked AANN. Note that some of the links are left out for clarity.

This unique network structure forces the network to develop a compact representation of the training data that better models the underlying system parameters. The bottleneck layer works like a nonlinear principal component

filter. Such networks use nonlinear activation functions, typically a sigmoidal function, in the hidden layer section. It has been reported that AANN gives considerably better results than linear networks using Oja's rule. On the other hand, Oja (1995a) underlines that a five-layer fully nonlinear network may be problematic to train by backpropagation, especially as the second and fourth layers may have to be large. Also the generalisation ability of the five-layer network may not be as good as that obtained by the linear PCA. Details of AANN can be found in Kramer (1991) and Oja (1995a).

An interesting and useful extension of PCA is Independent Component Analysis (ICA), which has been widely discussed in the past few years. ICA is a linear transformation of data such that the components become statistically independent. Instead of requiring that the coefficients of a linear expansion of the data vectors be uncorrelated, in ICA the coefficients must be mutually independent or as independent as possible. It has been reported that ICA provides in many cases a more meaningful representation of data than PCA. Discussion of the principles and use of ICA can be found in Comon (1994).

Two comparative studies carried out by Mao and Jain (1995) and De Backer *et al.* (1998) compare some of the linear and nonlinear projection techniques discussed above. In addition, Plumbley (1991) gives a mathematical description of major learning algorithms for linear PCA networks. He also discusses the problems, including information lost by PCA and scaling problems in PCA, inherent in the statistical PCA method. The problem of scaling in PCA is also discussed in Chatfield and Collins (1980).

#### **4.3.1.2 Factor Analysis**

Factor Analysis is also an orthogonal transformation method, which estimates an optimum configuration for points of high-dimensional space into a lower dimension. It has been mainly used for educational and business-related research, such as analyses of questionnaire responses and test scores. The method is focused

on whether the covariances or correlations between a set of observed variables can be described with regard to a smaller number of unobservable, latent variables, assuming that the correlation between each pair of observed variables results from their mutual association with the latent variables.

Factor analysis is quite similar to principal components analysis in that they both seek to project multi-dimensional data into a subspace of lower dimension, using the correlation or covariance matrix. Due to similar types of processes and outputs, factor analysis and principal components analysis are sometimes confused. However, they differ both conceptually and mathematically. As highlighted by Bailey and Gatrell (1995), factor analysis is based on an assumption that the observed correlations between the attributes are mainly the result of some *a priori* underlying regularity or structure in the data, rather than one that is defined purely on the basis of mathematical criteria, such as maximising the variance or 'separation' of observations, as in principal components analysis. More specifically, an *a priori* model is proposed whereby each of the observed variables is assumed indirectly and partially to measure a fixed number of pre-defined characteristics or latent factors, which cannot themselves be directly measured. In short, factor analysis attempts to explain correlations in the original variables with regard to a model that proposes a certain number of unobservable 'common factors'.

The aims of the analysis are to identify the number of latent factors, their relative order, and their relations with the observed data. After identifying the separate dimensions of the data structure, factor analysis can be used for two purposes, summarisation and data reduction. In summarising the data, factor analysis derives underlying dimensions that describe the data with a smaller number of concepts than the original individual variables. On the other hand, data reduction can be performed by calculating scores for each underlying dimension of the subspace and substituting them for the original variables.

Once calculated, the results of factor analysis can be used in a similar way to the results of principal components analysis. For example, two and three-dimensional views of the data can be prepared by using two and three factor solutions respectively.

It has been reported that factor analysis has probably attracted more critical comment than any other statistical technique because of its limitations. Two of these limitations are crucially important and are discussed shortly. The first limitation comes from the fact that factor analysis is used to describe complex matrices of correlations by factors chosen for completely mathematical reasons. However, an understanding of the phenomena being investigated should be the main criterion, as the mathematics alone cannot guarantee a 'correct' result. Another critical comment on factor analysis is that since factor loadings are not determined uniquely by the basic factor model, investigators can choose to rotate or transform factors in such a way as to get the answer they are looking for.

### **4.3.2 Nonlinear Projection Techniques**

#### **4.3.2.1 Multidimensional Scaling**

Multidimensional scaling (MDS) refers to a series of methods that are widely used, especially in behavioural, econometric and social sciences, to identify key dimensions underlying the data. The starting point of every MDS application is the estimation of a matrix that consists of the set of pairwise dissimilarities of the entities, or points. MDS searches a configuration of a low-dimensional representation of the data in order to locate a global minimum of the error ( $\min \sum (d'_i - d_i)^2$ ). The goodness of fit is estimated by a measure known as 'stress' that shows the relationships between the two rank orderings. All MDS algorithms work by minimising the stress, or error. The calculation can be carried out by using gradient descent, the simplex method, simulated annealing or some other optimisation technique.

There exist a variety of multidimensional scaling methods with slightly different cost functions and optimisation algorithms. The algorithms designed to analyse a single dissimilarity matrix can be grouped into two: metric and non-metric multidimensional scaling methods. The idea of metric MDS is to approximate the original set of distances by distances corresponding to a configuration of points in a lower dimension. Original distances can be estimated from the Euclidean distance or other types of distance measures. In other words, the aim of metric MDS is to derive a new set of points in a low-dimensional space such that corresponding inter-point distances are as close as possible to the original distances. Thus, the error defined as the sum of the squared differences between the true and approximated distances is minimised. Non-metric MDS was developed to meet the needs of users of ordinal scale data. It can also be used for data in a Euclidean space. In this case, instead of original distance values, MDS then only tries to preserve the rank order of the distances between points.

The first MDS method for metric data was developed in the 1930s and later generalised for analysing non-metric data. According to Everitt and Dunn (1991), 'the objective of MDS is to determine both the dimensionality of the model (that is the value of  $d$ ) and the position of the points in the resulting  $d$ -dimensional data, so that there is, in some sense, maximum correspondence between the observed proximities and the interpoint distances'. In other words, the larger the dissimilarity between two points, or the smaller their similarity, the further apart should be the points in the represented dimension.

A perfect reproduction of Euclidean distances may not always be the best possible goal, especially if the components of the data vectors are expressed on an ordinal scale. Then, only the rank order of the distances between the vectors is meaningful, not the exact values. The projection should try to match the rank order of the distances in the low-dimensional output space to the rank order in the original space. The best possible rank ordering for a given configuration, or points can be guaranteed by introducing a monotonically increasing function that acts on

the original distances, and always maps the distances to such values that best preserve the rank order (Kaski, 1997).

There are some problems in the use of MDS. For instance, between  $m$  data points there are  $m \cdot (m-1)/2$  distance relationships. MDS processes require time depending on  $m$ . In order to store the distances for  $10^6$  data points, more than 7 terabytes of memory (7 million megabytes) would be required. In addition, as data dimensionality increases, so MDS has more trouble finding a global minimum of the error function. To extend MDS so that it can be used with larger datasets, the idea of using subset-MDS is suggested. As noted by Alt (1990), the concept of similarity between objects is psychologically a difficult one that can lead to considerable problems in interpreting the results derived from Multidimensional Scaling.

#### 4.3.2.2 Sammon's Nonlinear Mapping

Sammon (1969) describes a widely used nonlinear mapping algorithm that has become very popular. In this algorithm the starting point is a random configuration of  $n$ -points in  $d$ -dimensions that correspond to  $n$ -points in an  $N$ -dimensional space ( $N \geq d$ ). The method of gradient descent is employed to reconfigure the points in the  $d$  dimensional space so that the mean square error between the original interpoint distances in  $N$ -dimensions and the interpoint distances in  $d$ -dimensions is minimised. This iterative algorithm stops either when the mapping error is below a user-defined threshold or when the user-defined number of iterations is completed and no convergence has occurred. Sammon's method involves a great amount of calculation, which results in the use of large memory space in computers.

In terms of its mathematical formulation, non-metric Multidimensional Scaling (MDS) and Sammon's nonlinear mapping (NLM) are similar. However, the mapping criteria 'stress' for MDS and 'mapping error' for NLM are different.

Another difference between the two is that MDS uses only the ordinal properties of the similarities or distances being used.

Sammon's nonlinear mapping algorithm is applied to reduce the dimensionality of the Elveden dataset, and to display seven land cover types in a three-dimensional view. The result is presented as Figure 4.11.

As Sammon's nonlinear mapping algorithm (NLM) is used in this study to reduce and map high-dimensional remotely sensed image data, the mathematical foundations of the technique are given in detail in the following paragraphs.

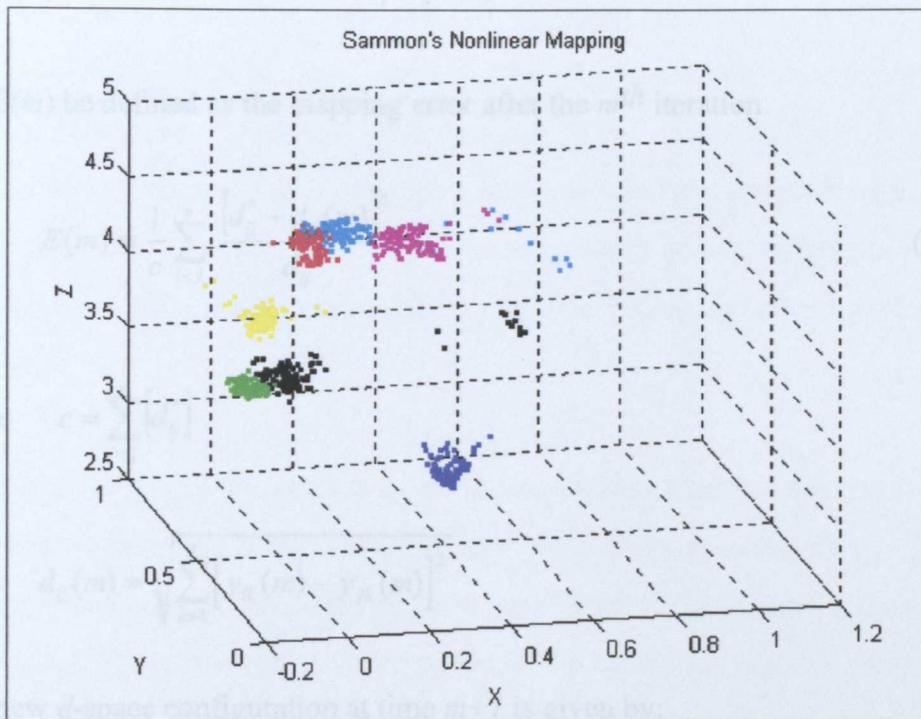


Figure 4.11 Result of Sammon's nonlinear mapping algorithm for Elveden data.

Let the original data exist as a set of  $n$  vectors in an  $N$ -space. Let there also exist a set of vectors in a  $d$ -space. The positions of the  $d$ -vectors are iteratively adjusted until their interdistances approximate as closely as possible to the corresponding  $N$ -space interdistances. The Euclidean distance between the vectors in the  $N$ -space is defined as  $d_{ij}^*$ , whilst the corresponding distance in the  $d$ -space is defined by  $d_{ij}$ .

The mapping error is then described as:

$$E = \frac{1}{\sum_{i<j} [d_{ij}^*]} \sum_{i<j} \frac{[d_{ij}^* - d_{ij}]^2}{d_{ij}^*} \quad (4.3)$$

The error is a function of the  $d \times n$  variables  $y_{pq}$ , where  $p = 1, \dots, n$  and  $q = 1, \dots, d$ . The next step is to adjust the point locations (i.e. change the configuration in the  $d$ -space) in order to reduce the mapping error. A steepest-descent procedure is used for this purpose;

Let  $E(m)$  be defined as the mapping error after the  $m^{\text{th}}$  iteration.

$$E(m) \equiv \frac{1}{c} \sum_{i<j} \frac{[d_{ij}^* - d_{ij}(m)]^2}{d_{ij}^*} \quad (4.4)$$

where  $c = \sum_{i<j} [d_{ij}^*]$

and,  $d_{ij}(m) = \sqrt{\sum_{k=1}^d [y_{ik}(m) - y_{jk}(m)]^2}$

The new  $d$ -space configuration at time  $m+1$  is given by:

$$y_{pq}(m+1) = y_{pq}(m) - (MF) \cdot \Delta_{pq}(m) \quad (4.5)$$

where,

$$\Delta_{pq}(m) = \frac{\partial E(m)}{\partial y_{pq}(m)} \bigg/ \left| \frac{\partial^2 E(m)}{\partial y_{pq}(m)^2} \right|$$

and MF is the “magic factor” (or step length) that was determined empirically to be  $MF \approx 0.3$  or  $0.4$  (Sammon, 1969).

The main problem with NLM is its computational requirements. Since the interdistance matrix, which contains  $n \cdot (n - 1) / 2$  elements ( $n$  is the number of data points), must be computed and stored, a large computer memory is needed. As a result, NLM gradient descent algorithms can be slow, especially for large datasets. Several modifications to Sammon's algorithms have been proposed to reduce the computational effort. Pykett (1978) introduced the idea of using of a clustering archetype (one for each class in the data) and the adjustments are carried out to only these archetypes rather than the entire set of pattern vectors.

The archetypes are defined as the centroids of each class. Thus, the computations are considerably faster than original NLM. Niemann and Weiss (1979) show that another difficulty with iterative methods like NLM is to find an algorithm with good convergence properties. The "magic factor" for NLM is determined empirically. Of course, a step size that is reasonable for one sample may be wrong for another one. Therefore, they suggested an iterative descent algorithm using an optimal step size in each iteration. This step size assures the convergence of the algorithm.

Mao and Jain (1995) proposed an unsupervised backpropagation learning algorithm to train a multilayer feed-forward neural network to simulate Sammon's nonlinear projection. The proposed learning algorithm, which needs no category information about patterns, is an extension of the backpropagation algorithm. The number of input nodes is set to the input dimensionality of the feature space, whilst the number of output units is specified as the dimensionality of the projected space. The mathematical basis together with a comparison with other neural network structures for projection methods can be also found in the study of Mao and Jain (1995).

#### **4.3.2.3 The Self-Organising Map**

The idea of the Self-Organizing Map (SOM) was introduced in 1981 by Teuvo Kohonen, who had a great influence on the development of Artificial Neural

Networks (ANNs). Kohonen's Self-Organizing Map has become one of the most popular artificial neural network models; in fact, it has been reported that the SOM is the most widely used unsupervised neural network model. It requires only an input dataset to learn and form its own output representation for a problem. The idea underlying the SOM is based on a model of the human sensory system, which works in such a way that spatial or other relations among stimuli correspond to spatial relations among the neurons.

Various forms of the SOM have been used for applications in fields ranging from engineering (including image and signal processing and recognition, telecommunication, process monitoring, and robotics) to medicine, humanities, economics, and mathematics. Kaski *et al.* (1998) compiled a list of all the scientific articles (3,343 papers in total) in a classified bibliography on the theory and the use of the SOM between 1981 and 1997. Unsupervised SOMs are found useful, particularly for applications where no prior knowledge is available about the input. The self-organizing map offers a number of very important attractions as a neurospatial classifier (Openshaw, 1994);

- 1) simplicity in algorithmic design,
- 2) ability to handle complexity,
- 3) well-defined mathematical properties,
- 4) user induced flexibility,
- 5) a plausible degree of biological inspiration.

Kohonen's Self-Organizing Map (SOM) consists of two layers; the input and output layers (Figure 4.12). The input layer is called the sensory cortex, which has a number of neurons equal to the total number of input features. The output layer, which is a competitive layer, is termed the mapping cortex, and is in  $n \times m$  neurons in size (generally  $6 \times 6$  or  $8 \times 8$  neurons). The mapping cortex is usually a two-dimensional regular grid of nodes. The neurons in the input and output layers are connected to each other by synaptic weights  $W_{ij}$  where  $i$  and  $j$  refer to input neurons and output neurons respectively. In addition, the nodes in the mapping

cortex are locally interconnected as illustrated in Figure 4.12. The weights are iteratively modified during the learning stage so as to identify and reflect the characteristics of the data via the sensory cortex. Once a SOM is trained, the weights define the clusters in the topological feature space.

A simple stochastic learning process, based on the competitive learning concept, is employed to train a SOM. In the learning stage, the SOM units are adjusted by small steps with respect to the feature vectors that are extracted from the data and presented in a random order. One important characteristic of the SOM learning process is that the learning algorithm takes into account not only a specific output neuron but also the neighbourhood of that neuron. Thus, the weights associated with these neighbouring neurons are modified simultaneously. As a consequence, output neurons that are close to each other in the mapping cortex will maintain similar characteristics. This means that these neurons are also close to each other in the input space. At the end of the learning process, the weights connecting input and output layers are estimated in a way that they represent the characteristics of the input dataset.

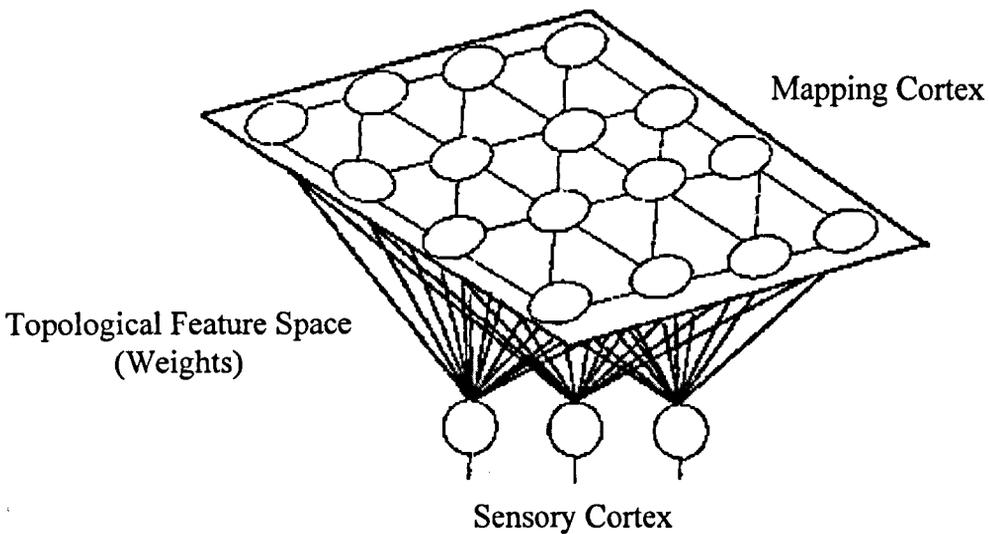


Figure 4.12 Kohonen's Self-Organizing Map.

The first step in unsupervised learning of the SOM is initialisation, which involves definitions of the geometry, dimensionality, and the size of the mapping cortex

(output layer). At this stage, all the weights  $w_{ij}$  in the network are set to small random values. Next, for each input pattern, the squared distances  $d_j^2$  between the  $i^{th}$  input neuron and the  $j^{th}$  neuron in the mapping cortex are calculated as follows:

$$d_j^2 = \sum (x_i(t) - w_{ij}(t))^2 \quad (4.6)$$

where  $x_i(t)$  is the input to sensory cortex neuron  $i$  at iteration  $t$ , and  $w_{ij}(t)$  is the weight associated with the link from input neuron  $i$  to output neuron  $j$  at iteration  $t$ . The selected output neuron is determined from:  $\min\{d_j^2\}$ ,  $j \in$  mapping cortex. The synaptic weights of neuron  $j$  and its neighbouring neurons are adjusted by a competitive Hebbian-type learning law:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t) \cdot \varepsilon_j(\sigma(t)) \cdot (x_i(t) - w_{ij}(t)) \quad (4.7)$$

where the learning rate  $\alpha(t)$  is a time-decaying function expressed as;

$$\alpha(t) = \alpha_{\max} \left( \frac{\alpha_{\min}}{\alpha_{\max}} \right)^{\frac{t}{t_{\max}}} \quad (4.8)$$

with the constraints  $1 \geq \alpha$ ;  $\alpha_{\min}, \alpha_{\max} \geq 0$ . The neighbourhood function  $\varepsilon_j$  determines a Gaussian neighbourhood range for all neurons  $j$  surrounding the winning neuron  $j$ . The error  $\varepsilon_j$  is calculated by,

$$\varepsilon_j = \exp\left(\frac{-(j-j')^2}{2\sigma(t)^2}\right) \quad (4.9)$$

where  $j'$  is neighbourhood of  $j$ , and  $\sigma(t)$  is a time-decaying function, defined by:

$$\sigma(t) = \sigma_{\max} \left( \frac{\sigma_{\min}}{\sigma_{\max}} \right)^{\frac{t}{t_{\max}}} \quad (4.10)$$

Usually, a value of  $\sigma_{\min}$  in the region of 1.0 is chosen, and  $\sigma_{\max}$  is set to a value in the region of 8.0.

The values of the functions  $\sigma(t)$  and  $\alpha(t)$  influence the learning mechanism. Several studies (Erwin *et al.*, 1992a, 1992b; Lo *et al.*, 1993) have investigated the effects of varying the values of the input parameters for both  $\sigma(t)$  and  $\alpha(t)$ . Their conclusion is that if a suitable value of  $\sigma_{\max}$  is chosen such that the neighbourhood function covers the whole mapping cortex, then the SOM will probably terminate in a well-ordered state. Also, researchers pointed out that the learning rate should be large (of the order of 0.9) at the beginning of training and should decrease during the training process to a value as small as 0.01 (Tso, 1997).

Kohonen's Self-Organizing Map (SOM) has a very important property of topology preservation. At the end of training, this property allows the investigator to obtain some insights into the data by looking into the activation of the neurons in the mapping cortex. If the classes in the data are easy to distinguish, the mapping cortex will be mapping some regions that correspond to clusters.

A SOM is also considered to be an effective tool for the visualization of high-dimensional data. A SOM converts complex and nonlinear statistical relationships between high-dimensional data points into more simple geometric relationships on a low-dimensional display. This can be thought of as data compression in that the most important features present are preserved while the dimensions of the data are reduced. It may also be considered to produce abstractions. As these two aspects, visualisation and abstraction, are the main purpose of many research projects, the SOM has found a large variety of applications in many fields.

Several modifications and extensions to the SOM have been proposed. One of these extensions is the use of a flexible map structure instead of a fixed grid to improve the preservation of topology. Another is to reduce the computational complexity of the SOM and, thus, speed up the learning process. Other

modifications are the use of a hierarchical clustering scheme and the use of an additional layer, which is called the Grossberg layer, to achieve supervised training.

#### **4.4 Summary and Conclusions**

The issue of visualising high-dimensional data, which are by nature complex, is discussed in this chapter. As visualisation helps the user to understand, or gain some insights into, the characteristics of such data, it is a particularly important topic for remote sensing studies where a large volume of data is available from many sources. The trend towards higher spatial and spectral resolution instruments in recent years is resulting in greater volumes of higher-dimensional data. Visualisation of multispectral, multitemporal, and multisensor data is thus of great importance. The primary aim of this study is to efficiently use graphical and projection methods in understanding the nature of the data at hand as well as to gain some insights into the behaviour of artificial neural networks.

The techniques developed to display high-dimensional data are discussed here under two main categories: graphical visualisation techniques and projection techniques. Major techniques for each of the two categories are reviewed in detail. Synthetic and satellite image data are used to demonstrate the advantages and disadvantages of each method. Since Sammon's nonlinear mapping algorithm was chosen to be the main technique to reduce the dimensions of the data into two and three dimensions, only the underlying mathematical theory of this method is given in detail.

In the toolkit, described in Appendix A, Sammon's nonlinear mapping algorithm as a projection method, parallel coordinate plots and Andrews' plots as graphical visualisation methods are available for use. In addition to these methods, animations can be produced for displaying and assessing the ANN learning process.

# CHAPTER V

## FEATURE SELECTION

### 5.1 Introduction

New sensors carried by recent remote sensing satellites provide higher spatial resolution and more spectral bands (or channels). The number of these spectral bands varies from a few, such as SPOT HRV, Landsat MSS and TM, to more than two hundred, such as MODIS (MODerate Resolution Imaging Spectroradiometer), and AVIRIS (Airborne Visible InfraRed Imaging Spectrometer). While each of these bands individually provides invaluable information to aid understanding of the nature of the remotely sensed objects, the data in many bands are highly correlated and therefore the dataset as a whole contains a degree of redundancy. It is necessary to eliminate such redundancy in order to produce more efficient methods of classification. In addition, some bands are sometimes irrelevant for the purpose of the investigation. A well known example is that thermal bands are not generally employed for delineation of land cover features in remote sensing. While using a large number of spectral bands increases feature space dimensionality, it gives rise to high performance costs and low classification accuracy.

The use of high-dimensional data can also have a severe impact on statistical classifiers. When the ratio of the number of training samples to the number of features is decreased, the parameters estimated for statistical classifiers become more variable and ambiguous. As a result, more samples are required to obtain

precise estimates of the parameters for high dimensional datasets. For a fixed training data sample size, the variances of the estimators of  $\mu$  and  $\Sigma$ , the mean vector and variance-covariance matrices for a particular class, will increase until a point is reached such that the instability of the estimators is greater than the increased information content of the additional features. This is often referred to as the *Hughes phenomenon* or the peaking phenomenon (Hughes, 1968). According to Landgrebe (2000) and Jimenez and Landgrebe (1998), there are two important characteristics of high dimensional feature spaces:

- As dimensionality increases the volume of a hypercube concentrates in the corners.
- As dimensionality increases the volume of a hypersphere concentrates in an outside shell, away from the centre of the spheres.

These two unique characteristics of high dimensional feature spaces indicate that higher dimensional space is mostly empty, which means that the multivariate data in any case generally occupy a subspace of lower dimensionality. Therefore, a high dimensional dataset can be projected to a lower dimensional space without significantly losing the level of separability.

The issue mentioned above suggests the need to select the most appropriate number of bands for a particular classification problem. The general tendency, reported in the literature, is to search for possible feature subsets of the full dataset in order to find one that is optimal in terms of a performance measure. The process of searching a subset of the whole dataset based on some kind of evaluation measure is called feature selection. Feature selection is a problem that has to be addressed in various fields. The main goal is to eliminate those bands that carry redundant or irrelevant spectral information.

There are three major advantages in using feature selection techniques. Firstly, the performance of a classification can be improved by reducing the number of bands to a new set of relevant and uncorrelated bands. In the case of artificial neural networks (ANNs), this issue is quite important for improving the generalisation capabilities of the network since, for a given number of training samples, a larger

network may have poorer generalisation capability than one with fewer inputs. Secondly, using a smaller number of bands reduces the time needed for processing. It is a well-known fact that the long training time requirement is one of the major drawbacks to the use of ANNs. Finally, due to the direct relationship between the dimensions of the data and the size of the sample set, lower-dimensional datasets would be more appropriate in cases where a limited number of training data are available. The issue of feature selection is therefore an important one, particularly where artificial neural networks are used.

As the evaluation of every possible subset drawn from the whole dataset is generally infeasible because of the computational effort required, a variety of search techniques has been developed and used for many research purposes. Major search techniques are discussed in the following sections after the section covering the feature selection techniques.

## **5.2 Test Sites, Data and Analysis Tools**

In this study, in order to make objective judgements about the performances of separability measures, an artificial neural network is applied to two classification problems involving two datasets from eastern England. Both test sites are fertile agricultural areas and rotational crop plantation techniques are in use.

While for the first test site the ground data were produced by digitising the field boundaries on a SPOT HRV image, the ground data of the second test site were produced by digitising field boundaries from several Ordnance Survey maps, which were published in 1987. The digitised polygons representing land cover classes were labelled based on the information collected.

### **5.2.1 Test Site 1**

Multisensor and multitemporal data including two Landsat TM and four SPOT HRV images were used to classify seven agricultural crops, namely, wheat, fallow, potato, sugar beet, onion, peas, and bulbs (daffodils). These crops cover the majority of the

study area. A total of 24 spectral bands (12 bands from Landsat TM images and 12 bands from SPOT HRV images) was available (Table 5.1). The study area (Figure 5.1) is located near the town of Littleport, in eastern England. The area selected for study is about 73.1 km<sup>2</sup> of fairly flat land (slope angles between 3<sup>0</sup> and 10<sup>0</sup>).

The images were registered to the Ordnance Survey of Great Britain's (OSGB) National Grid using the ERDAS Imagine image processing software (version 8.3) by applying a first-order polynomial transformation, which is in fact a linear transformation. The RMSE (Root Mean Square Error) values estimated for image transformations were less than one pixel. In the resampling process, all images were resampled at a spatial resolution of 30 metres, and 285-pixel by 285-pixel portions of the images covering the study area were extracted for subsequent analysis.

Table 5.1 Detailed information for the images used for the first test site.

Date	Time	Sensor	Site Path/Row	Centre Latitude	Centre Longitude	Band Order
27/06/94	10:10:50	Landsat-TM5	201/000	+52.41639	0.66084	1-6
20/07/94	10:16:11	Landsat-TM5	202/023	+53.10403	359.4584	7-12
13/05/94	11:15:17	SPOT-HRV1	East Anglia	+52.25277	0.60417	13-15
14/06/94	10:59:47	SPOT-HRV1	East Anglia	+52.25277	0.57222	16-18
30/07/94	11:15:11	SPOT-HRV2	East Anglia	+52.25277	0.65888	19-21
14/08/94	11:26:44	SPOT-HRV2	East Anglia	+52.25277	0.50889	22-24

The ground dataset (Figure 5.2) was generated from Field Data Printouts for the 1994 crop season, which provide details of the crop (or crops) growing in each field in the study area. These printouts were collected from individual farmers or their representative agencies. The boundaries of the land parcels in the study area were digitised, and each polygon was labelled with a number corresponding to the crop it contained.

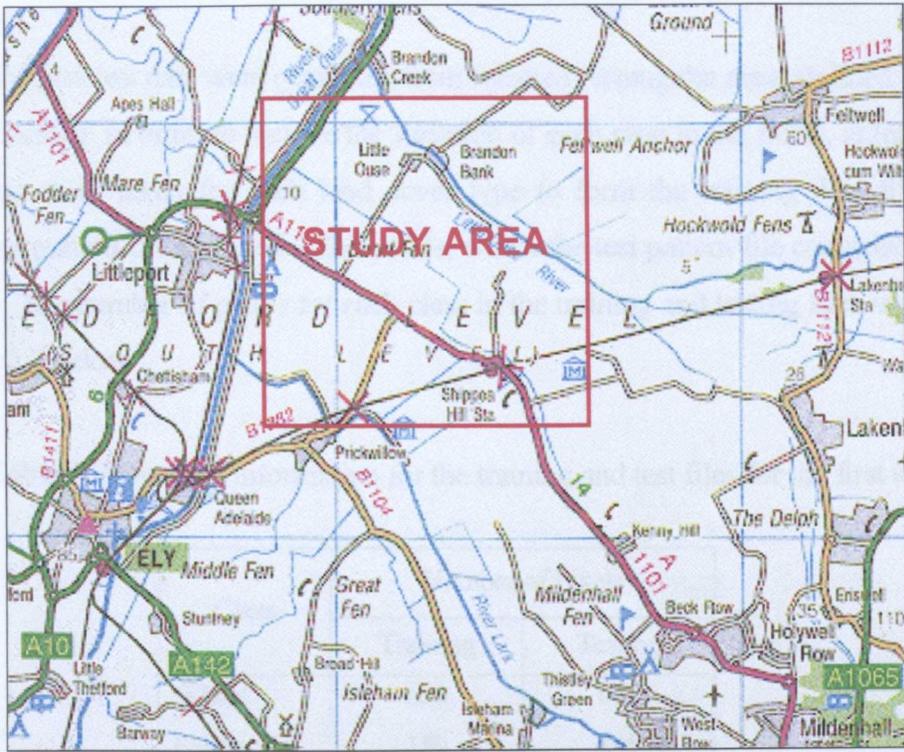


Figure 5.1 First area of interest near Littleport.

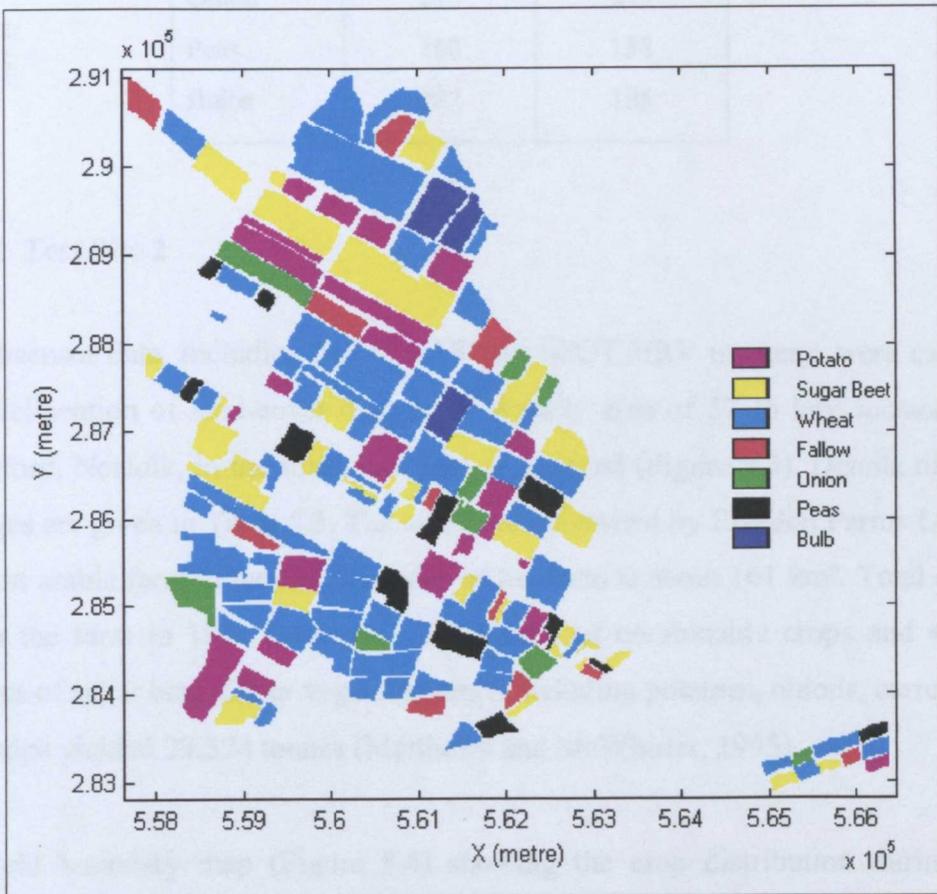


Figure 5.2 Ground reference data for the first test site.

Training and test data were generated from selected rectangular areas defined by rows and columns. In order to include the variation of each crop in the fields, at least three samples were taken for each land cover type to form the training data files. The training pattern file included 2,262 pixels, whilst the test pattern file comprised 2,204 pixels. The number of pixels for each class in the training and testing files is given in Table 5.2 below.

Table 5.2 Detailed information for the training and test files for the first site.

Class	Number of Pixels	
	Training	Testing
Wheat	620	642
Fallow	159	145
Potato	495	468
Sugar beet	431	387
Onion	215	219
Peas	160	158
Bulbs	182	185

### 5.2.2 Test Site 2

Multisensor data, including SIR-C SAR and SPOT HRV imagery, were used for the delineation of land-cover classes for a study area of 57.26 km<sup>2</sup> located near Thetford, Norfolk, in the south-east part of England (Figure 5.3). Details of these images are given in Table 5.3. The study area is owned by Elveden Farms Ltd, the largest arable farm in the UK. The size of the farm is about 101 km<sup>2</sup>. Total output from the farm in 1994 included 9,084 tonnes of combinable crops and 49,129 tonnes of sugar beet. Other vegetable crops including potatoes, onions, carrots and parsnips yielded 29,374 tonnes (Matthews and McWhirter, 1995).

A field boundary map (Figure 5.4) showing the crop distribution during late spring/early summer 1994 was created by digitising 65 field boundaries from a 1:25,000 scale Ordnance Survey map produced in 1987. The class labels are based

Table 5.3 Detailed information for the images used in the second test area.

Date	Time	Sensor	Site	Latitude	Longitude	Band Order
14/04/94	06:47:59	SIR-C	Thetford	+52.37000	0.76667	1-4
14/04/94	06:47:59	Filtered SIR-C	Thetford	+52.37000	0.76667	5-8
13/05/94	11:15:17	SPOT-HRV1	East Anglia	+52.25277	0.60417	9-11
14/06/94	10:59:47	SPOT-HRV1	East Anglia	+52.25277	0.57222	12-14
28/06/94	11:30:37	SPOT-HRV1	East Anglia	+52.25444	0.32250	15-17
30/07/94	11:15:11	SPOT-HRV2	East Anglia	+52.25277	0.65889	18-20
14/08/94	11:26:44	SPOT-HRV2	East Anglia	+52.25277	0.50889	21-23

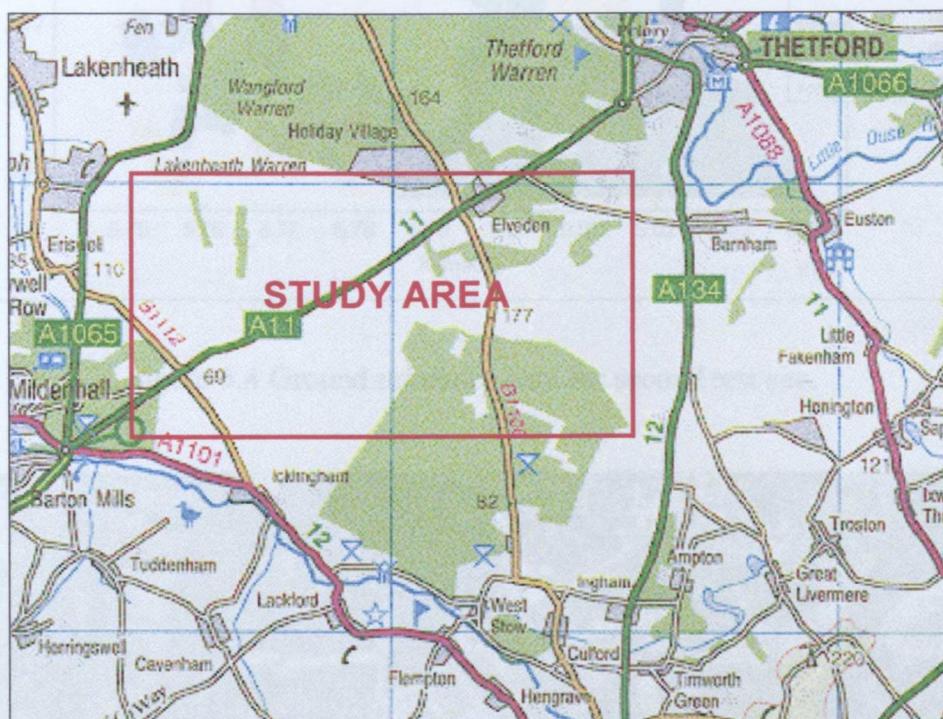


Figure 5.3 Location of the second area of interest near Thetford.

on information from a previous study performed in the Geography Department of Nottingham University. Quad-polarised L-band (~24cm wavelength) SIR-C SAR data in four polarisation modes (HH: transmit horizontal and receive horizontal; HV: transmit horizontal and receive vertical; VH: transmit vertical and receive horizontal; and VV: transmit vertical and receive vertical) were acquired by the NASA/JPL SIR-C (Shuttle Imaging Radar) system on April 14, 1994. Five SPOT

HRV images, acquired between May and August 1994, were also available. Due to the short time difference between the acquisition dates, it is assumed that there was no dramatic change on the types of ground cover classes. The extracted section of the SIR-C SAR image for the study is given in Figure 5.5. As can be seen from the figure, there are no data available for the lower right part of the image due to the limited extent of the SIR-C SAR coverage.

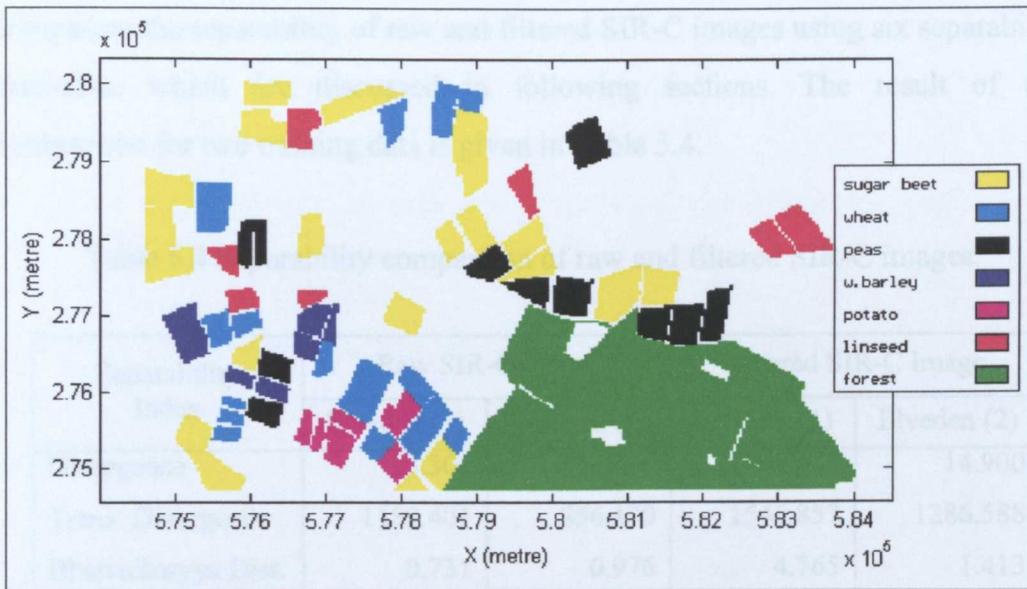


Figure 5.4 Ground reference data for second test site.



Figure 5.5 SIR-C SAR image containing the study area. The clipping of the image in the lower right corner is due to the limited extent of the SIR-C SAR coverage.

As radar systems generate images by the coherent processing of scattering signals, they are highly susceptible to speckling effects. The presence of multiplicative speckle noise in an image reduces the ability of the user to distinguish and classify. Thus, pre-processing of the image is necessary (Chen *et al.*, 1996). A 5 by 5 Lee filter was used to reduce the effects of speckle noise, as it was found highly effective by Tso and Mather (1999). Filtering also leads to improvements in the separability of the classes. This philosophy has been tested in this study by comparing the separability of raw and filtered SIR-C images using six separability measures, which are discussed in following sections. The result of this comparison for two training data is given in Table 5.4.

Table 5.4 Separability comparison of raw and filtered SIR-C images.

Separability Index	Raw SIR-C Image		Filtered SIR-C Image	
	Elveden (1)	Elveden (2)	Elveden (1)	Elveden (2)
Divergence	23.301	10.268	80.533	14.900
Trans. Divergence	1189.401	856.170	1545.857	1286.588
Bhattacharyya Dist.	0.731	0.976	4.765	1.413
J-M Distance	886.154	767.080	1168.203	1038.353
Wilks' $\Lambda$	0.258	0.288	0.054	0.122
Hottelings $T^2$	345.896	897.618	5178.119	1437.770

Elveden (1): training data with pure pixels, Elveden (2): training data with mixed pixels.

As can be seen from the above table, the use of the Lee filter considerably improved the separability of the classes for all indices used. Please note that lower values of Wilks'  $\Lambda$  criterion indicate better separation.

Both the raw and filtered SIR-C images were employed in the feature selection process in order to observe the effectiveness of filtering for land cover class discrimination. As either of HV or VH polarisation images is chosen for further analysis in the literature, a cross-correlation table (Table 5.5), that can be used as an index for the level of dependence between the images, was prepared for the polarisation images available. Although the correlation between HV and VH polarisation images is the highest, neither of these bands was excluded since their

inter-correlation was not considerably higher than the others. Therefore, there is a total of 23 spectral bands available (15 bands from SPOT images and 8 bands from SIR-C images). SPOT and SIR-C images were then georeferenced to the Ordnance Survey of Great Britain's National Grid using the ERDAS Imagine image processing software (version 8.3) by applying a first-order polynomial transformation. The RMSE values of the reference points chosen for image transformations were less than half a pixel. A sub-image of size 228×436 pixels covering the study area were extracted at a spatial resolution of 24 metre to be used in subsequent analysis.

Table 5.5 Cross-correlation between SIR-C polarisation images.

	HH	HV	VH	VV
HH	1.0000			
HV	0.7842	1.0000		
VH	0.8285	0.8638	1.0000	
VV	0.8597	0.7427	0.8118	1.0000

On the basis of a number of experiments, it was decided to use seven land-cover classes, which included the bulk of the study area. ANN classification procedures were applied to examine these seven classes (sugar beet, wheat, peas, forest, winter barley, potato and linseed). Training and test datasets were produced using a random selection method. For each of the seven land cover classes, 300 pixels were randomly selected for training the network (a total of 2,100 pixels for training), and 250 pixels were randomly selected for testing the trained networks (a total of 1,750 pixels for testing). Although the training and test areas were selected to be as homogeneous as possible, there was some heterogeneity within the areas. Specifically, digitising exact field boundaries gives rise to the inclusion of boundary pixels that are basically mixtures of adjacent pixels. As a consequence, a 100% overall accuracy would not be expected from any classification method.

ERDAS Imagine image processing software (version 8.3) was used for preparing the satellite data and selecting the training and testing data for the neural network

classification. SNNS (Stuttgart Neural Network Simulator), developed by the Institute for Parallel and Distributed High Performance Systems at the University of Stuttgart, was used for neural network implementations. SNNS is an efficient and portable neural network simulation environment for UNIX workstations, and is used to generate, train, test and visualise artificial neural networks. PC version of the SNNS program has been introduced recently, which can be run via the visualisation toolkit written for this study. The toolkit, which is described in Appendix A, and some programs written in Turbo C and MATLAB (version 5.3) were used to prepare data for SNNS and analyse the results of SNNS.

### 5.3 Filters and Wrappers

A number of criteria can be used to categorise feature selection techniques. As they can be classified on the basis of whether they are graphical or statistical in nature (Jensen, 1996), so they can also be classified into two categories based on whether or not they use classification algorithms to evaluate subsets (Figure 5.6). Techniques that use classifiers to evaluate the performance of subsets are called ‘wrapper techniques’. Otherwise, they are called ‘filter techniques’, in which no classifier is employed to evaluate subset solutions.

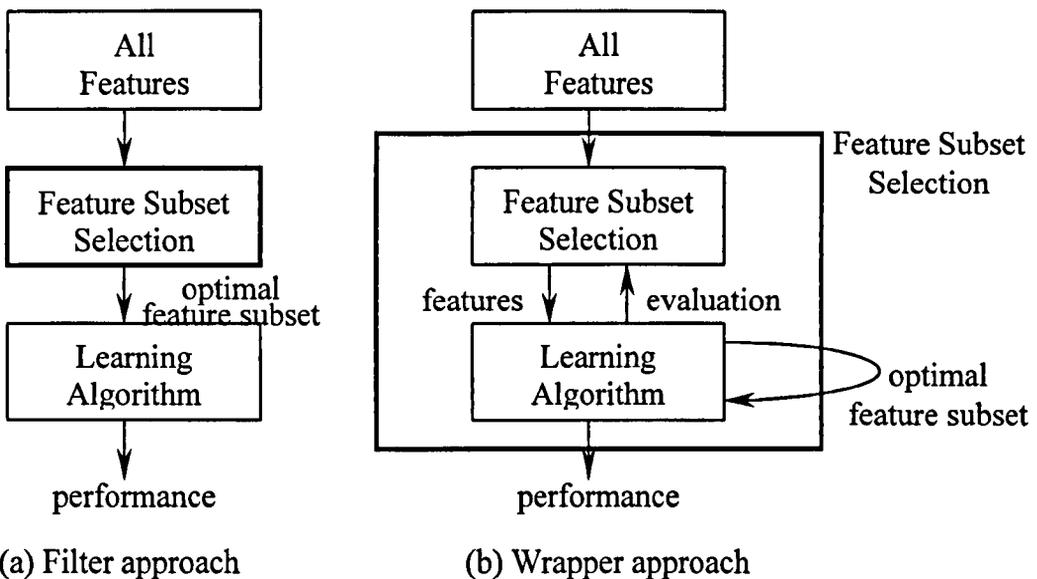


Figure 5.6 Two approaches to feature subset selection based on the incorporation of a learning algorithm (Yang and Honavar, 1998).

The filter approach is generally computationally more efficient; however, this approach may not find the optimal subset solution for the classifier. The wrapper approach is, on the other hand, based on the evaluation of the number of feature subsets by executing a selected classification algorithm and selecting the best one of the candidate subsets. The main problem with such methods is their computational requirements. They are generally used when the classification process employed is relatively fast.

A filter is defined as a feature selection algorithm using a performance metric based entirely on the training data without reference to the classifier for which the features are to be selected. The name is derived from the way in which the features are filtered before the classification system is trained and tested (Scott *et al.*, 1998).

The most widely-used filter methods are based on class separability indices. Such indices have been extensively used by researchers in the remote sensing area for many investigation purposes (Goodenough *et al.*, 1978; Thomas *et al.*, 1987a, 1987b; Mather, 1999a; Jensen, 1996; Aha and Bankert, 1996; Dutra and Huber, 1999; Tso and Mather, 1999).

In the following discussion, the fundamentals of each separability measure are given, and the classification results derived from the use of artificial neural networks, whose input nodes correspond to best band combinations, are presented. The best band combinations are reached as a result of a search process in which separability measures are used as evaluation (or fitness) functions to evaluate the performance of each subset solution. In the search for the best band solutions, both sequential forward selection (SFS) and genetic algorithm (GA) methods, which are discussed in subsequent sections, are employed. Best band combinations found by separability measures and the Mahalanobis distance classifier are used in a feed-forward artificial neural network to delineate land cover classes.

## 5.4 Class Separability Indices

The measurements associated with each class exhibit a statistical probability distribution. Such probability distributions often overlap, to a greater or lesser extent, and the class separability problem becomes a function of both the separation of the means and statistical distribution of data points, within each class, for each dimension (or spectral band). The evaluation of class separability in multidimensional space from a combination of class separabilities in one-dimensional spaces leads to errors because the correlations between the dimensions must be considered in addition to the single-dimensional measures of the distributions of the class data points, the variances, and the separations between the class means (Thomas *et al.*, 1987a).

The logic behind the separability indices is that the larger the separation between the classes in the feature space the easier it will be to discriminate between the features as a result of better decision boundary determination, thus a lower error rate (better performance) can be achieved by the classifier following feature selection.

A number of procedures that measure inter-class separability are described in the literature. The best known are the Euclidean distance, the Mahalanobis distance, the Divergence, the Transformed Divergence, the Bhattacharyya distance, and the Jeffries-Matusita distance.

These indices with the exception of the Euclidean distance and the Mahalanobis distance, which are theoretically and mathematically well-known distance measures, are discussed in the following section.

### 5.4.1 Divergence and Transformed Divergence Indices

Divergence is based on the derivation of a measure of the difference between all pairs of classes. It was one of the first separability indices used in remote sensing and is still in use for processing of remotely sensed data (Goodenough *et al.*,

1978; Swain and Davis, 1978; Mather, 1999a; Thomas *et al.*, 1987a, 1987b; Jensen, 1996).

Divergence is computed using the mean and variance-covariance matrices of the data representing feature classes. For two feature classes ( $i$  and  $j$ ), the divergence between the classes is calculated according to the formula:

$$D_{ij} = \frac{1}{2} \text{tr}[(V_i - V_j)(V_j^{-1} - V_i^{-1})] + \frac{1}{2} \text{tr}[(V_i^{-1} + V_j^{-1})(M_i - M_j)(M_i - M_j)^T] \quad (5.1)$$

where the symbol  $\text{tr}[\cdot]$  indicates the trace of a matrix, which is the sum of its diagonal elements,  $V_i$  and  $V_j$  are the variance-covariance matrices for class  $i$  and  $j$ , and  $M_i$  and  $M_j$  are the corresponding sample mean vectors. In cases in which more than two classes are involved, average divergence is computed. This involves the estimation of divergence values for each pairwise combination of the classes. The best subset band combination can be found by searching for the highest  $D_{avg}$  value from all the possible subsets. As the effect of several well-separated classes may increase the average divergence value and make it misleading, the transformed divergence is introduced, which can be expressed as:

$$TD_{ij} = c \left[ 1 - e^{\frac{-D_{ij}}{8}} \right] \quad (5.2)$$

where  $c$  is a constant that defines the range of transferred divergence values. In the literature,  $c$  has been chosen as 100, 1000 and 2000. The transformed divergence applies an exponentially decreasing weight as distance between the classes increases, and also scales the divergence values between 0 and  $c$ . According to Jensen (1996), if  $c$  is chosen as 2000, then values of  $TD$  above 1900 indicate good separability while values below 1700 indicate poor separability. On the other hand, it has been suggested by Mather (1999a) that when  $c$  is 100, transformed divergence values can be interpreted in the same way as percentages. In this case, values of 80 or higher indicate good separability between the classes.

The main problem in the use of divergence as a measure of inter-class separability is the assumption of multivariate normal distribution for the data representing the classes. In other words, divergence estimation is based on the assumption that the data used are normally distributed. Divergence values will be less reliable when the data depart significantly from multivariate normality.

In order to determine the optimum number of features that can produce accurate classification results, the sequential forward selection (SFS) search method is used in conjunction with the divergence measure to determine the best feature combinations ranging from 5 to 24 features for the first dataset and 5 to 23 features for the second. These solutions are used to construct training and test files. All the network and learning parameters were kept constant except for the number of input nodes in the network. Networks were trained for 15,000 iterations using the backpropagation learning algorithm. The classification accuracies produced for the test datasets were plotted against the number of input features. This analysis showed that a minimum of eight features is needed for the neural network to learn the characteristics of the training data with around 90% overall classification accuracy.

The primary aim of the study is thus to determine the best eight bands for both problems to distinguish seven land cover features. All the feature selection techniques were used to accomplish this task. On the other hand, in order to observe the effects of different network architectures and number of iterations, results are given for three network structures (8-10-7, 8-15-7, 8-20-7, where 8 shows the number of input bands, 10, 15 and 20 indicate the number of nodes in the hidden layer, and 7 is for the number of output classes to be identified) and for every 2,500 iterations, for a total of 15,000 iterations. All the classification results were assessed using contingency matrices to determine the overall, and individual class accuracies. While assessing the results produced by the networks, pixels are left unclassified if none of their membership values exceeds 0.5.

Divergence, like the other separability measures, is used in this study as a fitness measure to determine the performance of each subset band combination in a search process. For the first problem involving the selection of the best eight band combination from 24 bands, the best subset solutions of 6-21-19-13-11-18-24-12 and

5-12-13-17-18-19-21-24 were found for the sequential forward selection (SFS) and genetic algorithm (GA) search methods to be described in section 5.7, respectively. Training and test data were created using these solutions to distinguish seven land cover classes. Three network structures were trained with the training dataset and later were assessed using the test dataset. The results of these analyses are given in Table 5.6 and Table 5.7. The divergence, transformed divergence, the Bhattacharyya and Jeffries-Matusita distances for the solutions are estimated and given below the tables using the abbreviations D, TD, B, and JM. It should be noted that the values of the Kappa coefficient in Tables are represented by the values multiplied by 100.

Table 5.6 ANN classification results of the solution attained by divergence using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	91.15	89.27	91.33	89.44	92.70	91.08
5000	90.70	88.73	90.79	88.81	92.11	90.38
7500	90.11	88.05	91.02	89.11	90.47	88.44
10000	89.88	87.82	90.43	88.41	90.02	87.91
12500	90.52	88.57	90.56	88.59	90.06	88.00
15000	90.29	88.33	90.34	88.31	90.20	88.16

D: 391.265      TD: 1999.926      B: 13.765      JM: 1413.299

As can be seen from Table 5.6, the solution produced by divergence analysis gives over 90% classification accuracy based on overall accuracy, and 88% accuracy based on the use of the Kappa coefficient. These high accuracy values suggest that using eight bands instead of twenty-four bands (16 out of 24 eliminated) is appropriate to delineate seven agricultural crops with a reasonably high classification accuracy. This reduction also shortened the training and testing processes and, more importantly, it produced a much smaller network with higher generalisation capabilities.

Table 5.7 ANN classification results of the best band combination found by divergence using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	87.48	84.90	90.29	88.14	89.52	87.27
5000	87.30	84.65	90.34	88.19	89.25	86.94
7500	86.71	83.93	89.88	87.63	89.38	87.12
10000	86.75	83.92	89.34	86.99	89.75	87.55
12500	86.03	83.03	89.11	86.72	89.16	86.85
15000	85.21	82.02	88.88	86.46	89.16	86.86

D: 478.471      TD: 1998.757      B: 14.162      JM: 1411.675

The results in Table 5.7 show that the genetic algorithm (GA) could not find a better solution than the sequential forward selection (SFS) search method in terms of the classification accuracy produced by neural networks. However, it should be pointed out that GA, in fact, found a solution with higher divergence, but this did not improve the accuracy of the classification. The main reason for this may be that GA approach searches the best bands by considering only the average divergence rate (mean of the divergences calculated for each possible pair of bands), whereas the SFS program written for this study seeks the first four bands to improve the average divergence, and then the next bands to improve the poorest (lowest) divergence among the band pairs. Hence, it may be the reason that these poor divergence correlations between the classes reduced the accuracy of resulting classification. Another point should be made is that larger networks do not necessarily provide higher accuracies, as can be noticed from Table 5.6 and Table 5.7.

The smallest network (8-10-7) trained (2,500 iterations) for the solution found by SFS based on divergence was used to classify the test image and the result of the classification was portrayed using four colour tones depending on output activation values where each class was represented by a distinct colour. The result

of this operation is presented as Figure 5.7. Such representation clearly provides a better understanding of the classification results in that some kind of accuracy assessment can be made visually.

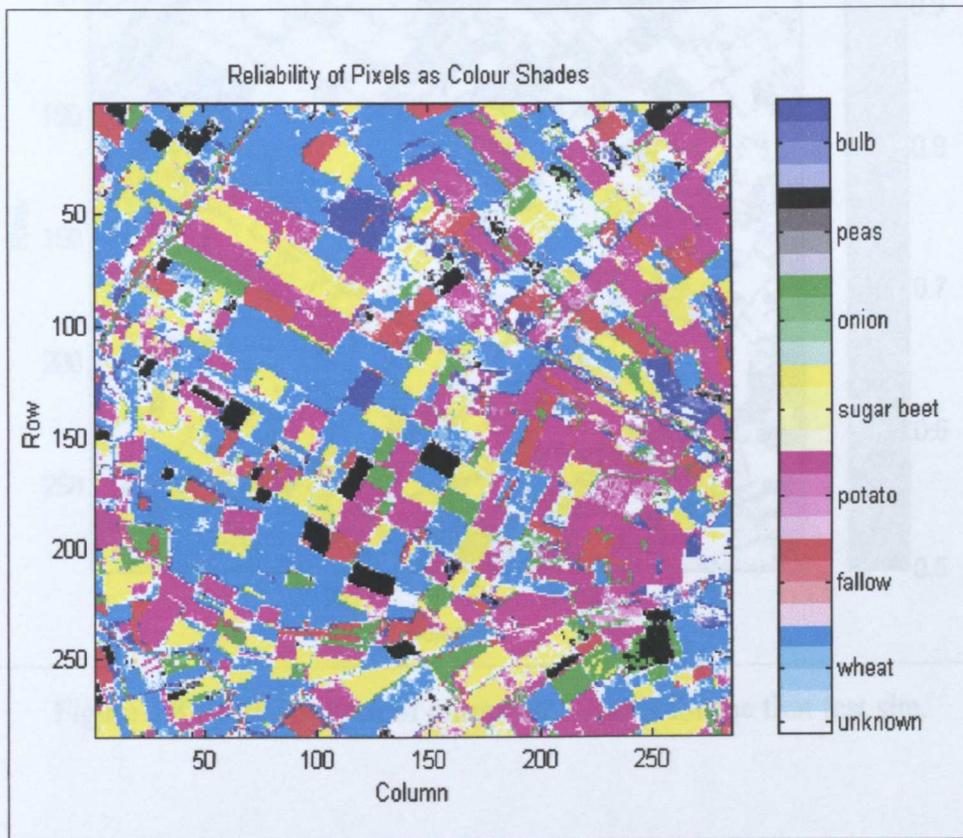


Figure 5.7 ANN classification of the test image for the solution found by SFS based on divergence for the first test site. The output activations for each pixel are shown in one of four levels of colour.

It is also possible to present the results of an ANN based classification using only the output activation values, independent of land cover classes. Thus, one can analyse each pixel, and observe the effect of spectral variation in individual fields. Moreover, fields including different crops than the ones used in the training process can be seen as totally dark. The membership levels of all pixels are represented on a grey scale on which output activations lower than 0.5 are set to black, whilst output activation of 1 is set to white. The result of this process is shown in Figure 5.8.

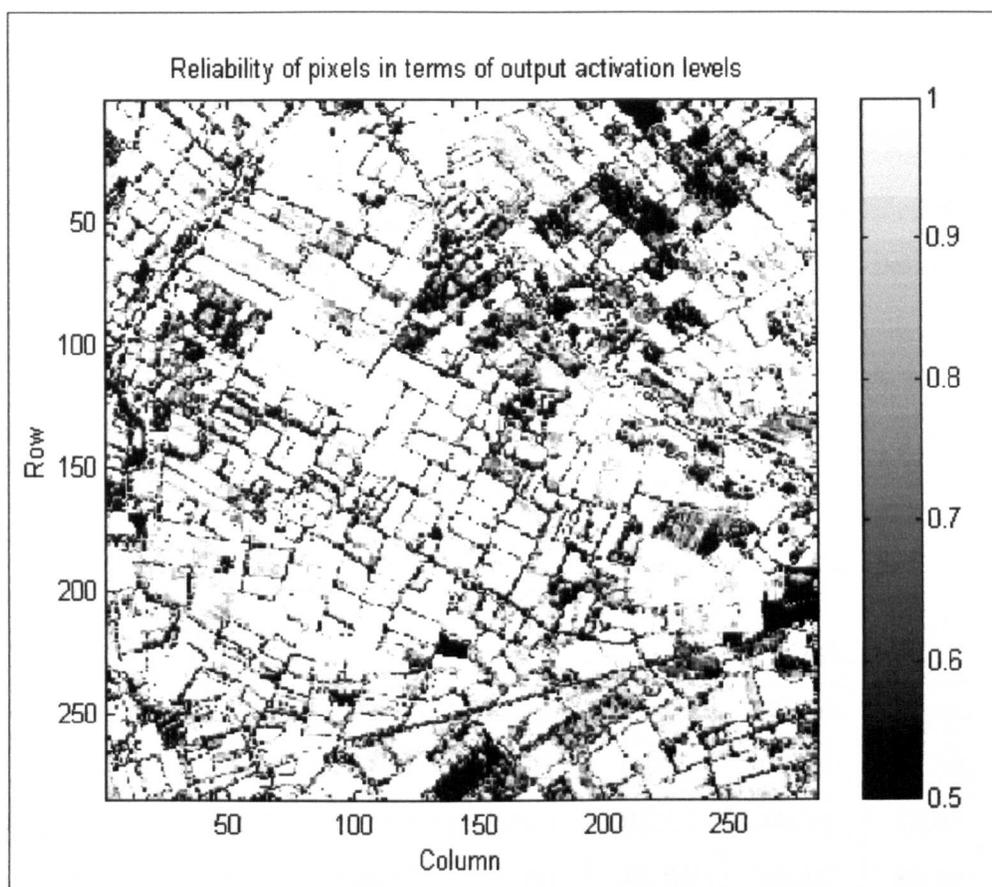


Figure 5.8 Spatial pattern of output activations for the first test site.

Table 5.8 ANN classification results of the best band combination found by divergence using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.66	88.13	89.66	88.16	90.51	89.12
5000	90.17	88.69	89.60	88.09	90.86	89.50
7500	90.11	88.62	89.71	88.23	90.46	89.06
10000	89.54	87.99	90.51	89.10	90.91	89.56
12500	89.83	88.31	90.40	88.96	90.91	89.56
15000	89.60	88.04	90.63	89.21	90.17	88.71

D: 130.641

TD: 1999.475

B: 4.834

JM: 1397.656

When the SFS and GA search methods were applied to determine the optimum band subsets for second dataset, the solutions of 20-11-16-17-23-19-13-14 and 9-11-14-16-17-19-20-23 band combinations were obtained. These solutions have been used to form training and test pattern files for ANN classification. After training, the three network structures were tested for every 2,500 iteration. The results of these processes are given in Table 5.8 and Table 5.9.

Table 5.9 ANN classification results of the best band combination found by divergence using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.20	87.63	89.54	88.03	90.06	88.58
5000	89.43	87.87	90.00	88.54	89.94	88.45
7500	90.29	88.81	90.00	88.56	90.46	89.05
10000	90.23	88.75	90.11	88.69	90.11	88.65
12500	90.11	88.63	89.66	88.18	89.77	88.26
15000	90.51	89.09	89.77	88.30	90.17	88.69

D: 140.678

TD: 1999.125

B: 5.206

JM: 1398.213

Although GA found a solution with considerably high divergence rate, the classification results of both search methods are comparable. It can be seen that none of the network structures is superior to others despite the slight changes in the accuracy. The overall accuracy of 90% is achieved by both solutions.

Transformed divergence was also used to find out the best band combination for discriminating the land cover classes for both test sites. Sequential forward selection (SFS) method and genetic algorithm (GA) were employed for this purpose using transformed divergence as the fitness measure. For the first test dataset, the band combinations 11-21-18-19-5-24-12-2 and 2-5-11-12-13-18-20-24 were found using SFS and GA, respectively. The results of these solutions are presented here in Table 5.10 and Table 5.11.

Table 5.10 ANN classification results of the best band found by transformed divergence using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	87.52	85.01	89.34	87.16	86.39	83.77
5000	87.89	85.41	88.20	85.88	85.84	83.15
7500	87.30	84.75	88.25	85.96	85.48	82.71
10000	87.02	84.42	88.29	86.02	84.75	81.89
12500	86.84	84.19	88.43	86.18	84.53	81.61
15000	87.21	84.57	88.38	86.11	84.94	82.09

D: 352.246      TD: 1999.945      B: 12.772      JM: 1413.250

Table 5.11 ANN classification results of the best band found by transformed divergence using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	91.24	89.41	91.42	89.61	93.15	91.66
5000	90.29	88.30	87.75	85.30	92.70	91.10
7500	90.38	88.36	89.34	87.15	91.79	90.03
10000	89.56	87.40	89.70	87.54	92.20	90.51
12500	88.43	86.07	90.02	87.93	92.02	90.28
15000	87.89	85.40	90.29	88.24	91.97	90.22

D: 301.096      TD: 1999.961      B: 13.773      JM: 1413.467

The solution found by GA based on transformed divergence produced results with over 92% overall accuracy and 90% Kappa coefficient whereas the solution found by SFS gave results with around 88% overall accuracy and 86% Kappa coefficient. One conclusion that can be drawn is that GA reached a better solution in terms of both the transformed divergence and classification accuracy. Another point should be made is that the best structure appeared to be 8-15-7 for SFS

solution, and 8-20-7 for GA solution. The highest accuracy of 93.15% overall and 91.66% Kappa coefficient was reached by network structure of 8-20-7 at 2,500 iterations.

For the second test site, SFS and GA procedures were also applied to find the optimum subset band combination in terms of transformed divergence value. Whilst the SFS technique reached the same solution that the divergence reached (Table 5.8), GA found a solution containing 10-14-16-17-19-20-21-23 bands. The solution found by GA was employed in ANN processes using the three network structures and the results are shown in Table 5.12. It can be seen from the comparison of two tables (Table 5.8 and Table 5.12) that although GA reached a better solution in terms of the fitness measure, transformed divergence value, classification results produced are slightly worse than those produced by SFS procedures.

Table 5.12 ANN classification results of the best band found by transformed divergence using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	88.97	87.38	89.89	88.37	89.09	87.52
5000	89.60	88.09	89.09	87.50	90.06	88.60
7500	89.31	87.76	90.06	88.57	89.43	87.91
10000	88.91	87.31	89.66	88.12	89.49	87.95
12500	88.91	87.31	89.83	88.31	89.26	87.72
15000	89.26	87.68	89.71	88.20	89.66	88.15

D: 126.819      TD: 1999.519      B: 4.801      JM: 1398.494

The smallest network structure (8-10-7) trained for the solution (20-11-16-17-23-19-13-14) found by SFS based on divergence for the second dataset was also used to classify the test image, and the result is given in Figure 5.9. Grey scale activation level analysis was also carried out to show problematic areas, and the effect of spectral variations. It is also possible to define boundaries of fields since

mixed pixels are classified with low possibility of membership, and therefore they are darker than the pixels inside fields. The image produced by this process is shown in Figure 5.10. The dark and considerably large area in the middle of the image was not recognised by the ANN as it belongs to another class (possibly grass) that was not included in the training set.

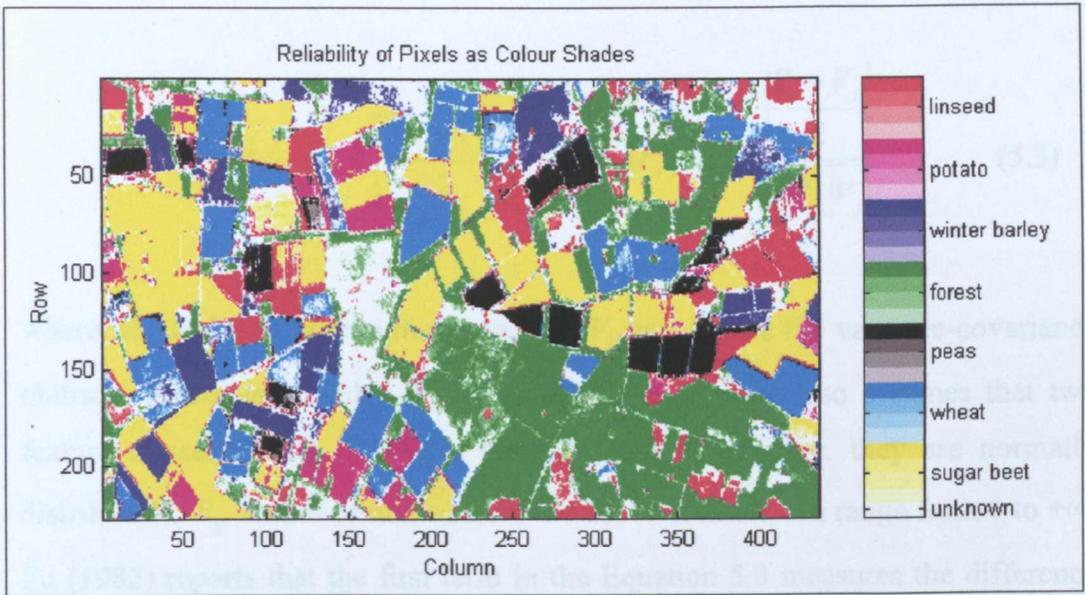


Figure 5.9 ANN classification of the test image for the solution found by SFS based on divergence for the second site using bands 20-11-16-17-23-19-13-14.

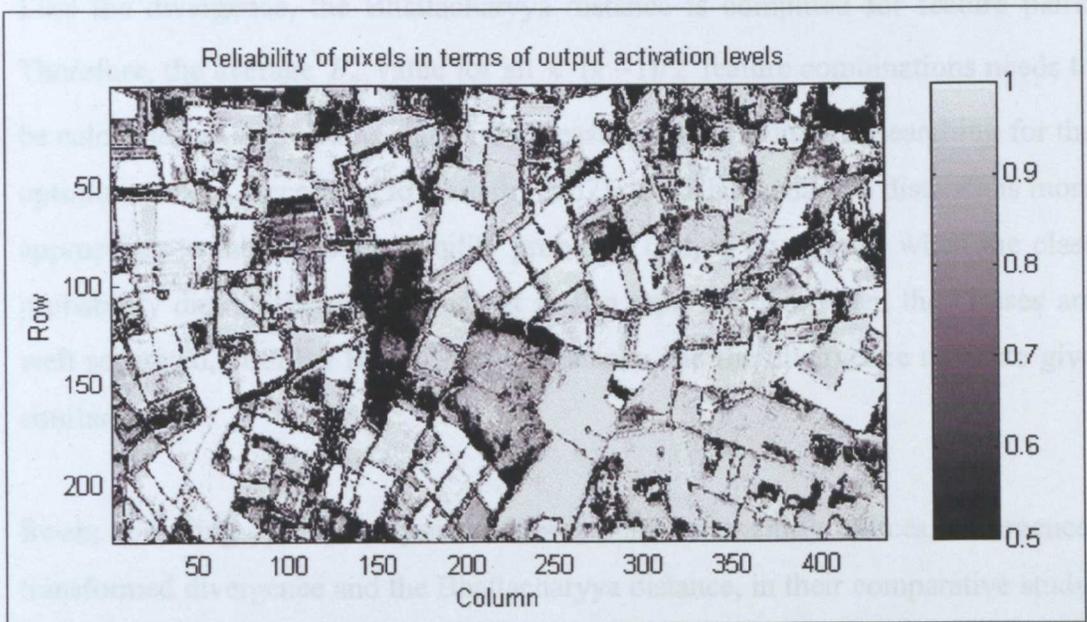


Figure 5.10 Spatial pattern of output activations for the second site.

### 5.4.2 Bhattacharyya Distance

The Bhattacharyya distance is a widely-used class separability index. While divergence measures statistical separability, the Bhattacharyya distance estimates the probability of correct classification. The Bhattacharyya distance is calculated from the formula below:

$$B_{ij} = \frac{1}{8} (M_i - M_j)^T \left[ \frac{(V_i + V_j)}{2} \right]^{-1} (M_i - M_j) + \frac{1}{2} \ln \frac{\left| \frac{V_i + V_j}{2} \right|}{\sqrt{|V_i| |V_j|}} \quad (5.3)$$

where  $M_i$  and  $M_j$  are the mean vectors,  $V_i$  and  $V_j$  are the variance-covariance matrices of classes  $i$  and  $j$ . The Bhattacharyya distance also assumes that two feature classes ( $i$  and  $j$ ) have a Gaussian distribution (i.e. they are normally distributed).  $B_{ij}$  values estimated from the above formula can range from 0 to  $+\infty$ . Fu (1982) reports that the first term in the Equation 5.3 measures the difference between the class means, and the second term measures the difference between the within-class variance-covariance matrices.

Like the divergence, the Bhattacharyya distance is computed for feature pairs. Therefore, the average  $B_{ij}$  value for all  $k \cdot (k-1)/2$  feature combinations needs to be calculated as an overall separability measure, which is used in searching for the optimum subset. According to Kailath (1967), 'the Bhattacharyya distance is more appropriate to interclass separability problems than is divergence when the class probability distributions are broad'. It is also reported that, when the classes are well separated, both the Bhattacharyya distance and the divergence measure give similar results.

Swain and King (1973) analysed the three class separability indices, divergence, transformed divergence and the Bhattacharyya distance, in their comparative study. They found that transformed divergence and the Bhattacharyya distance performed best. However, it should be pointed out that normally distributed artificial data was

generated and used in their study. Therefore, the conclusion reached may be misleading, especially in cases where the data are not normally distributed.

When SFS and GA methods were applied to determine best band combinations based on Bhattacharyya distance, 24-11-18-23-16-5-17-20 and 5-11-13-15-17-18-22-23 solutions were found for the first dataset, and 11-20-19-9-18-15-12-17 and 9-11-16-17-18-19-20-23 for the second dataset, respectively. Training and test pattern sets were produced using these solutions. Then, the networks trained with the pattern dataset and subsequently assessed using the test pattern sets. Results of the solutions obtained for the first dataset are given in Table 5.13 and Table 5.14. The comparison of two tables suggests that there is a slight improvement in the classification accuracy for the solution attained by GA procedure. As it can be noticed from the comparison of the tables, GA solution provided results with consistency. However, the results in Table 5.13 show larger deviations in terms of classification accuracy. Another point should be made is that the 8-15-7 network structure for both cases produced the best results at 2,500 iterations (90.56% for SFS solution and 90.70% for GA solution).

Table 5.13 ANN classification results for the band combination attained by the Bhattacharyya distance using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	86.43	83.75	90.56	88.61	88.20	84.82
5000	87.93	85.50	88.16	85.73	88.38	85.63
7500	87.30	84.71	88.07	85.63	88.43	84.97
10000	86.48	83.70	88.11	85.67	88.48	84.60
12500	86.03	83.19	87.84	85.36	88.52	84.76
15000	85.53	82.60	87.66	85.13	88.16	84.81

D: 399.974      TD: 1999.796      B: 15.346      JM: 1413.010

Table 5.14 ANN classification results for the band combination attained by the Bhattacharyya distance using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	87.70	85.28	90.70	88.73	87.93	85.49
5000	88.29	86.04	89.16	86.90	88.61	86.23
7500	88.70	86.49	88.88	86.54	88.16	85.64
10000	88.52	86.30	88.52	86.11	87.11	84.42
12500	88.61	86.37	88.79	86.44	87.84	85.22
15000	88.88	86.65	88.79	86.43	87.66	85.00

D: 396.771      TD: 1999.686      B: 17.822      JM: 1412.561

Table 5.15 ANN classification results for the band combination attained by the Bhattacharyya distance using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.26	87.70	89.54	88.02	89.03	87.47
5000	89.77	88.26	89.66	88.13	89.60	88.11
7500	89.89	88.41	89.83	88.34	89.43	87.91
10000	89.66	88.15	89.43	87.89	89.94	88.47
12500	89.49	87.94	89.71	88.20	89.77	88.28
15000	89.31	87.78	89.83	88.32	89.83	88.36

D: 128.512      TD: 1997.236      B: 5.123      JM: 1396.055

SFS and GA procedures were also applied to determine the optimum subset band combination for the second dataset. 11-20-19-9-18-15-12-17 and 9-11-16-17-18-19-20-23 band combinations were found to be the best subset solutions by SFS and GA, respectively. The results of applying these solutions to ANN classification are given in Table 5.15 and Table 5.16. While an overall accuracy of less than 90% and a Kappa coefficient of 88% were achieved by the solution

obtained from SFS, around 91% overall accuracy and 89% Kappa coefficient accuracy values were produced from the solution attained by GA method. In this case, GA method reached a solution with both high Bhattacharyya value and high classification accuracy. It should be also pointed out that the solution found by GA also provides better estimates for other separability indices.

Table 5.16 ANN classification results for the band combination attained by the Bhattacharyya distance using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.37	87.83	91.03	89.67	91.03	89.70
5000	90.40	88.97	90.74	89.38	91.14	89.82
7500	90.51	89.10	90.11	88.69	90.86	89.50
10000	90.06	88.59	90.23	88.80	90.63	89.25
12500	90.51	89.10	90.17	88.74	90.63	89.25
15000	90.17	88.71	89.94	88.47	90.80	89.44

D: 144.858

TD: 1998.845

B: 5.322

JM: 1399.370

### 5.4.3 Jeffries-Matusita Distance

The Jeffries-Matusita distance separability index, often referred to as the J-M distance, is very similar to the transformed divergence in terms of its formulation. It is a saturating transformation applied to the Bhattacharyya distance. The J-M distance between two normally distributed classes ( $i$  and  $j$ ) is given by:

$$J_{ij} = \sqrt{2(1 - e^{-B_{ij}})} \quad (5.4)$$

where  $B_{ij}$  is the Bhattacharyya distance.  $J_{ij}$  has a saturating behaviour with increasing class separability and is more suitable as a measure of interclass separability than is divergence. However, it tends to suppress high separability values, whilst overemphasising low separability values.

The J-M distance has been used in remote sensing for a variety of investigations. For example, Dutra and Huber (1999) use the J-M distance to measure the discriminating power associated with each set of features and to rank these features, with the aim of deleting the four worst features from total 14 features that are extracted from ERS1/2 SAR data. A comparative study reported by Mausel *et al.* (1990) assesses the performances of separability indices for finding the best subset of four bands from eight-band image data. Exhaustive search was used to evaluate all 70 possible four-band subset combinations with respect to supervised maximum likelihood classification. Transformed divergence and the Jeffries-Matusita both found the best solution from the 70 subsets, which gave the highest classification accuracy. On the other hand, the Bhattacharyya distance and divergence picked the eleventh and twenty-sixth ranked four-band subset solutions, respectively.

The Jeffries-Matusita distance was employed in a sequential forward selection and genetic algorithm processes to determine the optimum eight bands for the test datasets. This process resulted in band combinations of 11-18-10-24-20-19-5-4 and 2-5-11-13-15-18-20-24 for the first dataset; 10-16-17-23-20-15-9-19 and 9-10-16-17-18-19-20-23 for the second dataset employing SFS and GA respectively. These solutions were then used to form pattern files by selecting the bands in order. Next, the three network structures were trained using the training set and then the generalisation capabilities of the trained networks were tested using the test pattern file. The results are given in Table 5.17 and Table 5.18 for the first dataset, and in Table 5.19 and Table 5.20 for the second dataset.

For the first dataset, the GA solution gave considerably better results than the SFS solution in terms of classification accuracy. Specifically, the solution found by GA yielded around 92% overall accuracy and 90% Kappa coefficient, whilst the solution found by SFS gave results around 90% overall accuracy and 88% Kappa coefficient. It should be noted that both tables suggest the optimum number of iterations as 2,500.

Table 5.17 ANN classification results for the band combination attained by the Jeffries-Matusita distance using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	86.98	84.30	89.88	87.78	91.15	89.30
5000	85.80	82.98	87.88	85.43	90.74	88.87
7500	86.43	83.70	86.93	84.33	89.93	87.92
10000	85.66	82.82	86.48	83.83	89.70	87.66
12500	85.84	82.88	86.84	84.23	89.29	87.19
15000	86.39	83.51	88.02	85.65	89.16	87.02

D: 322.154      TD: 1999.846      B: 14.898      JM: 1413.035

Table 5.18 ANN classification results for the band combination attained by the Jeffries-Matusita distance using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	92.24	90.55	93.28	91.82	92.42	90.80
5000	89.75	87.60	92.24	90.54	90.74	88.82
7500	90.38	88.37	92.42	90.77	91.70	89.93
10000	90.56	88.57	91.70	89.90	90.29	88.25
12500	90.84	88.90	90.79	88.82	92.15	90.46
15000	90.79	88.84	91.20	89.28	91.47	89.65

D: 356.669      TD: 1999.903      B: 14.783      JM: 1413.481

The results given in Table 5.19 and Table 5.20 include the accuracy assessment of ANN classifications using sequential forward selection and genetic algorithm search procedures for the second dataset. Although the highest accuracy achieved is 90% in both tables, the results in Table 5.20 show consistency in terms of the network structures involved. In other words, the solution found by GA provided

almost the same accuracy (90% overall accuracy) for the three network structures considered in this study. One again over-training reduced the classification accuracy slightly. Therefore, it can be stated that 2,500 or 5,000 iterations are relevant to produce a network with high generalisation capabilities.

Table 5.19 ANN classification results for the band combination attained by the Jeffries-Matusita distance using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	88.29	86.62	89.89	88.40	90.34	88.92
5000	88.69	87.07	89.94	88.46	90.40	88.97
7500	88.63	87.02	89.94	88.46	90.29	88.85
10000	88.97	87.40	89.83	88.33	90.23	88.76
12500	89.03	87.46	89.54	88.02	90.34	88.89
15000	89.09	87.51	89.49	87.95	90.11	88.63

D: 128.542      TD: 1999.177      B: 5.188      JM: 1399.968

Table 5.20 ANN classification results for the band combination attained by the Jeffries-Matusita distance using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	90.34	88.92	90.86	89.51	90.57	89.18
5000	90.86	89.49	90.86	89.50	90.40	89.00
7500	90.69	89.27	90.40	88.98	90.46	89.06
10000	90.69	89.28	90.51	89.10	90.17	88.74
12500	90.34	88.91	90.34	88.91	90.06	88.61
15000	90.40	88.97	90.34	88.91	90.06	88.62

D: 138.215      TD: 1998.992      B: 5.352      JM: 1400.412

## 5.5 Statistical Tests

Several multivariate statistical test techniques can be employed to estimate the degree of discrimination between the classes in a dataset, using the means and variance-covariance matrices of the classes. The two most popular statistics of such techniques are Hotelling's  $T^2$  statistic and Wilks'  $\Lambda$  criterion. These techniques assume that the data are multivariate-normally distributed. When they are used as descriptive statistical tests, they estimate the discriminating power of a feature (or relative importance of a feature) and when they are used as fitness measure (evaluation function) they are used to stop the feature selection process.

### 5.5.1 Hotelling's $T^2$

Hotelling's  $T^2$  statistic is used to test the null hypothesis that the multivariate means of the two groups under study do not differ significantly. It provides a multivariate generalisation of the Student's  $t$  test and is related to the problem of how best to discriminate between two groups.  $T^2$  is calculated from:

$$T^2 = \frac{n_1 n_2}{n_1 + n_2} \cdot D^2 \quad (5.5)$$

$$D^2 = (\bar{x}_1 - \bar{x}_2)^T S^{-1} (\bar{x}_1 - \bar{x}_2) \quad (5.6)$$

where  $D^2$  is the coefficient known as Mahalanobis'  $D$ -squared. It is a measure of the overall similarity between the two groups.  $S^{-1}$  is the inverse matrix of the pooled variance-covariance matrix  $S$ , and  $\bar{x}_1$  and  $\bar{x}_2$  are mean vectors for the groups, which contain  $n_1$  and  $n_2$  individuals, respectively.

The value of Hotelling's  $T^2$  increases as inter-class separation increases. The statistical significance of  $T^2$  statistic can be evaluated using a transformation to the  $F$  distribution. It should be noted that the number of observations need not be the same for the two samples, but the number of features must be the same.

According to Overall and Klett (1972), Hotelling's  $T^2$  statistic can be used in following situations:

- 1) equivalence of multivariate mean vectors derived from two independent samples,
- 2) equivalence of multivariate mean vectors for paired observations, such as derived from test-retest situation,
- 3) equivalence of sample mean vector to hypothesised population mean vector,
- 4) several types of tests that are peculiar to the multivariate situation.

Hotelling's  $T^2$  was also employed in the process of searching the best bands to recognise land cover classes for two problems. For the first one 11-18-23-22-17-13-15-24 and 6-11-13-17-18-21-22-23 band combinations were found as a result of applying SFS and GA methods. These solutions were tested to determine their performance, which are presented here as Table 5.21 and Table 5.22. As can be seen from both tables, overall accuracy of over 92% and Kappa coefficient of over 90% have been achieved by the solutions, which are higher than all the previous results produced by separability indices.

Table 5.21 ANN classification results for the band combination attained by Hotelling's  $T^2$  using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	92.15	90.45	92.60	91.02	91.43	89.59
5000	91.56	89.72	91.02	89.15	90.65	88.65
7500	91.52	89.68	89.93	87.82	90.97	89.01
10000	91.92	90.16	90.38	88.36	92.70	91.06
12500	91.70	89.87	89.52	87.30	93.60	92.16
15000	90.97	88.98	90.84	88.87	94.01	92.67

$$T^2 = 20491.29$$

The network structure of 8-10-7 trained for the solution found by SFS based on Hotelling's  $T^2$  separability measure was employed to classify the image from which the first test dataset was derived. The classified image is given in Figure 5.11 representing each class with four levels (tones) of the colour assigned to land cover classes.

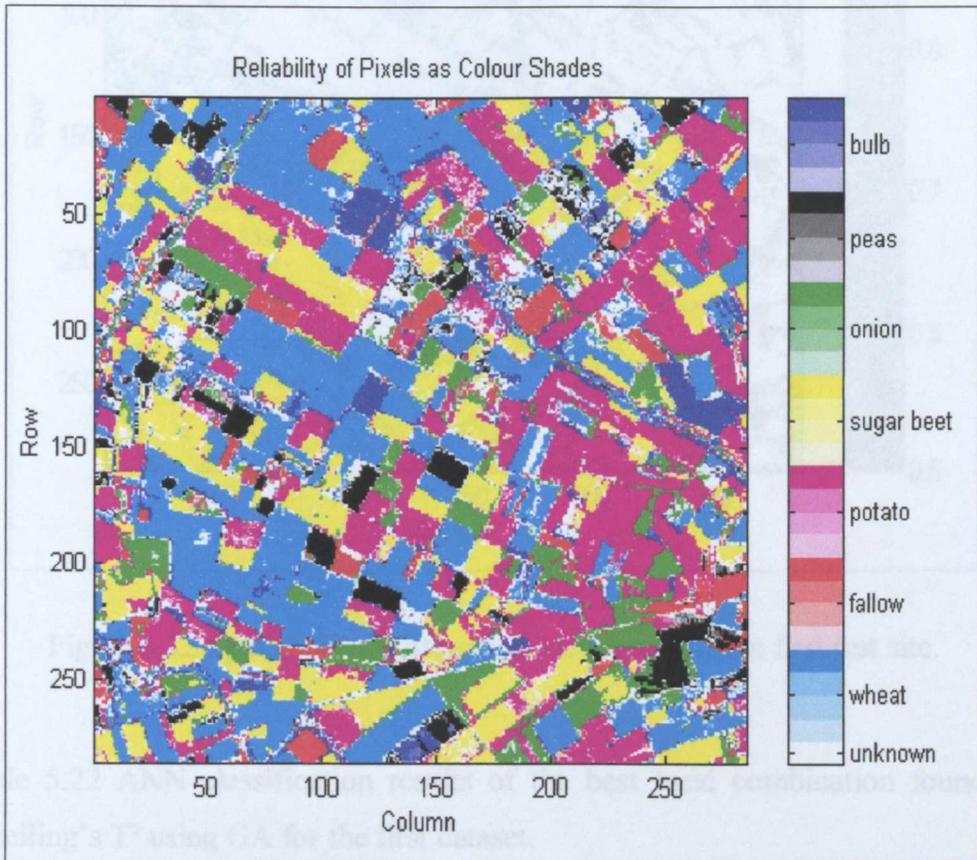


Figure 5.11 ANN classification result for the solution found by SFS based on Hotelling's  $T^2$  for the first test site using bands 11-18-23-22-17-13-15-24.

The result file, used to generate ANN classification results (Figure 5.11) for whole test image, was also used to produce the map of output activation levels in terms of using tones of grey colour to represent spatial accuracy. The result is shown in Figure 5.12. As can be noticed from the figure, there are some fields in black, which are not known by the trained network. It is most likely that these fields contain crops that are not included in the training set.

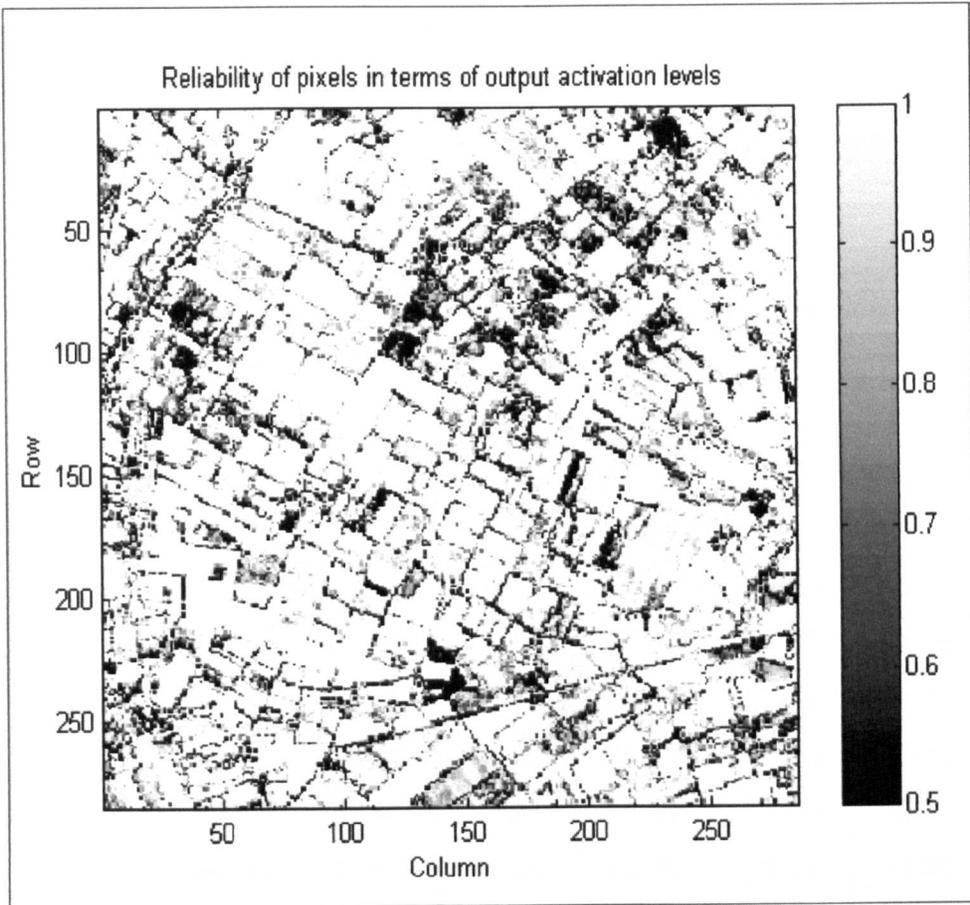


Figure 5.12 Spatial pattern of output activations for the first test site.

Table 5.22 ANN classification results of the best band combination found by Hotelling's  $T^2$  using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	91.56	89.71	92.15	90.50	91.65	89.88
5000	90.34	88.28	92.29	90.68	91.92	90.18
7500	91.06	89.12	92.10	90.46	91.88	90.11
10000	91.29	89.40	92.10	90.47	92.15	90.43
12500	91.06	89.13	92.33	90.73	92.47	90.83
15000	90.20	88.13	92.15	90.51	92.33	90.67

$T^2 = 20240.08$

For the second set band combinations of 11-9-20-18-15-23-17-16 and 9-10-15-16-17-18-20-23 were found by the SFS and GA approaches, respectively. Results of applying these solutions to neural networks are given in Tables 5.23 and 5.24. The accuracy values in these tables show that solutions found by Hotelling's  $T^2$  are again better than the solutions found by separability indices.

Table 5.23 ANN classification results for the band combination found by Hotelling's  $T^2$  using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.43	87.91	91.20	89.90	91.43	90.16
5000	90.11	88.67	91.37	90.08	91.49	90.23
7500	90.51	89.11	91.09	89.76	91.60	90.42
10000	90.06	88.61	90.86	89.51	91.49	90.23
12500	90.40	88.97	90.86	89.50	91.71	90.48
15000	90.23	88.81	90.74	89.39	91.77	90.54

$$T^2 = 4497.84$$

Table 5.24 ANN classification results of the best band combination found by Hotelling's  $T^2$  using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.49	87.93	90.86	89.51	90.86	89.51
5000	90.17	88.69	91.14	89.83	90.57	89.20
7500	90.63	89.20	90.91	89.57	90.91	89.58
10000	90.57	89.14	90.97	89.63	91.14	89.83
12500	90.63	89.20	90.69	89.32	91.20	89.91
15000	90.80	89.40	90.86	89.52	91.26	89.97

$$T^2 = 4546.78$$

An ANN-based classification process was applied to the test image for the second test site to assign each pixel to a land cover class under the condition that the output activation is higher than 0.5. If this is not the case, the pixel is labelled as unknown. The classified image is shown in Figure 5.13, and the corresponding grey scale activation level analysis is shown in Figure 5.14.

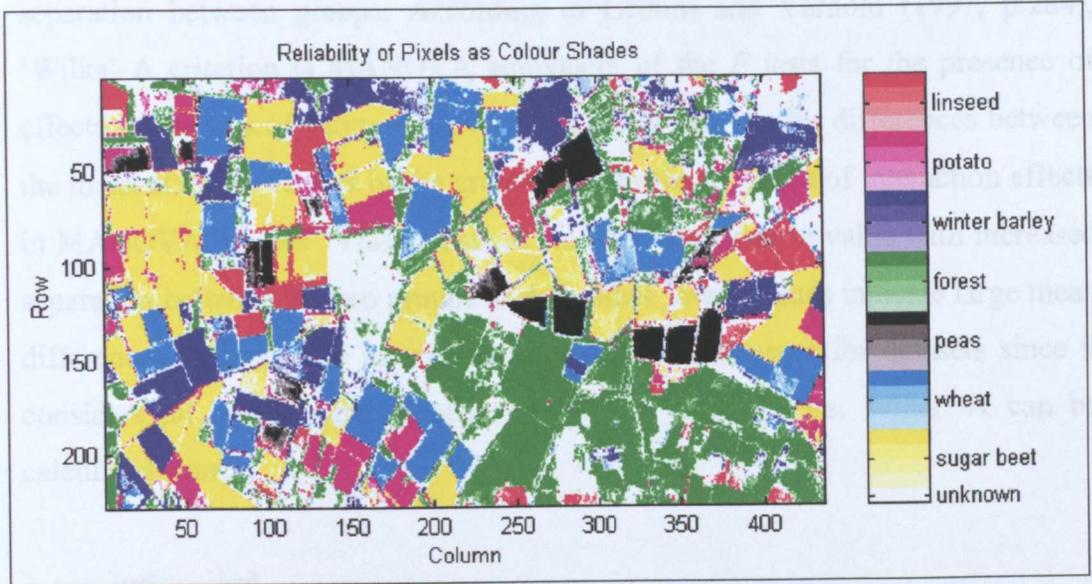


Figure 5.13 ANN classification result for the solution found by SFS based on Hotelling's  $T^2$  for the second test site using bands 11-9-20-18-15-23-17-16.

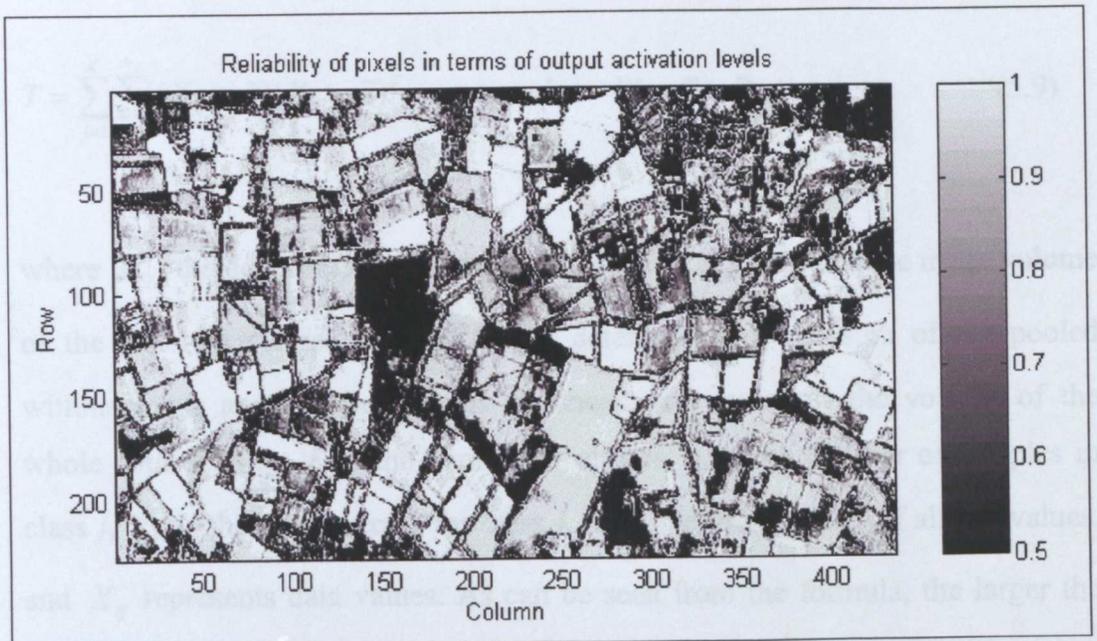


Figure 5.14 Spatial pattern of output activations for the second site.

### 5.5.2 Wilks' $\Lambda$ Criterion

Wilks'  $\Lambda$  criterion, introduced by Samuel S. Wilks in 1932, is a basic multivariate analysis of variance (MANOVA) test for equality of group means. It is also called the Wilks' lambda likelihood-ratio criterion. It provides a measure of degree of separation between groups. According to Grimm and Yarnold (1997, p.284), 'Wilks'  $\Lambda$  criterion is MANOVA equivalent of the  $F$  tests for the presence of effects in analysis of variance (ANOVA) models and test for differences between the mean attribute vectors of the groups and for the presence of interaction effects in MANOVA models'. This statistical measure decreases in value with increased separation between the two groups of data, since lower values indicate large mean differences. Wilks'  $\Lambda$  is an overall discrimination measure for datasets since it considers all bands simultaneously instead of feature pairs. Wilks'  $\Lambda$  can be calculated from:

$$\Lambda = \frac{|W|}{|W + B|} = \frac{|W|}{|T|} \quad (5.7)$$

$$B = \sum_{j=1}^K n_j (\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})^T \quad (5.8)$$

$$T = \sum_{j=1}^K \sum_{i=1}^{n_j} (X_{ij} - \bar{x})(X_{ij} - \bar{x})^T \quad \text{and} \quad W = T - B \quad (5.9)$$

where  $|W|$ , the determinant of within-groups variance, measures the mean volume of the different classes,  $|W + B|$  is the determinant of the sum of the pooled within-groups and between-groups variances, and measures the volume of the whole dataset. Also,  $K$  is the number of classes,  $n_j$  is the number of samples in class  $j$ ,  $\bar{x}_j$  is the mean vector for class  $j$ ,  $\bar{x}$  is the grand mean of all the values, and  $X_{ij}$  represents data values. As can be seen from the formula, the larger the distance between the groups the larger the denominator. The value of  $\Lambda$  will, therefore, reduce as inter-group separation increases. Wilks'  $\Lambda$  can have values ranging from zero to one.

When Wilks'  $\Lambda$  criterion was used in SFS and GA search methods as the fitness measure, solutions of 18-11-24-13-16-17-8-19 and 10-11-13-16-17-18-19-20 were found respectively. The three network structures were trained and tested for these solutions, and the results of testing these networks can be found in Table 5.25 and Table 5.26. The high accurate results, over 92% overall accuracy and 91% Kappa coefficient, indicate better definition for the optimum subset selection problem than any other method used earlier. This goes to show the effectiveness of the Wilks'  $\Lambda$  criterion for the particular problem considered.

Table 5.25 ANN classification results for the band combination attained by Wilks'  $\Lambda$  using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	92.92	91.37	90.84	88.87	91.52	89.66
5000	93.42	91.97	91.56	89.75	91.52	89.70
7500	93.19	91.69	91.92	90.20	92.20	90.50
10000	92.79	91.20	91.83	90.11	92.29	90.60
12500	92.15	90.44	92.33	90.68	91.56	89.79
15000	91.97	90.20	91.92	90.21	91.83	90.09

$$\Lambda = 0.000097$$

The 8-10-7 network structure trained for the solution found by SFS using Wilks'  $\Lambda$  criterion as the fitness measure was employed to classify the test image for the first site. The result of this process is shown in Figure 5.15. A grey scale map (Figure 5.16) showing the output activation levels was also produced for the same site. Both figures comply with the accurate results given in Table 5.25 in that the highest accuracy of 93.79% overall accuracy is achieved. It should also be noted that the best results were produced by the smallest network structure (8-10-7).

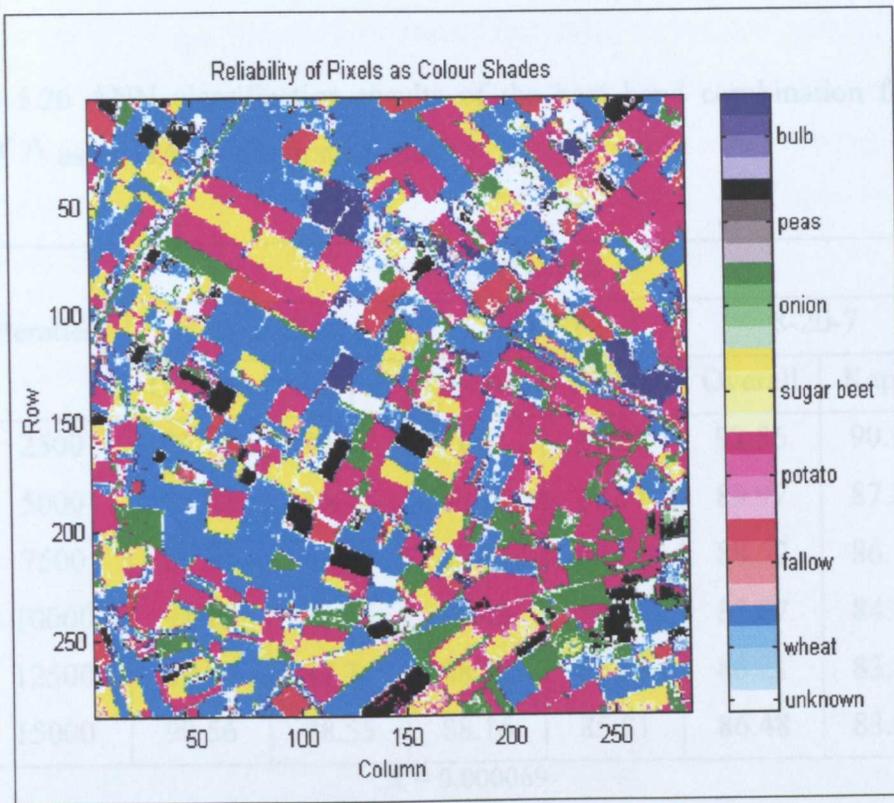


Figure 5.15 ANN classification result for the solution found by SFS based on Wilks'  $\Lambda$  for the first test site using bands 18-11-24-13-16-17-8-19.

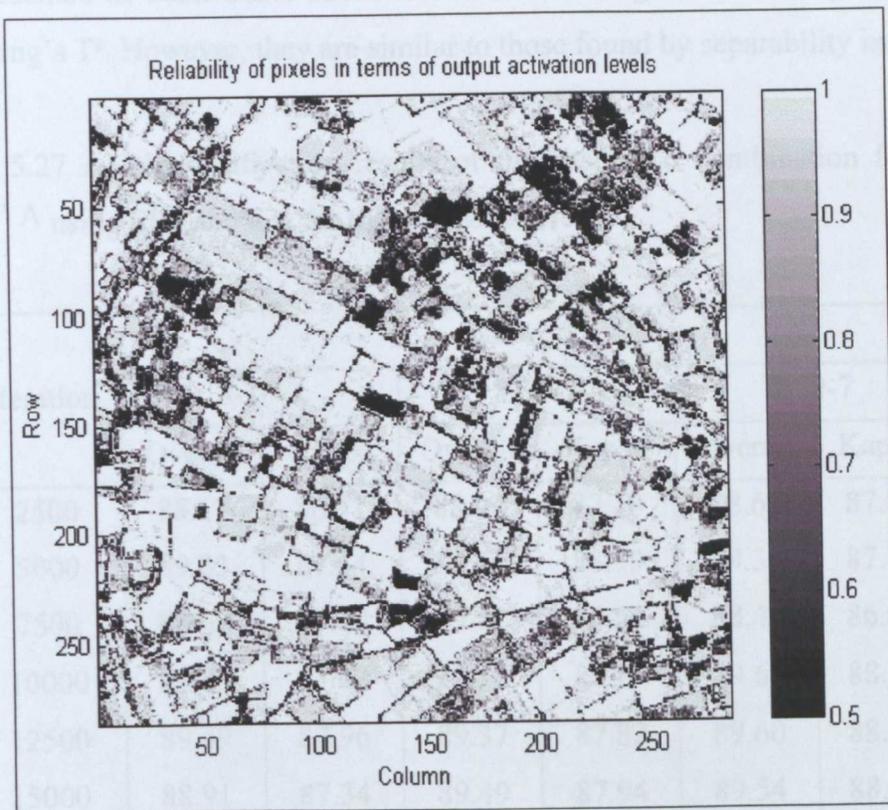


Figure 5.16 Spatial pattern of output activations for the first site.

Table 5.26 ANN classification results of the best band combination found by Wilks'  $\Lambda$  using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	91.74	89.93	91.61	89.80	92.56	90.91
5000	89.02	86.72	90.70	88.71	89.97	87.81
7500	89.93	87.80	89.97	87.88	88.52	86.16
10000	91.15	89.25	90.43	88.41	87.07	84.56
12500	89.88	87.74	88.88	86.60	86.21	83.57
15000	90.56	88.55	88.16	85.81	86.48	83.93

$$\Lambda = 0.000069$$

For the second dataset, both SFS and GA techniques reached the same solution (3-9-11-14-15-16-17-23). The results of using this solution in the ANN classification are presented in Table 5.27. These results are not as good as those produced by Hotelling's  $T^2$ . However, they are similar to those found by separability indices.

Table 5.27 ANN classification results of the best band combination found by Wilks'  $\Lambda$  using SFS and GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	88.17	86.51	88.97	87.38	88.63	87.04
5000	89.03	87.44	89.71	88.19	89.31	87.80
7500	88.69	87.09	89.77	88.26	88.40	86.82
10000	88.63	87.02	89.71	88.20	89.66	88.17
12500	89.49	87.96	89.37	87.82	89.60	88.11
15000	88.91	87.34	89.49	87.94	89.54	88.04

$$\Lambda = 0.0029998$$

An ANN classification process was carried out using the trained network for the test image of the second site, and the resulting image is shown in Figure 5.17. It can be seen that the classifier could not recognise pixels in the lower right corner of the image. This is due to the limited extent of the SIR-C SAR coverage that is involved in the classification process. Also, output activation levels were mapped using grey colour tones to observe the problematic areas as well as best classified (clearly defined) areas. Output from this process is presented in Figure 5.18.

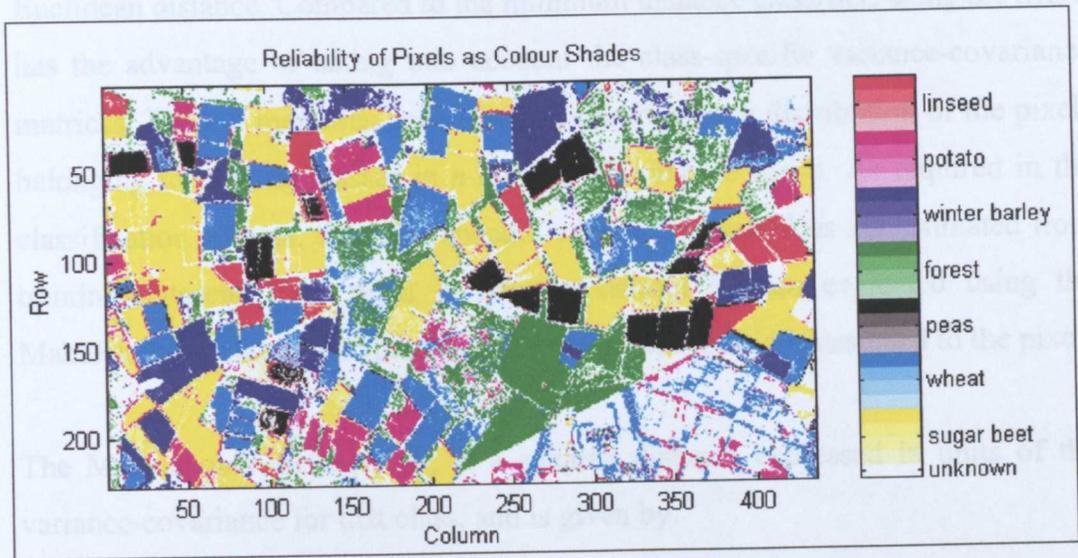


Figure 5.17 ANN classification result for the solution found by SFS based on Wilks'  $\Lambda$  for the second test site using bands 3-9-11-14-15-16-17-23.

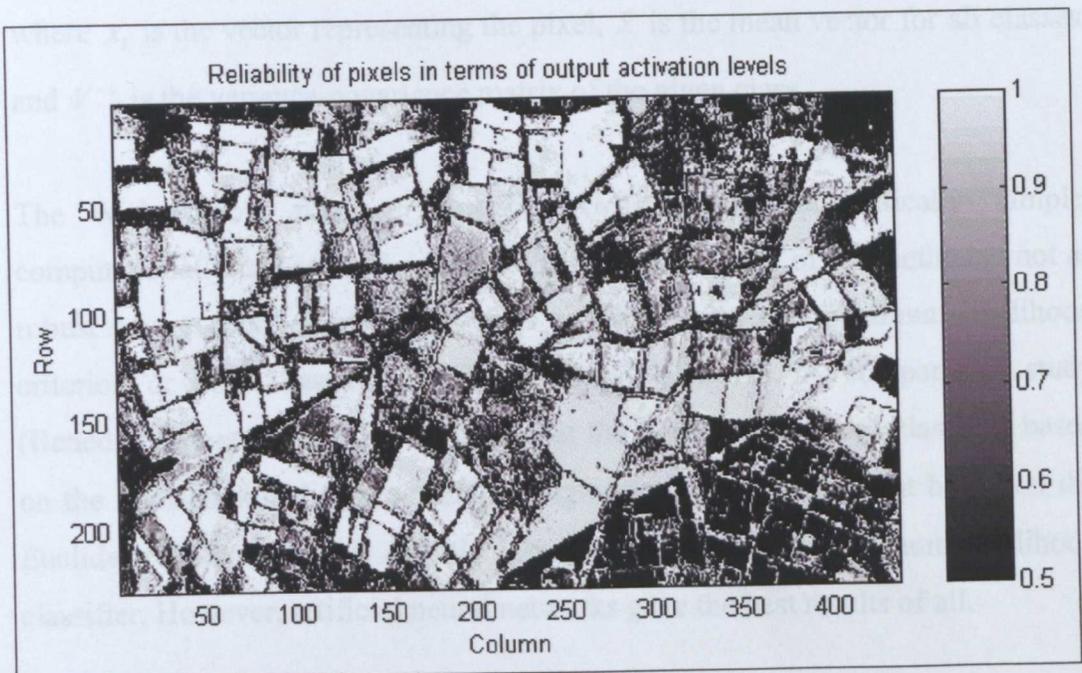


Figure 5.18 Spatial pattern of output activations for the second site.

## 5.6 Mahalanobis Distance Classifier

As an example of the wrapper approach, the Mahalanobis distance classifier (MDC), which is a supervised classification algorithm, is employed to determine the most effective eight bands. The criterion to determine class membership of a pixel is to find the minimum Mahalanobis distance between the pixel and the class centres, in a way similar to the minimum distance classifier that is based on the Euclidean distance. Compared to the minimum distance classifier, using the MDC has the advantage of taking into account the class-specific variance-covariance matrices. Thus, it measures and considers the frequency distribution of the pixels belonging to training classes in  $n$ -dimensional feature space. As required in the classification process, the mean spectral vectors for each class are estimated from training datasets. Each pixel in the test dataset is then evaluated using the Mahalanobis distance, and the label of the closest centroid is assigned to the pixel.

The Mahalanobis distance,  $D^2$ , is a squared distance expressed in units of the variance-covariance for that class, and is given by:

$$D^2 = (x_i - \bar{x})^T V^{-1} (x_i - \bar{x}) \quad (5.10)$$

where  $x_i$  is the vector representing the pixel,  $\bar{x}$  is the mean vector for all classes, and  $V^{-1}$  is the variance-covariance matrix of the given class.

The Mahalanobis distance classifier (MDC) is mathematically simple, computationally fast and efficient, but the theoretical basis of the method is not as robust as the complex classifiers such as those using the maximum likelihood criterion or those based on artificial neural networks. A comparative study (Benediktsson *et al.*, 1990) concludes that the minimum distance classifier based on the Mahalanobis distance performs significantly better than that based on the Euclidean distance, but is slightly less powerful than the maximum likelihood classifier. However, artificial neural networks gave the best results of all.

In this study, while evaluating the results of the MDC, three accuracy criteria were estimated so as to find the best one for improved ANN results. The first criterion is the average accuracy, which is the mean of all individual class accuracies. The main reason for using such a measure is to minimise the effect of the highest individual accuracy, and maximise the effect of the lowest individual accuracy. The second criterion is the overall accuracy, which is the most popular accuracy measure used. It is estimated by dividing the total number of correctly classified pixels by the total number of pixels used for testing. The third criterion can be called the quality measure that aims to find minimum difference between the highest and lowest individual class accuracies. The purpose of using such a measure is to find a solution that somehow improves the poorest class accuracy. This measure is calculated by:

$$qual = (max - min) / aver \quad (5.11)$$

where *max* is the maximum individual class accuracy, *min* is the minimum individual class accuracy and *aver* is the average accuracy obtained.

These three criteria were used as fitness measures to search for the best band combinations for two datasets in both sequential forward selection (SFS) and genetic algorithm (GA) search methods. The results of using MDC based on average accuracy, overall accuracy and quality measure (Equation 5.11) in SFS are given in Tables 5.28, 5.29 and 5.30, respectively. In addition, the results of using these criteria in GA are presented in Tables 5.31, 5.32 and 5.33.

In order to portray the differences between MDC and ANN classification results, result images are given for the first and second test sites for both techniques. The solutions (5-13-10-15-1-18-6-3 and 14-10-16-17-23-11-21-15) found by SFS method using MDC based on overall accuracy for both datasets were used to form pattern files. For the classification of the test images MDC and ANN classifiers were applied and the result images are given here as Figure 5.19 and Figure 5.20 respectively. Also, MDC and ANN classification results for the second test site are shown in Figure 5.21 and Figure 5.22.

The results given in Tables 5.28, 5.29 and 5.30 may indicate that the accuracies achieved by MDC method were similar to (or even better than) those produced by ANN method. This is because of the fact that no threshold is set for class membership allocation in MDC method whereas a threshold of 50% is used in ANN-based classifications. When no threshold is set for the class membership in ANN method, at least 3% improvement is observed in the classification accuracy.

Table 5.28 ANN classification results of the best solution (18-11-15-10-12-21-4-14) found by MDC based on average accuracy using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	89.79	87.67	90.61	88.67	90.47	88.42
5000	90.02	87.90	91.02	89.11	90.02	87.88
7500	89.43	87.21	90.65	88.66	90.20	88.07
10000	89.07	86.78	90.25	88.18	89.16	86.80
12500	88.97	86.65	90.25	88.17	89.02	86.65
15000	89.38	87.11	90.70	88.70	88.79	86.38

Av : 0.9131      Ov : 0.9183      Qu : 9.770

Table 5.29 ANN classification results of the best solution (5-13-10-15-1-18-6-3) found by MDC based on overall accuracy using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	90.38	88.31	88.57	86.10	89.97	87.77
5000	88.11	85.63	87.84	85.20	88.66	86.20
7500	88.43	86.01	88.07	85.48	88.07	85.49
10000	88.16	85.67	87.21	84.46	87.11	84.36
12500	88.52	86.10	86.71	83.90	87.02	84.27
15000	88.66	86.24	85.30	82.22	87.25	84.58

Av : 0.9090      Ov : 0.9170      Qu : 9.792

Table 5.30 ANN classification results of the best band combination (20-19-23-11-4-7-8-9) found by MDC based on quality measure using SFS for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	83.44	80.43	84.48	81.62	86.48	83.86
5000	86.98	84.49	84.71	81.49	87.11	84.63
7500	86.16	83.51	84.71	81.90	87.21	84.71
10000	86.12	83.42	84.12	81.27	87.21	84.72
12500	86.57	84.04	84.57	81.72	87.02	84.45
15000	85.03	82.12	85.30	82.61	86.52	83.92

Av : 0.9130      Ov : 0.9074      Qu : 9.855

Table 5.31 ANN classification results of the best band combination (2-5-11-15-16-17-18-23) found by MDC based on average accuracy using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	84.85	82.01	87.52	85.04	86.12	83.43
5000	85.30	82.54	87.89	85.43	87.07	84.56
7500	84.80	81.94	87.43	84.82	87.16	84.63
10000	84.30	81.36	87.34	84.71	86.07	83.36
12500	84.53	81.58	86.16	83.35	86.34	83.52
15000	84.39	81.42	86.52	83.72	86.16	83.35

Av : 0.9247      Ov : 0.9283      Qu : 9.802

The solutions found by the GA also suggest that MDC method performed well in terms of producing high-accurate results. However, the involvement of the threshold parameter in ANN-based classifications should be taken into consideration when comparing the results produced by the two classifiers.

Table 5.32 ANN classification results of the best band combination (10-11-15-16-18-19-20-21) found by MDC based on overall accuracy using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	90.38	88.31	89.11	86.74	89.56	87.39
5000	91.79	89.98	89.07	86.67	88.75	86.47
7500	90.84	88.85	88.11	85.59	89.52	87.35
10000	91.11	89.16	88.02	85.49	89.84	87.70
12500	90.38	88.31	86.93	84.19	89.11	86.85
15000	90.34	88.26	86.34	83.51	88.29	85.90
Av : 0.9126		Ov : 0.9215		Qu : 9.787		

Table 5.33 ANN classification results of the best band combination (2-5-6-11-12-15-19-20) found by MDC based on quality measure using GA for the first dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	90.20	88.03	89.84	87.65	89.88	87.65
5000	88.25	85.74	88.84	86.50	87.57	84.98
7500	88.61	86.11	88.70	86.30	86.34	83.61
10000	87.98	85.36	88.93	86.54	86.57	83.88
12500	87.66	84.97	87.75	85.17	87.11	84.47
15000	87.66	84.96	87.84	85.21	85.84	83.01
Av : 0.8917		Ov : 0.9002		Qu : 9.787		

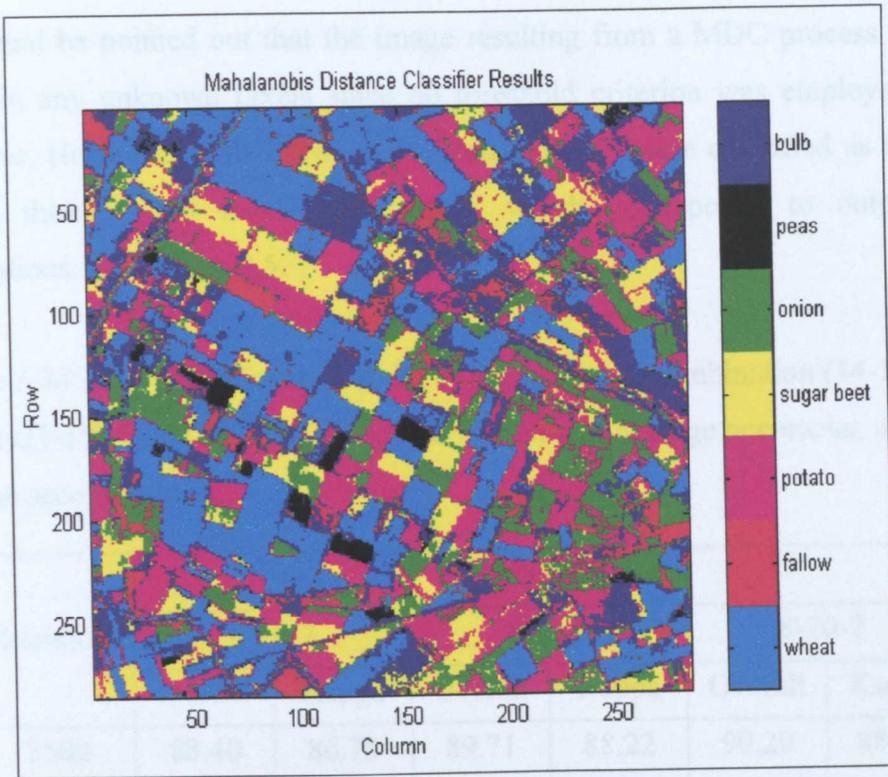


Figure 5.19 MDC classification results using the 5-13-10-15-1-18-6-3 solution for the first site.

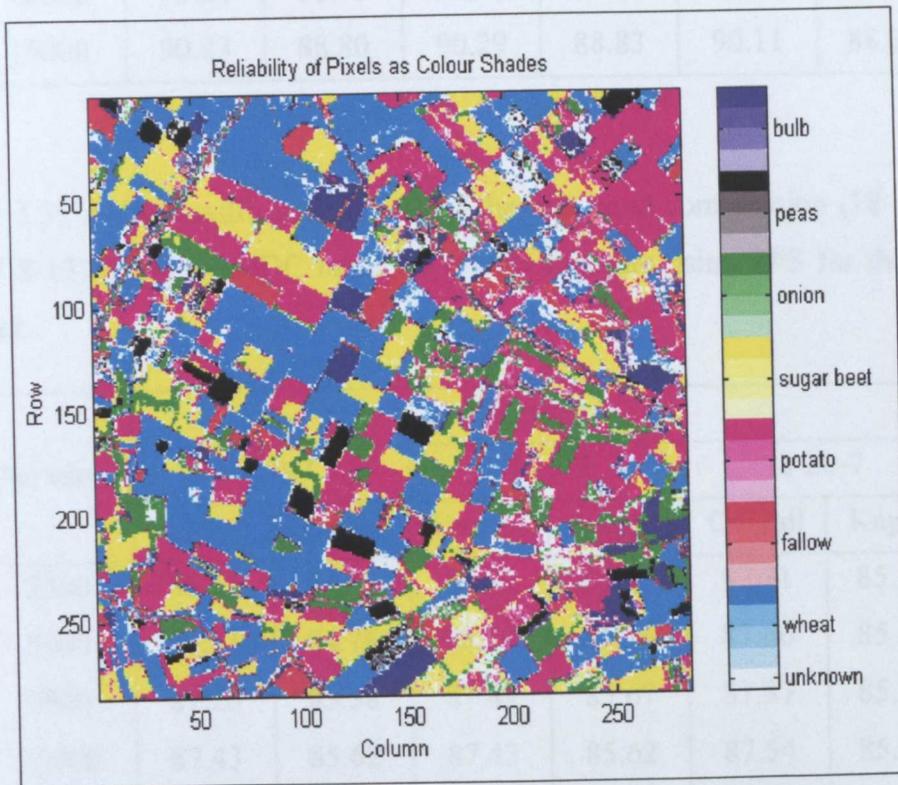


Figure 5.20 Artificial neural network classification results using the 5-13-10-15-1-18-6-3 solution for the first site.

It should be pointed out that the image resulting from a MDC process does not contain any unknown pixels since no threshold criterion was employed in the process. However, in the ANN classification pixels were classified as unknown when their highest membership value, which corresponds to output node activations, is less than 0.5.

Table 5.34 ANN classification results of the best band combination (14-10-16-17-23-11-21-15) found by MDC based on overall and average accuracies using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	88.40	86.75	89.71	88.22	90.29	88.85
5000	89.09	87.50	90.23	88.78	90.63	89.23
7500	89.83	88.35	90.34	88.90	89.89	88.39
10000	90.00	88.52	90.80	89.41	90.11	88.67
12500	90.34	88.92	90.63	89.21	90.40	88.96
15000	90.23	88.80	90.29	88.83	90.11	88.65

Av = Ov : 0.9091      Qu : 9.9208

Table 5.35 ANN classification results of the best band combination (18-12-9-22-21-17-8-13) found by MDC based on quality measure using SFS for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	86.29	84.39	87.60	85.82	87.31	85.52
5000	86.63	84.76	87.20	85.36	87.66	85.89
7500	87.20	85.38	87.49	85.67	87.37	85.58
10000	87.43	85.62	87.43	85.62	87.54	85.75
12500	87.26	85.43	87.60	85.81	88.06	86.32
15000	87.26	85.42	87.14	85.32	87.71	85.94

Ov : 0.8857      Qu : 9.9548

Please note that for the second dataset average and overall accuracy are the same because the same number of samples (250 pixels) were selected for each class unlike the first dataset. Therefore, search techniques were employed on the basis of overall accuracy and the quality measure.

Table 5.36 ANN classification results of the best band combination (4-8-11-15-16-17-20-23) found by MDC based on overall and average accuracies using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	87.89	86.20	90.91	89.56	90.80	89.45
5000	87.94	86.26	91.14	89.80	91.20	89.88
7500	89.43	87.91	91.09	89.74	90.63	89.25
10000	89.54	87.99	91.03	89.67	91.03	89.68
12500	89.20	87.62	90.80	89.41	91.14	89.81
15000	89.37	87.81	90.91	89.54	91.03	89.69

Ov : 0.9097      Qu : 9.9209

Table 5.37 ANN classification results of the best band combination (9-10-12-13-14-17-22-23) found by MDC based on quality measure using GA for the second dataset.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	87.83	86.05	88.57	86.91	88.86	87.21
5000	88.51	86.82	88.63	87.00	89.49	87.91
7500	87.83	86.07	88.51	86.86	88.63	86.97
10000	87.89	86.13	88.06	86.36	89.09	87.48
12500	87.89	86.13	88.34	86.66	89.26	87.66
15000	88.34	86.63	87.89	86.15	89.20	87.61

Ov : 0.9017      Qu : 9.9689

It can be observed from the results shown in Tables 5.34, 5.35 and 5.37 that the performance of the ANN method was slightly better than the MDC method, considering the involvement of threshold factor in the ANN process. The best ANN performance against the MDC method was produced when the GA based on overall accuracy was used to determine best eight-band combination (Table 5.36).

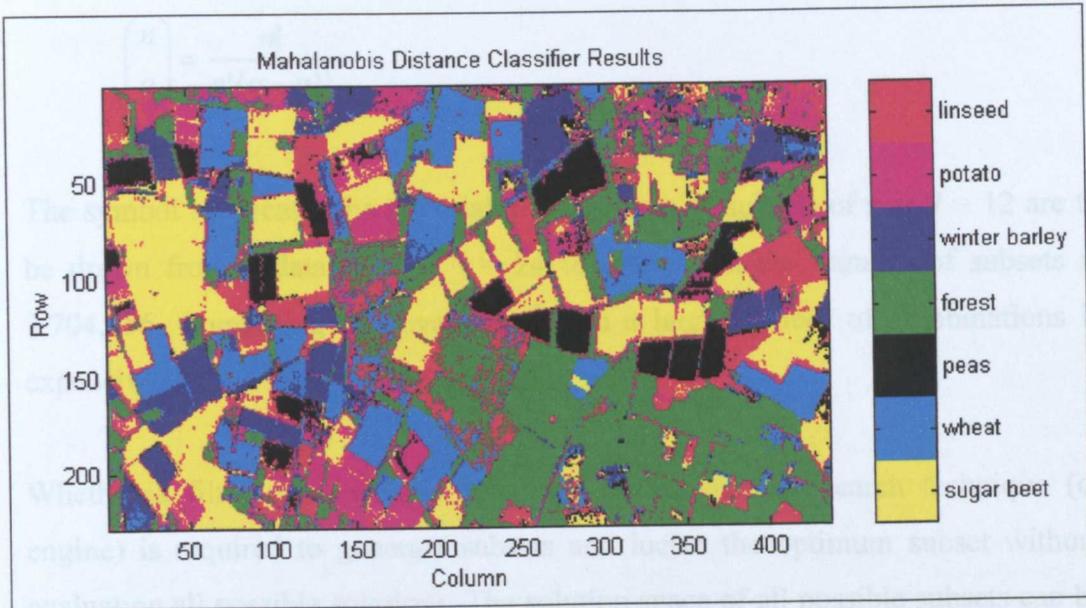


Figure 5.21 MDC classification results using the 14-10-16-17-23-11-21-15 solution for the second site.

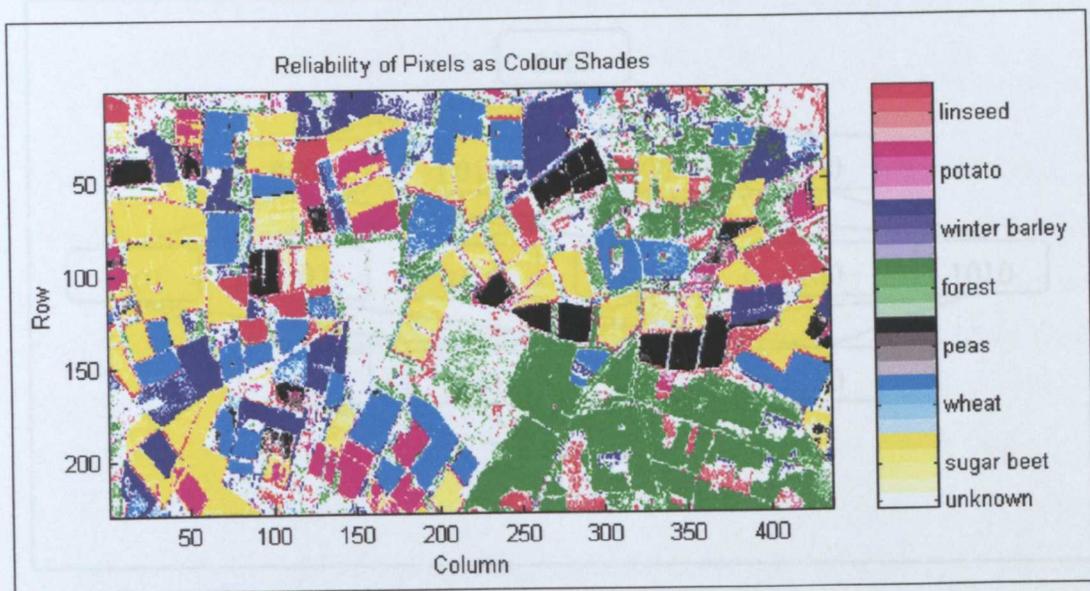


Figure 5.22 Artificial neural network classification results using the 14-10-16-17-23-11-21-15 solution for the second site.

## 5.7 Search Techniques

Separability measures help the analyst to determine the best band combination that is the optimum set of  $q$  bands selected from all  $n$  bands. The number of subsets of size  $q$  drawn from a set of  $n$  objects is given by:

$$\binom{n}{q} = \frac{n!}{q!(n-q)!}$$

The symbol ! indicates the factorial. For example, if subsets of size  $q = 12$  are to be drawn from a dataset with  $n = 24$  features, then the number of subsets is 2,704,156. Computing and evaluating such a large number of combinations is expensive.

Whether a filter or a wrapper approach is employed, a search technique (or engine) is required to generate subsets and locate the optimum subset without evaluating all possible solutions. The solution space of all possible subsets can be represented as a lattice. As an example, a problem with four features is shown as a lattice in Figure 5.23.

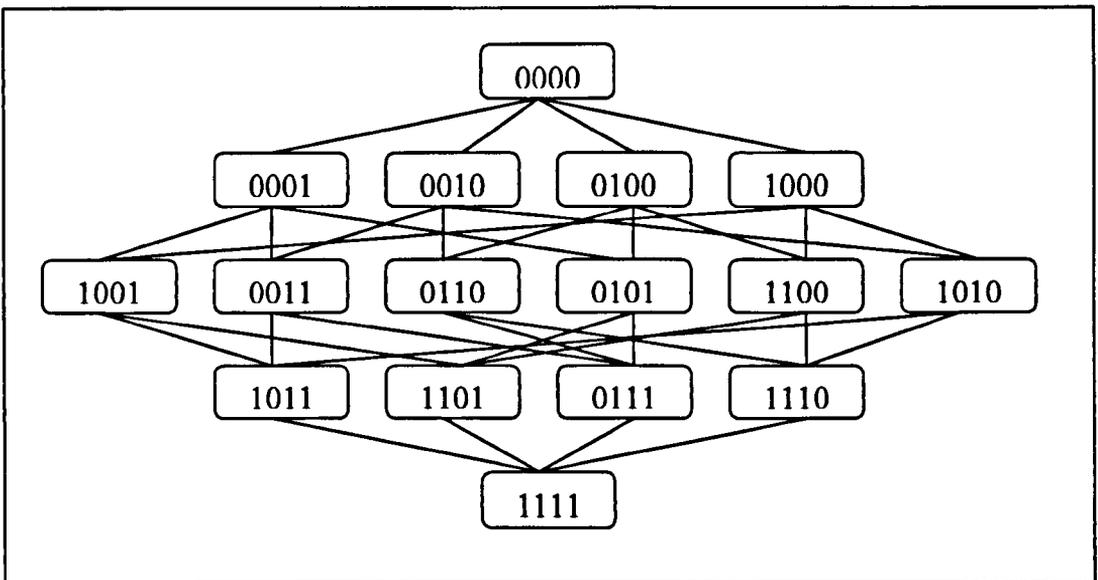


Figure 5.23 The solution space of the feature subset selection problem (from Scott *et al.*, 1998)

Each node in the lattice represents a feature subset, where the value one is used to indicate inclusion and the value zero is used to show exclusion. For example, in the case of six bands, the best subset of three bands found by the genetic algorithms may be represented as 1 0 0 1 0 1. The meaning of this coded solution is that the first, fourth and sixth bands may give best results when they are used together, and the second, third and fifth bands can be ignored because of their insignificant contribution.

Many search strategies have been developed to find the optimum subset. However, there are three basic search approaches used to determine the optimum subset without testing all possible candidate subsets. These are the exponential, randomised and sequential approaches.

Exponential algorithms, such as the branch-and-bound and exhaustive search techniques, have exponential complexity related to the number of features. In other words, the number of possible subsets  $\binom{n}{q}$  grows exponentially depending on the number of features ( $n$ ) and the size of the subsets ( $q$ ). In this approach, a large number of candidate subsets (all possible candidate subsets for the exhaustive search technique) are evaluated in terms of a performance measure. These algorithms are time and computer power demanding, and are not, therefore, preferred especially for high dimensional problems.

Randomised algorithms including genetic algorithms and simulated annealing use randomised steps or sampling processes, and yield results with high accuracies. However, they are not easy to implement as the parameters to be set by the analyst play a crucially important role. Sequential search algorithms, on the other hand, have polynomial complexity. They use simple addition and exclusion rules in implementing the search process. The most popular sequential search techniques are sequential forward selection and sequential backward selection. A taxonomy of feature selection algorithms (search methods) is presented by Zongker and Jain (1996), and is reproduced here as Figure 5.24.

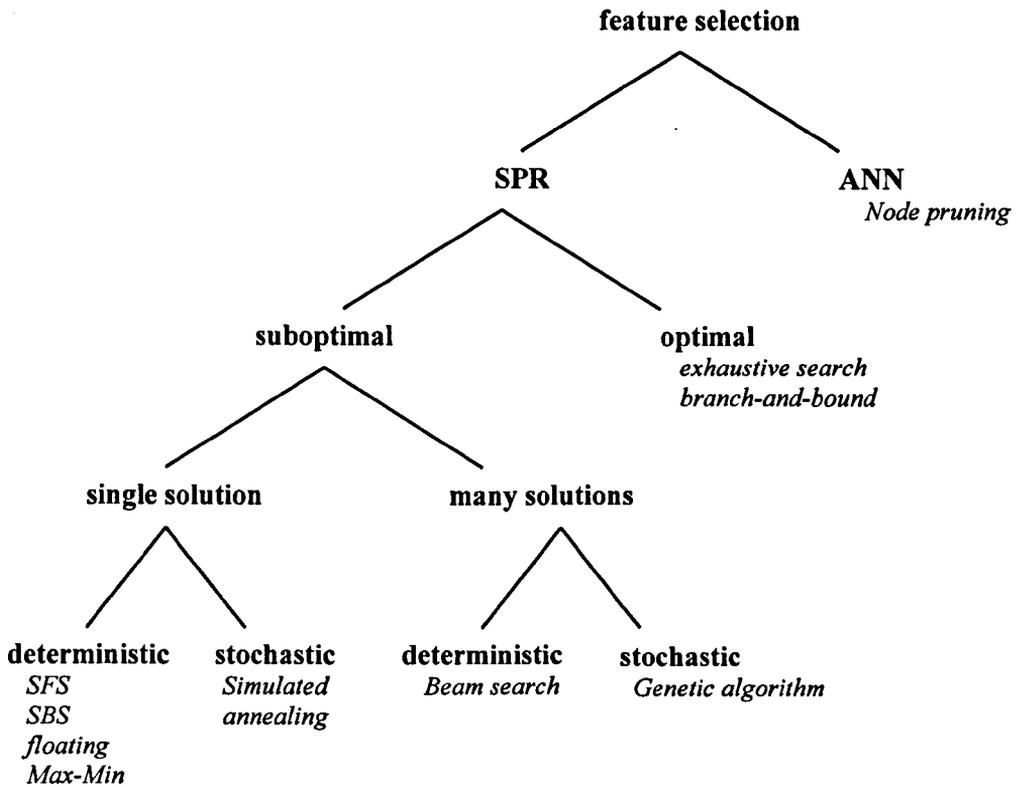


Figure 5.24 A taxonomy of feature selection algorithms (from Zongker and Jain, 1996). SPR: statistical pattern recognition, ANN: artificial neural networks.

### 5.7.1 Sequential Forward Selection and Sequential Backward Selection

The sequential forward selection technique (SFS) starts by finding the best single individual feature and then evaluates the remaining features one at a time to find the second best feature (i.e. the one that gives higher separability than other candidate features). This process continues iteratively until a desired number of features are selected. Unlike the SFS method, sequential backward selection (SBS), also called the sequential backward elimination, starts with the whole dataset and searches for the band that has the least effect when it is removed. In other words, it excludes each band one at a time and finds the least effective one by looking at the values of separability measures. This process is repeated until the required number of bands are selected.

Neither of these procedures is guaranteed to find the optimal subset. It generally produces a sub-optimal solution because of the so-called nesting effect, which

results from the fact that in the forward selection process, selected features cannot be discarded later and in the backward selection process, excluded features cannot be reselected. In fact, both techniques may find different solutions, especially in high dimensional cases. Another important point to be emphasised is that using backward searches with classifiers as evaluation functions may cause high complexity problems as more high dimensional data must be employed in the process. This would also result in a greater computing time requirement compared to SFS. This makes SBS techniques less attractive, especially when a classifier is used for evaluation.

Some extensions of the sequential forward and backward selection approaches are described in the literature. One of these extensions introduced by Pudil *et al.* (1994) is the floating search method that keeps the feature sets flexibly changing so as to approximate the optimal solution as much as possible. In other words, the resulting dimensionality in respective stages of the algorithm is not changing monotonically but is actually ‘floating’ up and down.

Mather (1999a) notes a study published in 1979 by R. Kumar who compared the exhaustive search results with sequential forward and backward selections in his study and found that the forward selection algorithms produced results as good as exhaustive search and better results than those produced by the backward elimination method. Similar results are also reported by Aha and Bankert (1996). They analysed the use of variants of the forward and backward feature selection techniques for the cloud classification problem, using a separability index and the nearest neighbour classifier. Four important conclusions drawn from the study are:

- 1) feature selection improves the accuracy of the classification task,
- 2) backward elimination does not always outperform forward selection, contrary to some claims,
- 3) using classifier accuracy as the evaluation function yields better results than using the separability index,
- 4) the general pattern is that forward selection is preferred when the optimal number of selected features is small, while backward elimination is preferred otherwise.

In a review study, Siedlecki and Sklansky (1988) underline the fact that both forward and backward selection can be easily “derailed”. For instance, the forward selection algorithm can add two features that are subsequently the best ones but which are bad if used together. In order to solve this problem, a bi-directional search technique that combines forward selection and backward elimination is proposed.

### **5.7.2 Branch-and-Bound Search Method**

The only search algorithm available at present that is guaranteed to find the optimal subset solution is the branch-and-bound search algorithm. The method is based on the monotonicity assumption that adding new features is not going to decrease the performance of the new subset. An evaluation (or fitness) function, generally a separability index, has to be used to assess the subsets.

The fundamental idea behind the algorithm is that if a subset of size greater than  $m$  has a performance value lower than an initially set threshold, all sub-subsets of this subset are eliminated as they will have values lower than that of the main subset. Thus, many subsets can be discarded without evaluating their fitness. This is the main advantage of the technique. The threshold value used in the process is either found by one of the simple search techniques, or set by the algorithm each time the highest performance is found. If the solution space is thought of as a tree, branch-and-bound prunes the tree step by step based on the fitness values.

Due to the superiority of the algorithm, branch-and-bound algorithm can also be applied to search clusters and nearest neighbours. Details of the algorithm and some of its application areas, such as restricted least squares, maximum likelihood paired comparison ranking, and selection variables in regression, can be found in Hand (1981).

It is worth pointing out that in many practical pattern classification scenarios, the monotonicity assumption is not satisfied. For example, addition of irrelevant information may significantly worsen the generalisation accuracy of a decision

tree classifier (Yang and Honavar, 1998). Moreover, feature subset selection techniques that rely on the monotonicity of the performance criterion, although they appear to work reasonably well with linear classifiers, can exhibit poor performance with non-linear classifiers, such as neural networks (Ripley, 1996). Another disadvantage of the algorithm, which should be noted, is that the branch-and-bound search algorithm is slower than sequential search techniques.

### 5.7.3 Genetic Algorithms

The development of the Genetic Algorithm (GA) was inspired by hypothetical mechanism of natural selection where the fittest individuals at one generation are more likely to survive and produce the new generation. GAs are simulated in a computer environment to carry out the process of biological evolution. Moreover, GA approach, as an adaptive search technique, is utilised to find global maxima or minima depending on the nature of the problem under investigation. They are used to search for optimum solutions when the evaluation of all possible solutions is too costly in terms of computing time. They are increasingly being used in many fields, due to their unique advantages over random and local search methods. Particularly, combining genetic algorithms with artificial neural networks is one of the most popular research agenda for recent studies. GAs have been applied to a wide variety of problems inherent in the ANN approach, including the determination of optimum initial learning parameters, initial weights, network structure (Kuscu and Thornton, 1994; Bebis *et al.*, 1997), training ANNs (Man *et al.*, 1999) and analysis of the solution achieved by an ANN. Details of GA's involvement in ANNs can be found in Whitley (1995).

J.H. Holland developed the genetic algorithm as a programming technique in the mid-1960s, after he discovered that the recombination of groups of genes by means of mating was a critical part of evolution. Almost a decade later, he managed to develop a classifier system based on a genetic algorithm to perform particular actions every time its conditions are satisfied by some piece of information (Holland, 1992).

Genetic algorithms (GAs) differ from traditional search and optimisation methods in several ways. The four most significant differences highlighted by Chipperfield (1997) are:

- GAs search a population of points in parallel, not a single point,
- GAs use probabilistic transition rules, not deterministic ones,
- GAs work on an encoding of the parameter set rather than the parameter set itself, except where real-valued individuals are used,
- GAs do not require derivative information; only the objective function and corresponding fitness levels influence the directions of search.

The underlying idea behind GAs is that every solution can be represented by an individual called a chromosome, and each parameter can be thought of as a gene of that chromosome. Such a structure has a finite length and is symbolised through a special coding technique, such as binary, integer, and real-valued. The most commonly used representation for GAs is the binary form, in which 0 and 1 are used for exclusion and inclusion respectively.

The genetic algorithm process starts with the generation of an initial population, generally selected randomly and sized typically between 30 and 100, depending on the problem. It then evaluates all the members of the population by an objective (or evaluation) function so as to determine their fitness in order to determine the quality (or goodness) of the chromosome for the particular problem. Definition of the objective function is very important, since a poor definition of the objective function can mislead the search and, consequently, affect the resulting solution. From the set of fitness values a subset of the highest performing chromosomes is selected as “parents” by a selection procedure, the most commonly used of which is the roulette-wheel selection method. The genes of the parents are exchanged and recombined in a mating pool to form offspring for the next generation. It is expected that new chromosomes not only reflect the superior characteristics of their parents but also are improved versions of their parents. While the “better” chromosomes are more likely to give improved performance, they also have a higher chance to survive in the next generation. Figure 5.25 shows the process of a simple genetic algorithm adapted to this study.

A simple genetic algorithm consists of three operators, namely reproduction, crossover and mutation, to produce new generations (or populations). Reproduction is the process of copying the bits (genes) in the chromosomes. Depending on the fitness value of the chromosome, it is copied a number of times. The better the fitness value, the more likely is a string to reproduce and contribute one or more offspring to the next generation. When a chromosome is chosen for reproduction, a copy (or copies, depending on how many times the chromosome has been chosen for reproduction) is entered into a mating pool for further genetic operator action (Clark and Canas, 1995).

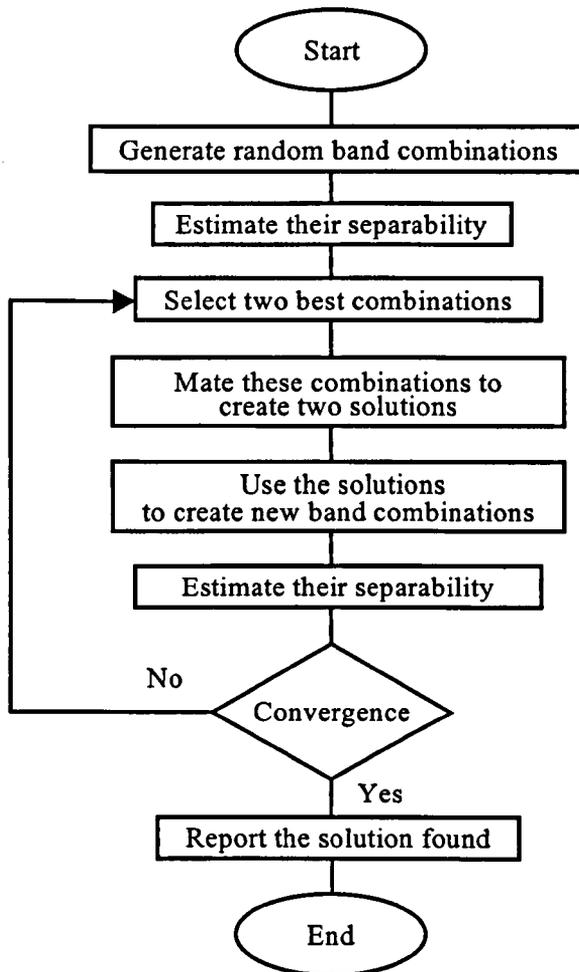


Figure 5.25 Process of Genetic Algorithms.

The crossover process, also known as recombination, is the basic operator for producing new chromosomes in genetic algorithms. Crossover generates new

individuals that carry the characteristics of both parents. This is done in two stages. Firstly, chromosome couples are randomly chosen from the mating pool, and secondly crossover points are selected randomly. Thus, genetic information is exchanged between crossover points. There are several crossover types, such as single-point, multi-point and uniform crossovers. The simplest type of crossover is the single-point crossover that swaps the bits (genes) between the parents around a randomly selected point (Figure 5.26).

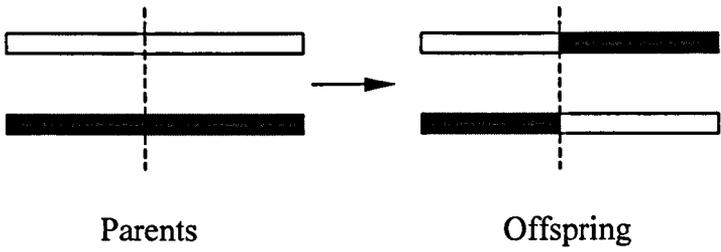


Figure 5.26 Example of single-point crossover.

In the case of multi-point crossover, several bits of genetic information are exchanged between parents (Figure 5.27).

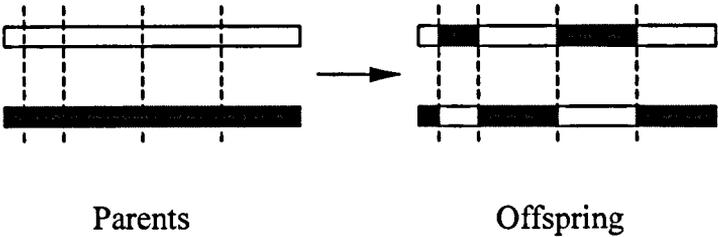


Figure 5.27 Example of multi-point crossover.

The third operator is mutation, which is performed after crossover. It is applied to each offspring individually and randomly changes the selected bits with a low probability, typically less than 0.1. The effect of mutation in binary GA is simply to change 0 to 1 and 1 to 0 (Figure 5.28). The role of mutation is to prevent the search process carried out by GA getting stuck into a local minimum (or maximum).

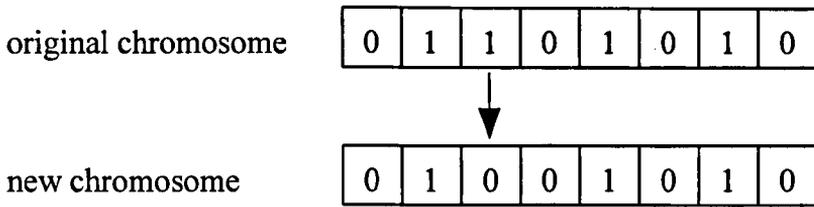


Figure 5.28 Bit mutation on the third bit.

The choice of crossover and mutation rates can be difficult, depending on the nature of the problem under analysis and the evaluation function used. Some recommendations on the selection of the parameters in practical applications of genetic algorithms are made by researchers. According to Man *et al.* (1999):

For large population size (100)

Crossover rate : 0.6

Mutation rate : 0.001

For small population size (30)

Crossover rate : 0.9

Mutation rate : 0.01

Since genetic algorithms are stochastic search techniques, it is very difficult to define termination criteria. Therefore, the GA process is usually run a pre-specified number of times and then the best solution is tested. If the result is not satisfactory, then either the process is restarted, or a new process is initiated with different parameters.

In the implementation of the genetic algorithm in this study, all features were represented by chromosomes, the length of which corresponds to the number of features available. As noted earlier, a minimum of eight features is required to achieve a satisfactory level of classification accuracy. Therefore, as an initial population a number of eight-band combinations were generated randomly. New combinations were then produced, employing mutation (0.01) and crossover (0.9) parameters. In the production of new feature combinations, combinations including more than eight features were penalised. The fitness of each solution, measured by separability, was computed directly from the formulae given above for separability measures.

## 5.8 Using Artificial Neural Networks for Feature Selection

Parallel development to the other application areas, artificial neural networks (ANNs) have recently been applied to the problem of feature subset selection, and have been found to be effective in locating optimal or near optimal solutions. The success of ANNs results from the nonlinear nature of the technique. Several studies (Mao *et al.*, 1994; Messer and Kittler, 1997; Leray and Gallinari, 1998) have employed neural networks in feature selection through input node pruning.

Mao *et al.* (1994) proposed a node saliency measure to remove insalient input and hidden nodes in the network. The saliency of a node in the network is defined as the amount of increase in the cost if this node is removed from the network. In other words, the effect of removing each input node is estimated and the least effective nodes (insalient nodes) in terms of the accuracy of the results are removed. After removing each node, the network is retrained for a small number of epochs. This process is repeated until the test set error rate starts to increase relatively faster. Thus, a parsimonious network (with a small number of parameters) is created. This pruning technique is similar to the optimum brain damage (OBD) and the optimum brain surgeon (OBS) algorithms, which are the most popular inter-connection pruning techniques (Kavzoglu and Mather, 1999). Whereas OBD and OBS remove the interconnections between nodes in the network, Mao *et al.*'s (1994) technique removes the nodes in the network, like a skeletonization pruning technique. Instead of first order approximation as in the skeletonization algorithm, the second-order information of the cost function is used. They concluded by underlining the fact that an advantage of the node-pruning procedure over classical feature selection methods is that the node-pruning procedure can simultaneously 'optimise' both the feature set and the classifier, while classical feature selection methods select the 'best' subset of features with respect to a fixed classifier.

Messer and Kittler (1997) compared a statistical feature selection technique to a neural network method. The statistical feature selection method that they use is the sequential floating forward selection algorithm (SFFS), developed by Pudil *et al.* (1994). On the other hand, a new method of analysing the network weight

values for feature selection problem is proposed. The neural network method involves several stages. First of all, the neural network is trained using the complete feature set on the specified query. The network weight values are then analysed in an attempt to find the most important features. The importance of each input node is assessed by estimating the sum of the weights connecting to that input node to the hidden layer. The sums of the input nodes are then ranked, and the lowest ones are eliminated, on the assumption that a high value implies that the corresponding input feature is more important. Both methods used performed equally well in the study. This suggests that artificial neural networks are very good alternatives to conventional search algorithms. Leray and Gallinari (1998) note that weights in the network cannot be interpreted easily since neural networks capture nonlinear relationships between variables. Hence, more sophisticated techniques are required to interpret the relationships between the nodes and the weights in the network.

A comprehensive study by Leray and Gallinari (1998) assesses a large number of neural network feature selection algorithms, including saliency based pruning (SBP), automatic relevance determination (ARD), optimal cell damage (OCD), and early cell damage (ECD). The neural network feature selection methods are described and discussed in detail, and comparative performances of different feature selection methods are presented for two problems using two different synthetically produced datasets. The first problem is a three-class waveform classification problem with 19 noisy dependent features, and the second problem is a two-class problem in a 20 dimensional space, in which the classes are in multivariate Gaussian distribution. Also, in this study, the neural network based feature selection techniques are categorised into three groups:

- Zero order methods which use only the network parameter values,
- First order methods which use the first derivatives of network parameters,
- Second order methods which use second derivatives of network parameters.

Whilst zero order techniques that involve simple interpretation ideas, are the simplest techniques, second order techniques that employ advanced and complicated procedures, are the most sophisticated methods.

In two recent studies carried out by Zongker and Jain (1996) and Jain and Zongker (1997), a large number of search algorithms (including variants of the forward selection and the backward elimination techniques, together with the branch-and-bound search algorithm and genetic algorithms) were compared using both synthetic and satellite image data. Reliability of feature selection methods when only small amounts of training data are available is also investigated in that study. It is found that there is a direct relationship between the number of training patterns per class and the average quality of the feature subset selection. It was concluded that the sequential forward floating selection (SFFS), proposed by Pudil *et al.* (1994), was the best search algorithm of those tested. They also noted that feature selection cannot only eliminate a large number of redundant features, but also avoid the “curse of dimensionality” (Bishop, 1995).

## **5.9 Using Pure Pixels for Delineation of Land-Cover Classes in Test Site 2**

The performance of an artificial neural network classifier is tested for the case where only pure pixels are involved. To achieve this aim, a new field boundary map (Figure 5.29) is produced through on-screen digitising of pixels in field centres on a SPOT HRV image. The class labels were given for each digitised polygon based on the information provided. For the training pattern set, 300 pixels per class and for the test dataset 260 pixels per class were randomly selected, giving a total of 2,100 training pixels and 1,820 test pixels.

The most effective eight bands in discriminating seven land cover classes were searched using seven separability measures (divergence, transformed divergence, Bhattacharyya distance, Jeffries-Matusita distance, Wilks'  $\Lambda$ , Hotelling's  $T^2$  and Mahalanobis distance classifier). As the results are very similar to each other in terms of the accuracy produced, only the result of ANN classifier for the solution found by the divergence separability index is given as Table 5.38.

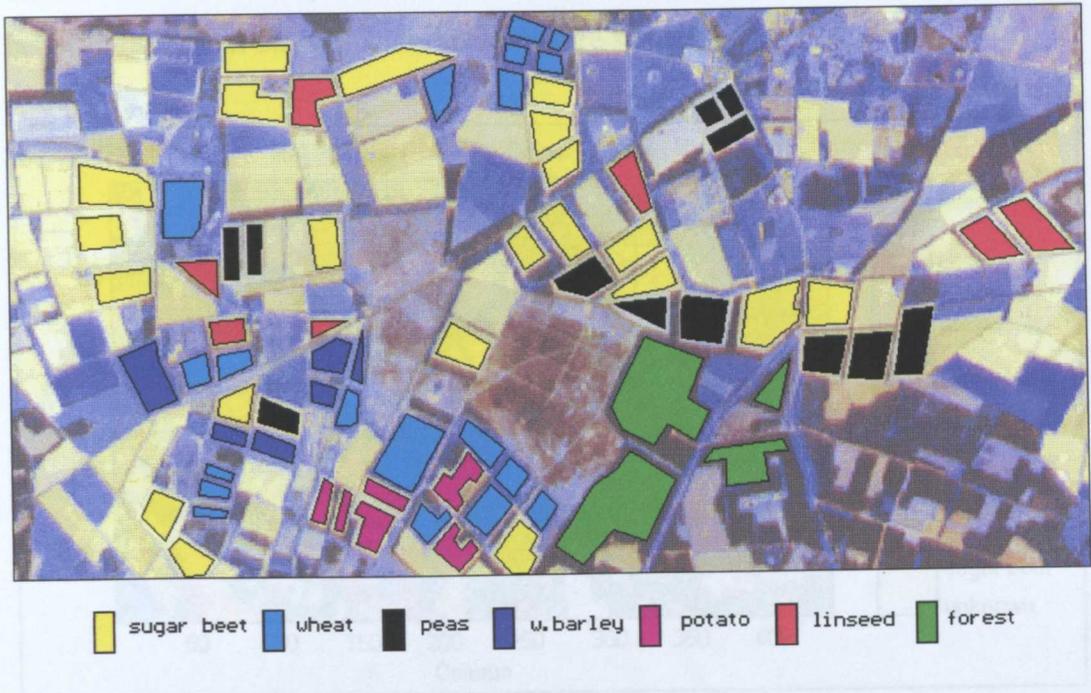


Figure 5.29 Ground control data including 75 fields for Thetford site.

Table 5.38 ANN classification results of the best band combination (11-9-17-16-13-20-19-23) found by divergence using sequential forward selection technique.

Iteration	Networks					
	8-10-7		8-15-7		8-20-7	
	Overall	Kappa	Overall	Kappa	Overall	Kappa
2500	98.96	98.78	99.23	99.10	99.01	98.85
5000	99.29	99.17	99.29	99.17	99.18	99.04
7500	99.01	98.85	99.23	99.10	99.29	99.17
10000	99.23	99.10	99.23	99.10	99.40	99.30
12500	99.12	98.98	99.29	99.17	99.34	99.23
15000	99.18	99.04	99.23	99.10	99.18	99.04

D: 934.39      TD: 2000      B: 40.52      JM: 1413.17

The solution found by divergence was also applied to whole image so as to observe and confirm the effect of such high accuracy on the image. The result of ANN classification for the full image is given in Figure 5.30. The resulting image reflects high accurate results found in that almost all the fields, especially the

forested area, are clearly defined due to introducing pure pixels to the network for better definition of class boundaries in feature space.

(confidence) in defining the fields containing the land cover types that are employed in classification, it would

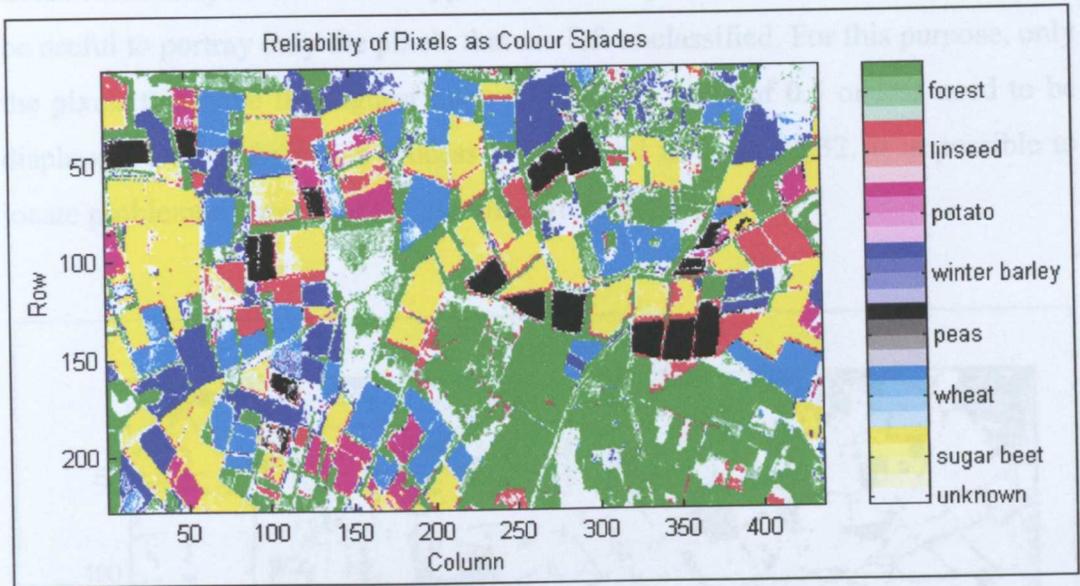


Figure 5.30 Result of ANN classification of the whole image using the network trained for the best divergence solution.

Also, output activation levels produced by the classification process were mapped using grey colour levels to acknowledge high accurate results derived from accuracy assessment. The result of this process is shown in Figure 5.31.

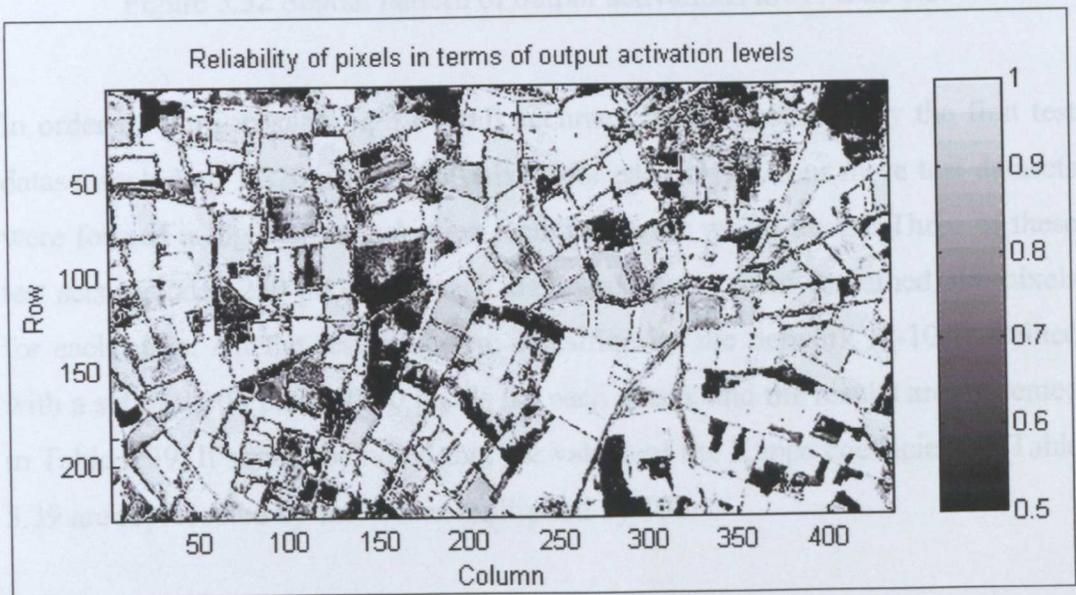


Figure 5.31 Spatial pattern of output activations presented in grey scale.

Although the figure clearly expresses high reliability (confidence) in defining the fields containing the land cover types that are employed in classification, it would be useful to portray only the pixels that are left unclassified. For this purpose, only the pixels that have the highest output activation value of 0.5 or less need to be displayed. Output from this process is presented in Figure 5.32. It is possible to locate problematic areas for the classifier from this figure.

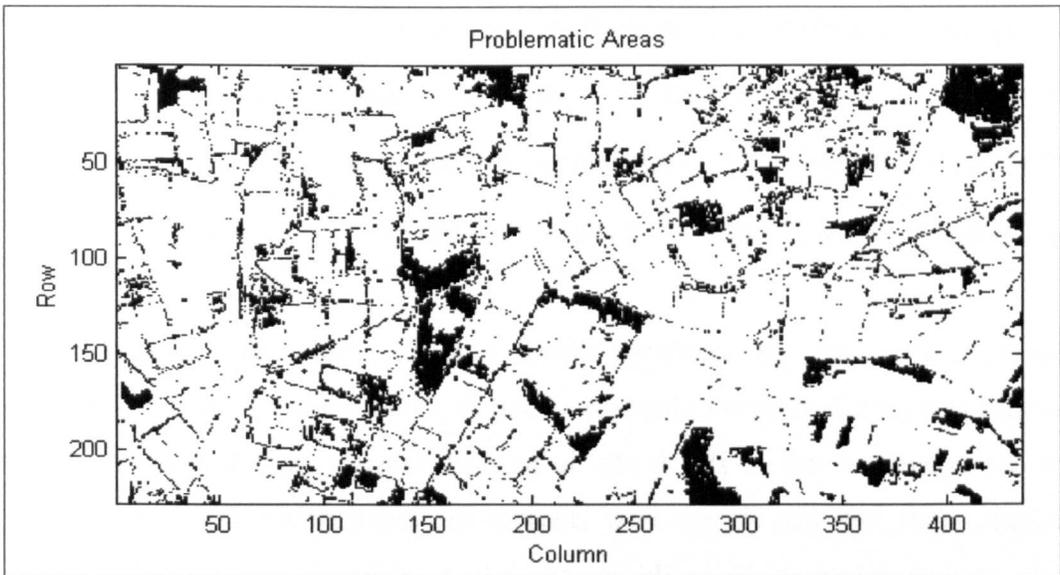


Figure 5.32 Spatial pattern of output activations lower than 0.5.

In order to confirm such high overall accuracy results produced by the first test dataset including 1,820 pixels (260 pixels for each class), four more test datasets were formed using random selection technique over whole image. Three of these test sets included 250 pixels for each class and one of them contained 300 pixels for each class. All the test sets were classified by the network (8-10-7) trained with a set of 2,100 pixels (300 pixels for each class), and the results are presented in Table 5.39. It should be noted that the values of the Kappa coefficient in Table 5.39 are represented by the values multiplied by 100.

Table 5.39 ANN classification of five test datasets using 8-10-7 network structure.

Iteration	Test 1	Test 2	Test 3	Test 4	Test 5
	Overall (Kappa)				
2500	98.96 (98.78)	99.14 (99.00)	98.91 (98.74)	99.31 (99.20)	98.62 (98.39)
5000	99.29 (99.17)	99.37 (99.27)	99.09 (98.94)	99.43 (99.33)	98.90 (98.72)
7500	99.01 (98.85)	99.26 (99.13)	98.97 (98.80)	98.97 (98.80)	98.81 (98.61)
10000	99.23 (99.10)	99.43 (99.33)	99.03 (98.87)	99.03 (98.87)	99.00 (98.84)
12500	99.12 (98.98)	99.20 (99.07)	99.09 (98.94)	98.91 (98.74)	99.05 (98.89)
15000	99.18 (99.04)	99.26 (99.13)	99.09 (98.94)	98.97 (98.80)	99.00 (98.84)

The reason for such high accurate results produced by the artificial neural networks may be that the number of pixels available for ground data was limited, that is, the actual class memberships of all the pixels in the test image are not known. Therefore, training and test datasets were mostly selected from adjacent pixels, which could certainly make the problem of determining the class memberships of such pixels easy for the classifier. However, the positive effect of employing pure pixels in the training cannot be ignored when it is considered that almost the same amount of data were available in the application where mixed pixels are involved.

Another confirmation was made by displaying individual pixels employed in training and testing the network to ensure that randomly selected training and test datasets do not include the same pixels, which could result in high performance. While the pixels used in training are given in Figure 5.33, the pixels used in testing (first test set) are shown in Figure 5.34. Also, all the pixels used for training and testing are displayed in Figure 5.35 to easily observe the difference.

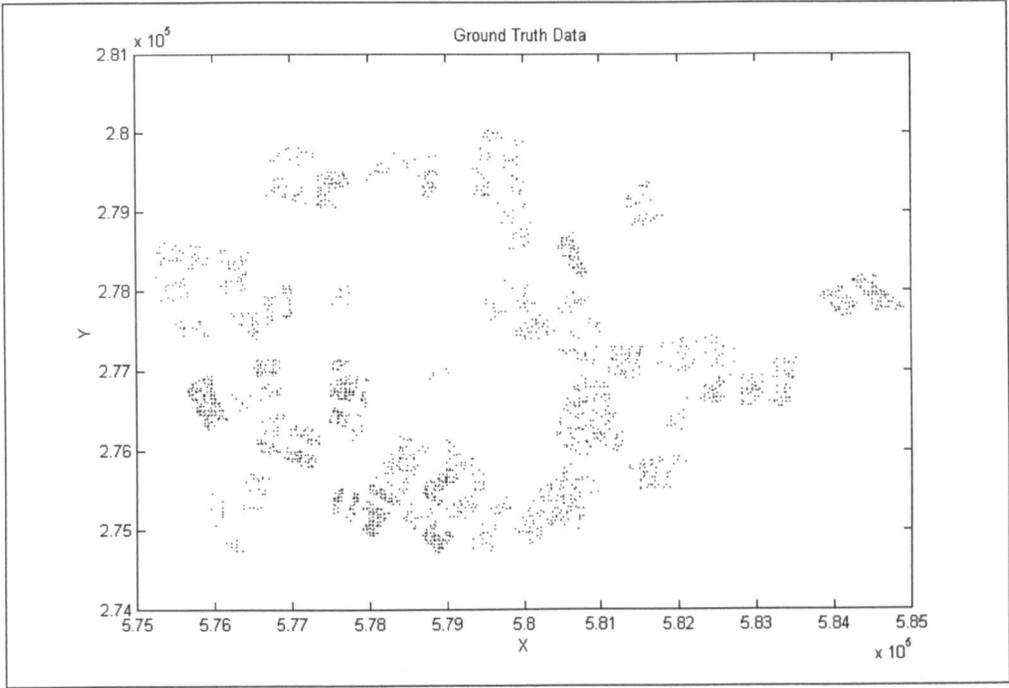


Figure 5.33 Training pixels used (2,100 pixels in total).

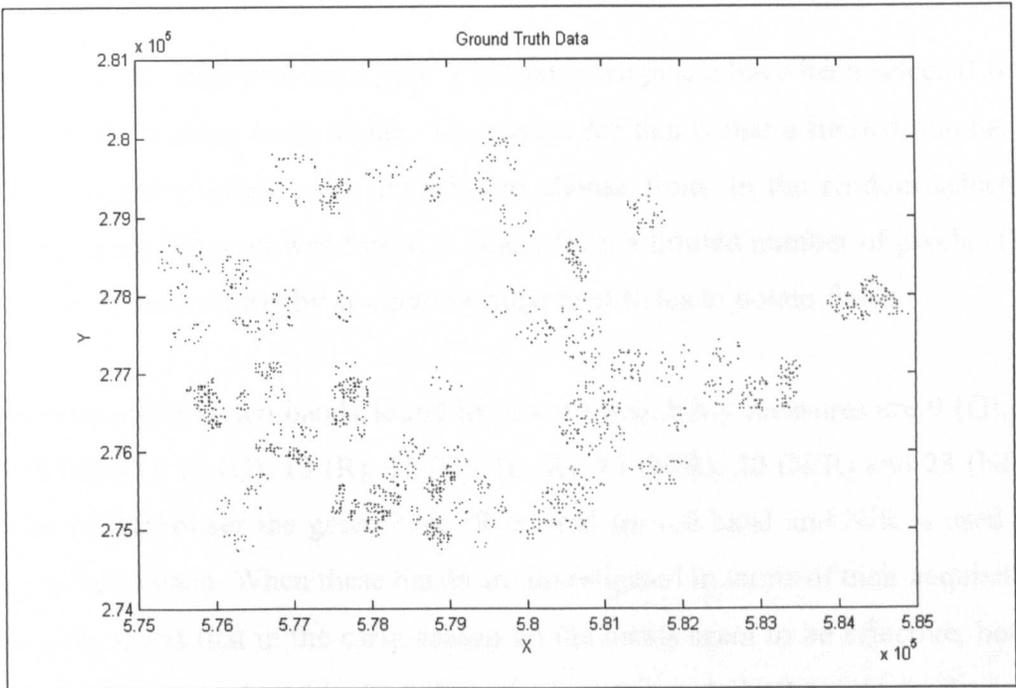


Figure 5.34 Testing pixels used (1,820 pixels in total).

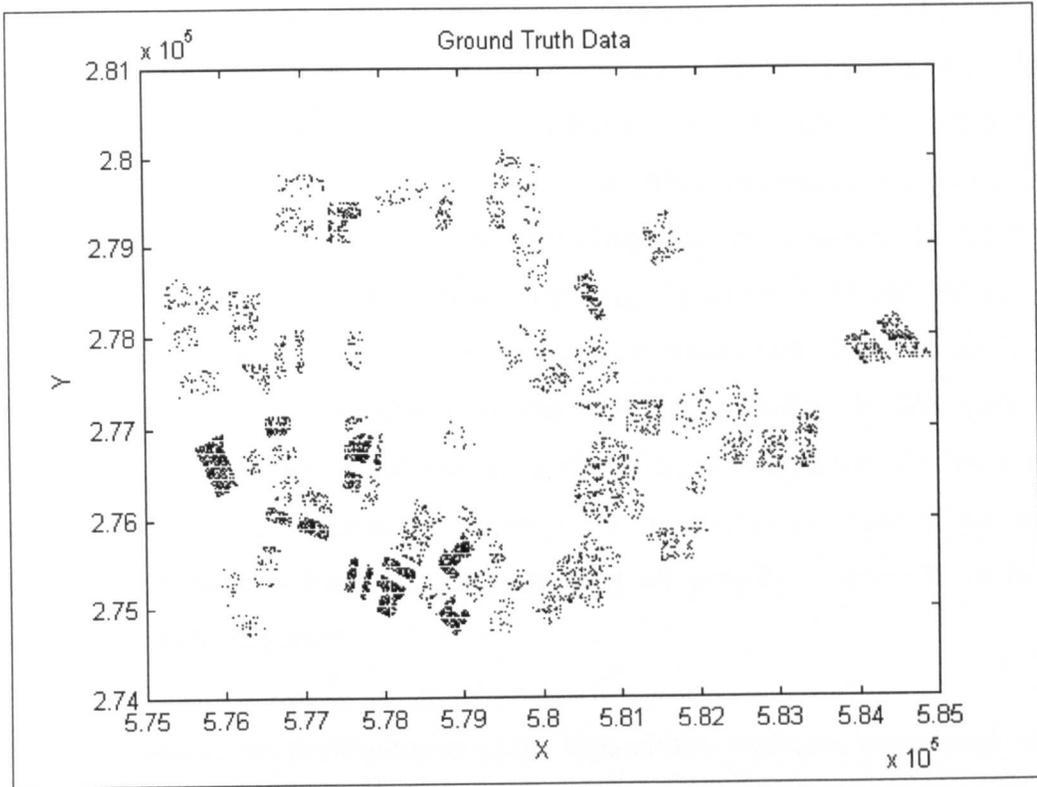


Figure 5.35 All the pixels used (training and testing pixels).

It can be easily observed by Figure 5.35 that more pixels have been selected from some fields, making them darker. The reason for this is that a limited number of fields, therefore pixels, were available to choose from. In the random selection procedure the program was forced to select from a limited number of pixels. This case can be seen clearly by comparing sugar beet fields to potato fields.

The most effective ten bands found by seven separability measures are 9 (G), 10 (R), 11 (NIR), 12 (G), 13 (R), 15 (G), 16 (R), 17 (NIR), 20 (NIR) and 23 (NIR), where G symbolises the green band, R is used for red band and NIR is used for near infrared band. When these bands are investigated in terms of their acquisition date, it is found that in the early season all the bands seem to be effective, but in the growing season between 28 June and 14 August only NIR bands are found to be effective in discriminating the land cover classes.

## 5.10 Conclusions

This chapter reviews the major feature selection methods that are used to search for the optimum subset in cases where many input bands are available. The methods are discussed under two main categories; namely, filters and wrappers. For the class separability indices using the filter approach, including the divergence, the transformed divergence, the Bhattacharyya distance, the Jeffries-Matusita distance and statistical tests, including Hotelling's  $T^2$  and Wilks'  $\Lambda$  criterion, are considered and their underlying theoretical bases have been discussed in detail. As an example of the wrapper approach, the Mahalanobis distance classifier, a supervised statistical classifier, is considered. On the other hand, major search techniques (or engines) that are used to generate subsets and select the optimum subset without evaluating all possible subsets have been comprehensively discussed.

In order to assess the performances of the separability measures considered, two search techniques, sequential forward selection (SFS) and genetic algorithm (GA), are employed. Solutions found by the search techniques using separability measures as fitness measures were used to generate training and test datasets that are later used to train and test three network structures (8-10-7, 8-15-7 and 8-20-7). In this study, two datasets have been used to make objective judgements about the performances of the methods used. Trained networks have been applied to classify the test images and presented in a special way in which pixels are represented by colour tones considering the highest output activations (class membership rates). These activations are also represented as grey scale values that are mapped to form an image of reliability. This representation provides accuracy assessment for each pixel in the image. Another research objective to be accomplished is determining the effect of the number of iterations on the learning or generalisation capabilities of the network. For this aim trained networks have been saved after every 2,500 iterations and the accuracy assessment has been carried out on each trained network.

Before drawing conclusions from the vast amount of results given earlier, it would be appropriate to present a summary that could help to ease the interpretation. For

this purpose, Table 5.40 and Table 5.41 are generated for the first and second test datasets, respectively. Optimum solutions found by sequential forward selection and genetic algorithm procedures, and their values in terms of the corresponding separability measure are given in both tables under the 'Value' heading.

Table 5.40 Solutions found by search techniques using nine separability measures for the first dataset. The column header 'Subset Solution' shows the spectral bands selected (1-24), as shown in Table 5.1.

Separability Measure	Sequential Forward Selection		Genetic Algorithm	
	Subset Solution	Value	Subset Solution	Value
Divergence		391.265		478.471
Tran. Div.		1999.945		1999.961
Bhatt.		15.346		17.822
J-M		1413.035		1413.481
Wilks' $\Lambda$		9.7E-05		6.9E-05
Hotel. $T^2$		20491.29		20240.08
MDC(A)		0.9131		0.9247
MDC(O)		0.9170		0.9215
MDC(Q)		9.855		9.778

Table 5.41 Solutions found by search techniques using eight separability measures for the second dataset. The column header 'Subset Solution' shows the spectral bands selected (1-23), as shown in Table 5.3.

Separability Measure	Sequential Forward Selection		Genetic Algorithm	
	Subset Solution	Value	Subset Solution	Value
Divergence		130.641		140.678
Tran. Div.		1999.475		1999.519
Bhatt.		5.123		5.322
J-M		1399.968		1400.412
Wilks' $\Lambda$		3.0E-03		3.0E-03
Hotel. $T^2$		4497.84		4546.78
MDC(A,O)		0.9091		0.9097
MDC(Q)		9.9548		9.9689

These tables are particularly useful to compare the robustness of the SFS and GA. It is also possible to derive the most effective bands from the subset band combinations. It should be noted that lower values of Wilks'  $\Lambda$  measure indicate better separability. Abbreviations of MDC(A), MDC(O) and MDC(Q) are used to represent the Mahalanobis distance classifier based on average accuracy, overall accuracy and quality measure in the search of best band combination. Performances of all separability measures for both search methods in the case of first and second datasets with 8-15-7 network configuration are shown in Figures 5.36 and 5.37, respectively. While these figures help to assess the relative effectiveness of the measures, they reveal some important characteristics of the behaviour of the neural networks.

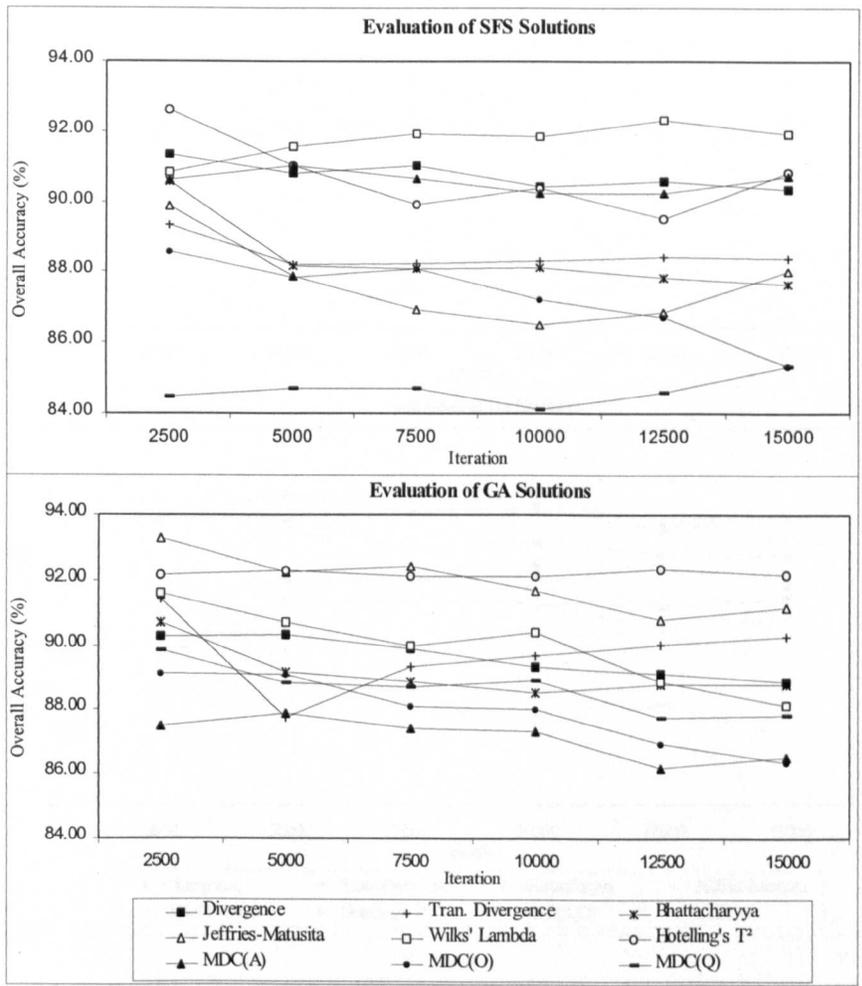


Figure 5.36 ANN evaluation of solutions found by SFS and GA procedures for the first dataset using network structure of 8-15-7.

The results presented in Figure 5.36 show that solutions attained using the MDC method, in general, were inferior to others in terms of the classification accuracy produced. Of the measures used for the MDC method, the quality measure performed the worst. Hotelling's  $T^2$  measure gave the most consistent and accurate results. It should be pointed out that Wilks'  $\Lambda$  criterion also performed well (around 92% overall accuracy and over 90% Kappa coefficient for the network structure of 8-15-7). Although the Jeffries-Matusita distance measure seemed to be effective in distinguishing the classes from each other for the solution found by GA, it did not perform well for the solution reached by SFS.

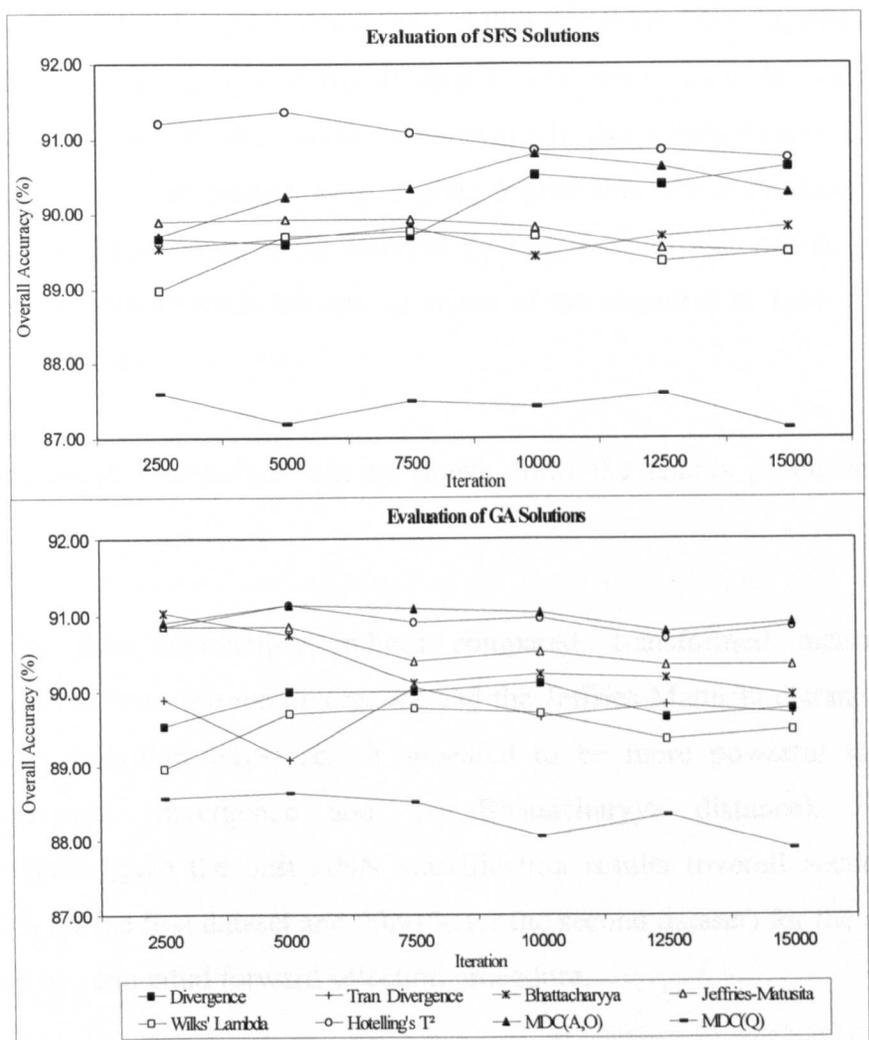


Figure 5.37 ANN evaluation of solutions found by SFS and GA procedures for the second dataset using network structure of 8-15-7. Note that lines for MDC(A,O) represent the results of both MDC(A) and MDC (O) as equal number of patterns were used for every class.

Figure 5.37 showing the results for the second dataset also reveals some important characteristics of the separability measures used in this study. Firstly, Hotelling's  $T^2$  and the Mahalanobis distance classifier based on overall accuracy criterion appeared to perform well in solutions determined by both SFS and GA procedures. Secondly, the performance of MDC based on the quality measure was inferior for the solutions found by both search techniques (around 87.5% for SFS and 88.5% for GA-found solutions). Finally, unlike the high-level of accuracy achieved by Wilks'  $\Lambda$  criterion for the first dataset, the accuracies produced for the second dataset were lower than others except for the MDC method based on the quality measure. One important finding is that there was not any considerable change in the network performances after 5,000 iterations. This suggests that the size of the network was appropriate to learn the characteristics of the training data at 5,000 iterations. Another point to be made is that classification accuracies produced for the first dataset were slightly higher than those produced for the second dataset. This could be the result of the degree of difficulty of the problem, which can be noticed when the critical values of the measures in Tables 5.40 and 5.41 are compared.

Some important conclusions can be drawn from the results produced in this chapter:

- Of the four separability indices compared, transformed measures, or derivatives (transformed divergence and the Jeffries-Matusita distance) in the genetic algorithm based search appeared to be more powerful than their counterparts (divergence and the Bhattacharyya distance). However, divergence gave the best ANN classification results (overall accuracies of 92.7% for the first dataset and 90.91% for the second dataset) for the solutions found by sequential forward selection procedure.
- The genetic algorithm usually finds a better solution than the sequential forward selection method in terms of the critical value of the measure considered. However, these solutions do not always guarantee better classification results. This behaviour could be attributed to the fact that GA approach searches for the best bands by considering only the average

separability. However, the SFS algorithm applied in this research locates the first four bands on the basis of average separability and selects the next four bands in such a way that each additional feature shows the greatest improvement in the poorest interclass separability. It may be the reason that these poor inter-class separabilities reduce the accuracy of resulting classification.

- The methodology proposed and used to search subsets for separability indices in SFS worked well in some cases. Comparable results are thus produced by SFS and GA methods. This can be acknowledged by comparing results of the subset solutions produced by SFS and GA. For example, the GA based on divergence found a solution with higher divergence (478.5) than the solution found by SFS (352.2) for the first site. However, the solution found by SFS produced better classification results. It can be concluded that higher separability does not guarantee better (more accurate) classification results.
- Employing the Mahalanobis distance classifier in the search process did not result in any improvement in the classification accuracy compared to the performances achieved by the solutions found by the separability indices. Of the measures used in the MDC method, the overall accuracy criterion gave the best results, and quality estimation gave the worst results. The failure of the quality measure may be resulting from the fact that search methods try to find closer pairwise accuracy instead of improvement in both overall and pairwise accuracy.
- It has been observed that there is no need to train the networks 15,000 times. For the first dataset, 2,500 iterations were generally found to be adequate for the networks used in this study and, for the second dataset, 5,000 (or 7,500) iterations were found to be adequate to perform. However, the number of iterations is related to the problem under consideration. It can be observed that overtrained networks give worse results.
- Of the three network structures considered, the 8-15-7 structure generally appeared to be the most appropriate one. In general, it is observed that small network structures generally gave better results.

- Although 16 bands out of 24 were eliminated for the first dataset, and 15 bands out of 23 were eliminated for the second dataset, eight-band subset solution has been found to be effective in identifying the land cover classes with around 90% overall accuracy.
- In the analysis of statistical separability measures, it is found that Hotelling's  $T^2$  measure is more effective than Wilks'  $\Lambda$  in terms of measuring the separability between classes. In fact, the best classification accuracies were generally produced by the use of Hotelling's  $T^2$ . The performance of the Wilks'  $\Lambda$  criterion, on the other hand, changes drastically depending on the characteristics of the dataset.
- Separability measure values for different datasets do not directly represent the accuracy that will be produced by an ANN-based classifier.
- Although SIR-C raw and filtered images were employed in the search processes, their bands were rarely selected. This shows that SIR-C radar data was not able to provide better separation for the selected land cover classes than SPOT HRV data. The main reason for this could be the acquisition date of the image, which indicates an early stage of crop development.
- It has been observed that setting the optimum rates for crossover and mutation in the GA search is of great importance to reach a solution subset. However, the rates that are recommended by Man *et al.* (1999) have been found to be appropriate for all cases considered in this study.

## CHAPTER VI

# ISSUES RELATED TO THE DESIGN AND USE OF ARTIFICIAL NEURAL NETWORKS

### 6.1 Introduction

The training of a neural network requires user interaction in order to define the network structure and set the learning parameters. These parameters are described earlier as the internal parameters, and it has been reported that they have considerable influence on network performance. Their importance and impact on the network's performance were briefly discussed earlier in section 3.9, "*Problems in the Use of Artificial Neural Networks*". This chapter aims to expand the discussion by providing detailed information about the issues concerning the design and use of neural networks, and reporting on the results of experiments to understand the nature of each parameter. A further objective of this chapter is to provide heuristics (or rules of thumb) and to compare their effectiveness by applying them to real-world problems.

Starting from the components of the network structure, all important factors related to neural network training are considered. It is common in practice that users design their networks using trial-and-error strategies, and employ pre-defined rates for the learning parameters. In fact, most software packages offer fixed rates for the learning parameters. It is well known that the optimum rates for

the parameters and the size of the network required are problem dependent, and should be determined individually for each dataset and network structure. A comprehensive examination is thus required in order to enable new users to apply neural network models confidently and successfully.

## **6.2 Number of Input Nodes**

The number of input layer nodes in neural networks generally corresponds to the number of independent variables. In the case of remote sensing applications, each node represents a specific feature, such as a spectral band or a specific type of information derived from image bands, such as context and texture. The size of the input layer is also defined by the encoding technique used. For example, if the binary-encoding technique is employed, 32 input nodes are required for four input features.

There is evidence that introducing the training data in different orders results in different classification performance. This is particularly valid for cases where data samples are grouped according to the class type. There is a danger that just after the network starts learning the characteristics of the first group, it may 'forget' these characteristics when learning the details of the second group. This goes on in the same way for other classes. The network is thus biased towards the last grouping. An effective solution to this problem is to randomly shuffle the order in which training samples are considered by the network. In the training processes employed in this study, a shuffling facility was always utilised.

Since the number of spectral bands available for a particular location has increased with the launch of satellites carrying instruments that provide more spectral bands in higher spatial resolution, the representation of multispectral, multitemporal and multisensor image data in artificial neural networks has become a major issue. Selection of the most relevant image bands depending on the nature of the problem or reducing the dimensions of the input data has become unavoidable. The number of input layer nodes can be reduced using

dimensionality reduction techniques, feature selection methods, or node pruning methods.

Dimensionality reduction techniques are described in Chapter IV, and the use of feature selection techniques is discussed and investigated for two datasets in Chapter V. Although the theories underlying node pruning methods are given in Chapter III, their use for feature selection purposes has not been investigated. Two of the most popular node pruning methods, known as the Skeletonization and the Noncontributing Units methods, have been applied to a dataset derived from the image concerning the area near the town of Littleport, the characteristics of which are given in Chapter V. For the test site there are 24 spectral bands available from two Landsat TM and four SPOT HRV images. In this experiment, 4,000 pixels were randomly selected and used in training, and 3,000 randomly selected pixels were utilised to test the performances of the trained networks. Network weights were initialised randomly in the range  $[-0.3, 0.3]$ , and the learning rate was set to 0.2. A network structure of 24-25-7 was found to be adequate and trained 4,000 times reaching an MSE (Mean Square Error) of 0.07073. Once the training was completed, the network was saved, and both node pruning techniques were applied. In the pruning stage, the learning rate was set to 0.1 to avoid possible oscillations, and 150 iterations were used to retrain the networks after pruning in order to recover from the loss of the pruned unit.

The results of the pruning practices were analysed in two ways. Firstly, Mean Square Error (MSE) values were recorded and analysed to observe the effect of pruning in terms of the change in error, and, secondly, the pruned networks were used to compute classification accuracy using a contingency matrix. Changes in MSE and in the classification accuracies when the Noncontributing Units method was applied are shown in Figures 6.1 and 6.2, respectively. Note that values in the horizontal axis in Figure 6.2 show the eliminated bands at a particular pruning stage.

As can be noticed from Figure 6.1, no significant change was observed in the MSE values until number of bands was reduced from 24 to 13. This shows that, in this example, half of the input nodes can be eliminated without causing any major

problems in discriminating between classes. However, a sharp increase in MSE occurred when the number of bands was reduced further. Figure 6.2, on the other hand, suggests that number of inputs can be reduced to 9 with a slight reduction in the overall accuracy. As indicated by MSE change, a sharp fall occurred in the overall accuracy when the number of input bands was less than seven.

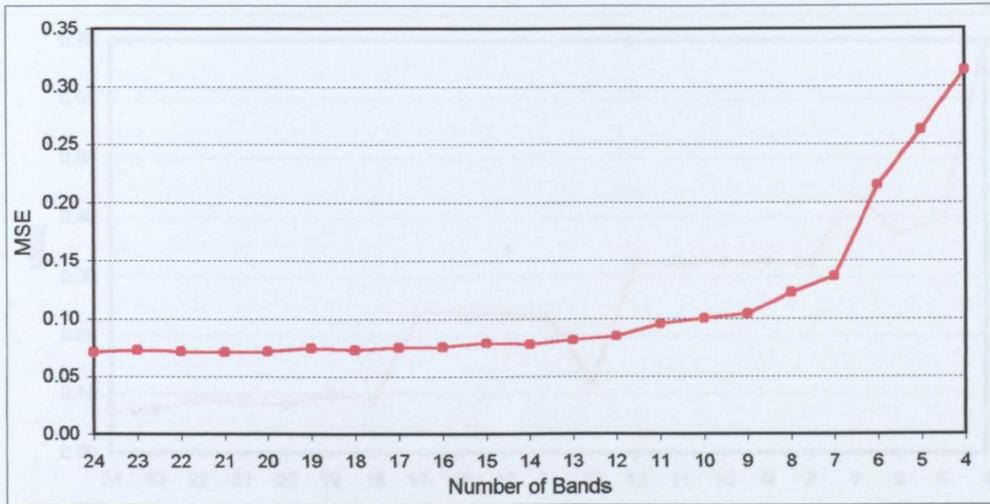


Figure 6.1 Changes in MSE using the Noncontributing Units pruning method.

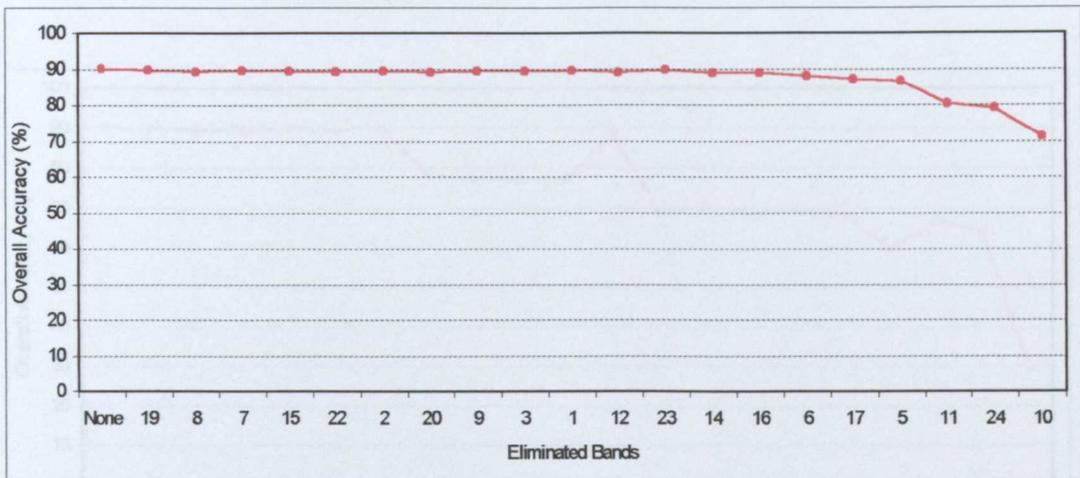


Figure 6.2 Overall accuracy versus bands eliminated by the Noncontributing Units method at each pruning step.

The Skeletonization pruning method was also applied to the same dataset using the same values of the parameters. The results of the process are presented in Figures 6.3 and 6.4. No significant change was observed in either MSE or overall accuracies when the first six bands were eliminated. However, when the

Skeletonization method eliminated band 7 to reduce the number of bands to 17, a drastic increase in MSE and a corresponding sharp decrease in overall accuracy were observed. It can be deduced that the method eliminated the wrong band at this stage, hence the accuracy decreased considerably. Overall, it has been noticed that sudden changes in MSE lead to abrupt changes in classification accuracies.

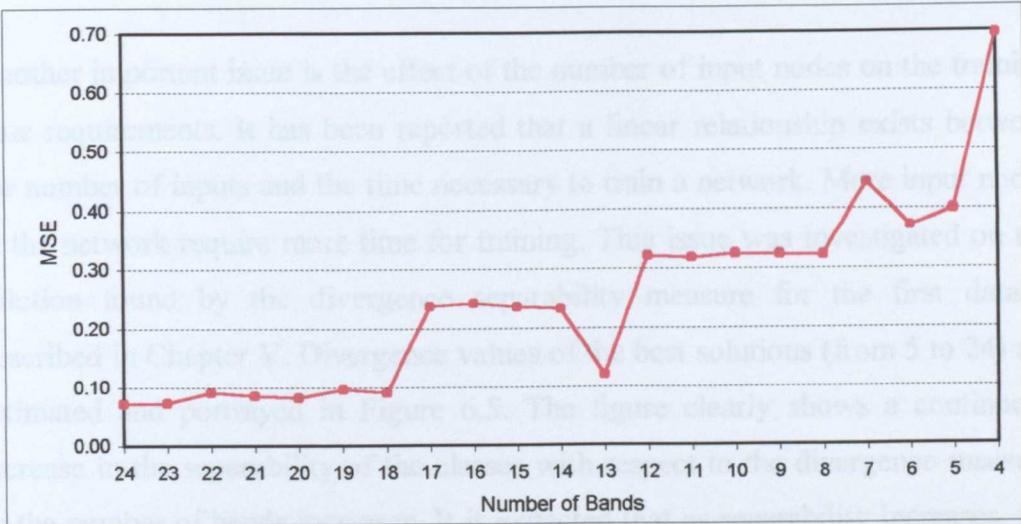


Figure 6.3 Changes in MSE using the Skeletonization node pruning method.

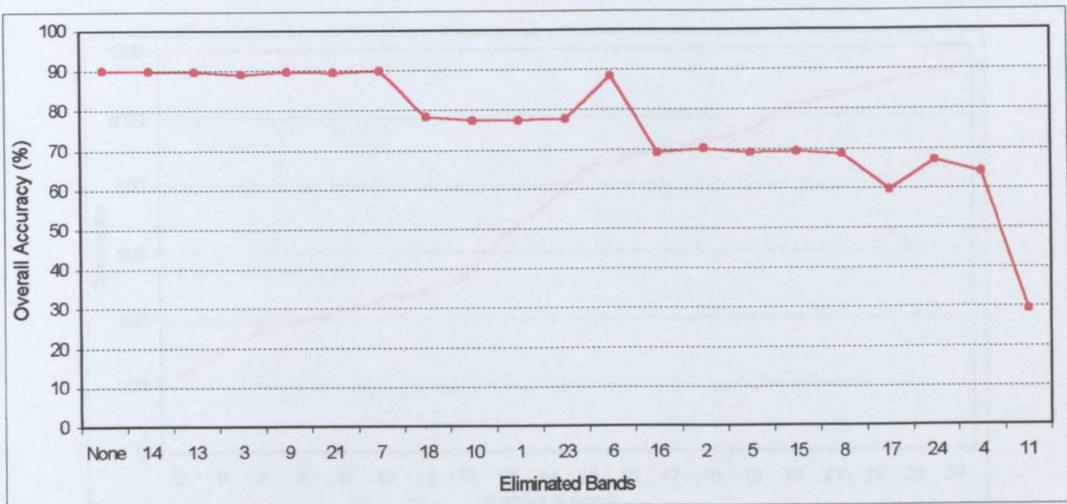


Figure 6.4 Overall accuracy versus bands eliminated by the Skeletonization method at each pruning step.

The results presented for the techniques clearly suggest that the Noncontributing Units node pruning technique performs better in terms of eliminating the least

effective bands. This can be noticed from Figures 6.1 and 6.2, in that the MSE values and overall accuracies vary smoothly. Whilst 16 input bands can be eliminated with confidence using the Noncontributing Units method, only six bands can be safely eliminated by the Skeletonization method. It is clear from the results that the Noncontributing Units pruning method can be effectively used for feature selection purposes.

Another important issue is the effect of the number of input nodes on the training time requirements. It has been reported that a linear relationship exists between the number of inputs and the time necessary to train a network. More input nodes in the network require more time for training. This issue was investigated on the solution found by the divergence separability measure for the first dataset described in Chapter V. Divergence values of the best solutions (from 5 to 24) are estimated and portrayed in Figure 6.5. The figure clearly shows a continuous increase in the separability of the classes with respect to the divergence measure as the number of bands increases. It is expected that as separability increases, the learning process will be easier and the corresponding classification accuracy will be higher.

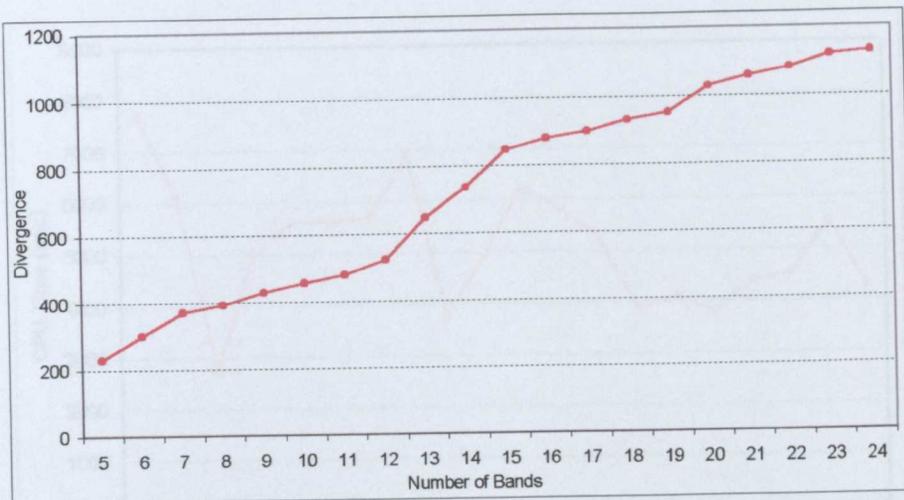


Figure 6.5 Relationship between divergence values and number of input bands.

In order to estimate the time required for each subset solution accurately, networks with 20 hidden and 7 output layer nodes were trained ten times on a system of the Sun Enterprise 450 Server configured with dual 400 MHz

UltraSPARC2 CPU processors with 256 Mb RAM. CPU times were based on the SNNS program (*batchman*) training the dataset 15,000 times. The result of this process is shown in Figure 6.6. The details of the operation are, however, presented in Table 6.1. In contrast to statements made in the literature on the trade-off between the network size and the time required to train the networks, Figure 6.6 does not suggest any linear relationship. However, it may indicate that using certain bands together can help the network recognise patterns quickly. It is likely that the time required for learning will increase when there is some confusion (or conflict) among the input information. For example, subsets of 7 and 13 input bands needed much less time than other combinations due to the low level of confusion among bands, which may be resulting from redundancy. However, 12 and 15 subset solutions required more time than the subsets having similar number of input bands due to a higher level of confusion. Another observation is that, although it might be expected that adding two inputs to an existing set of 5 inputs would increase training time (as there is more information to process), the CPU time actually reduced from 7675.74 sec to 2725.42 sec. This also indicates that adding two more spectral bands into the input layer facilitated learning by increasing the separability of the classes.

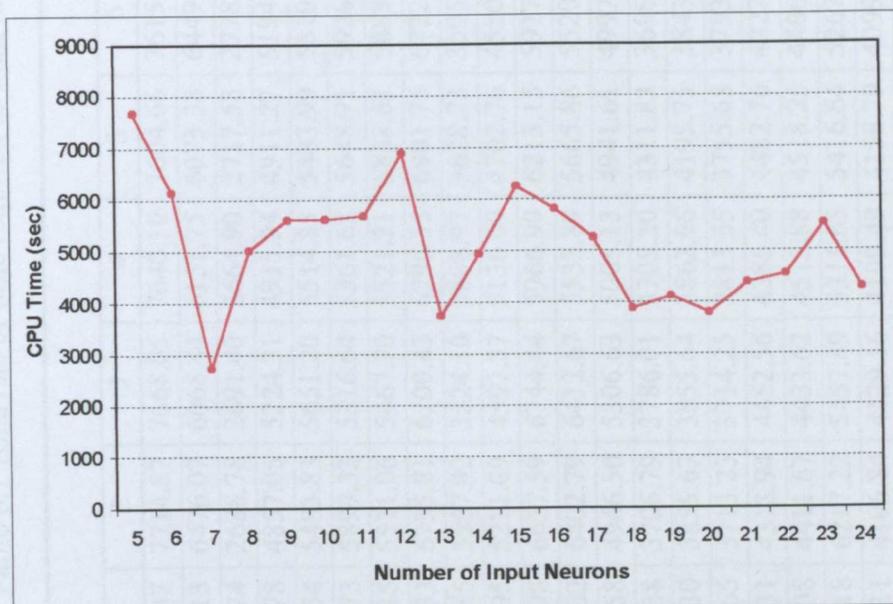


Figure 6.6 CPU time as a function of number of input nodes.

Table 6.1 Analysis of time required to train neural networks having number of inputs ranging from 5 to 24.

Number of Bands	CPU Times in Seconds														Mean	Std. Dev.
	1	2	3	4	5	6	7	8	9	10	Min	Max				
5	7440.47	7714.87	7668.05	7640.10	7604.66	7615.95	7644.62	7758.29	8066.44	7603.91	7440.47	8066.44	7675.74	160.70		
6	6031.13	6476.07	6068.44	6151.75	6073.38	6449.03	5986.21	5986.92	6207.05	5975.28	5975.28	6476.07	6140.53	185.20		
7	2740.74	2688.78	2691.40	2664.90	2717.53	2778.56	2664.33	2688.60	2956.44	2662.91	2662.91	2956.44	2725.42	89.22		
8	4942.78	4857.05	5224.31	4915.94	4911.27	5194.36	4897.41	4906.48	5387.96	4910.75	4857.05	5387.96	5014.83	183.27		
9	5595.54	5483.83	5651.20	5514.85	5483.99	5510.86	5696.02	5483.84	5894.96	5695.80	5483.83	5894.96	5601.09	134.85		
10	5783.73	5899.33	5316.64	5361.63	5648.91	5926.35	5332.12	5358.97	5709.03	5782.23	5316.64	5926.35	5611.89	245.61		
11	5519.15	5511.06	5467.70	5521.21	5838.61	5685.87	5699.53	5493.10	5885.69	6175.90	5467.70	6175.90	5679.78	229.35		
12	7103.33	6735.81	6700.43	6796.35	6901.75	6722.37	6981.61	6936.99	7285.12	6804.37	6700.43	7285.12	6896.81	187.33		
13	4090.85	3557.92	3524.10	3808.97	3628.73	3595.66	4185.60	3747.32	3710.73	3575.37	3524.10	4185.60	3742.53	227.87		
14	5198.95	5211.09	4597.37	5136.69	4702.76	4580.69	5177.94	5189.81	4910.17	4653.17	4580.69	5211.09	4935.86	275.69		
15	6014.98	6657.59	6744.44	5966.99	6213.15	5917.11	6213.22	6775.22	6012.46	5985.14	5917.11	6775.22	6250.03	343.49		
16	5490.02	6402.79	6412.67	5532.89	5665.86	5520.89	5594.93	6437.79	5543.08	5482.82	5482.82	6437.79	5808.37	423.84		
17	5601.58	4946.50	5306.63	5008.13	4941.61	4997.98	6073.19	5600.38	5028.31	5057.04	4941.61	6073.19	5256.14	383.33		
18	3726.38	3706.79	3786.01	3705.30	4311.83	3696.15	3970.69	4392.39	3680.27	3837.46	3680.27	4392.39	3881.33	263.67		
19	3946.30	3855.67	3855.64	3862.96	4195.75	3843.47	4085.17	4535.54	4353.65	4648.02	3843.47	4648.02	4118.22	302.85		
20	3710.65	3713.23	3714.25	3813.55	3755.68	3793.21	3948.68	3761.80	3695.25	3963.59	3695.25	3963.59	3786.99	96.99		
21	4313.31	4313.94	4452.56	4386.40	4482.76	4327.93	4534.61	4405.97	4252.15	4393.87	4252.15	4534.61	4386.35	86.99		
22	4766.08	4411.67	4433.62	4513.88	4518.25	4406.88	4586.47	4527.69	4416.75	4824.60	4406.88	4824.60	4540.59	147.67		
23	6153.18	6217.22	5457.49	5311.65	5416.64	5267.44	5440.39	5371.41	5278.87	5275.43	5267.44	6217.22	5518.97	358.26		
24	4222.11	4106.84	4129.16	4109.49	4158.79	4099.39	4307.93	4340.72	5133.28	4101.45	4099.39	5133.28	4270.92	315.54		

The last analysis in this section involves the investigation of the effect of the number of inputs on the classification accuracy produced by the trained networks. The networks trained for CPU time estimation were saved at every 5,000-iteration period. Trained network performances were evaluated using a test dataset including 2,204 patterns. The results are shown in Figure 6.7.



Figure 6.7 Change in overall accuracy with respect to the number of inputs and number of iterations used for training.

Several conclusions can be drawn from the results presented in Figure 6.7. First, the highest classification accuracy was achieved for the 23-band subset solution. It should be also noted that the subset solution including ten input bands also produced highly accurate results at 5,000 iterations. Second, considerable changes in the overall accuracy were noticed for different subset solutions, ranging from 89 percent to 93 percent. However, reducing the size of the input layer in the network did not result in a definite decrease in accuracy; on the contrary, similar results were obtained. Finally, in terms of the effect of the number of iterations on the performance, it is observed that 5,000 iterations generally produced the best results for the problem considered here. It can be easily seen from the figure that with the increased number of iterations slightly lower classification accuracies were produced.

The main conclusions reached in this section can be given as follows: relationship between the number of input nodes and the training time required is not linear, more input features do not necessarily produce more accurate results, and Noncontributing Units pruning method outperforms the Skeletonization method.

### **6.3 Number of Hidden Nodes**

A major task in designing a neural network is to determine the number of hidden layers and the number of nodes in those layers. In essence, the number of nodes in the hidden layers defines the complexity and power of the neural network model to be used to delineate underlying relationships and structures inherent in a dataset. The number of hidden layer nodes has a considerable effect on the classification accuracy and training time requirements. The level of classification accuracy that can be produced by a neural network is related to the generalisation capabilities of that network. Basically, the number of nodes in the hidden layer(s) should be large enough for the correct representation of the problem, but at the same time low enough to have adequate generalisation capabilities. While networks that are too small cannot identify the internal structure of the data (known as underfitting) and therefore produce lower classification accuracy results, networks that are too large are likely to become overspecific to the training data (known as overfitting). Such overspecificity also results in low classification accuracies and longer training time requirements. That is, such a network would perform well on the training data, but may fail to classify new data outside the range of the training data. For all these reasons, determination of the optimum number of hidden nodes has always been a serious concern to neural network users. This has posed a major difficulty and obstacle for new users, and thus undermines the popularity of artificial neural networks.

The question is not only to find the optimum number of hidden layer nodes but also to determine the optimum number of hidden layers. It has been reported by several researchers (Lippmann, 1987; Cybenko, 1989) that a single hidden layer should be usually sufficient for most problems, especially for classification tasks (Garson, 1998). However, some benefits arise from the use of a second hidden

layer, as discussed by Chester (1990) and Hand (1997) in terms of minimising the interaction between the neurons and allowing convex regions to be combined. In cases where the optimum number of hidden nodes on a single layer is large, two hidden layers with a smaller number of nodes on each layer could be more appropriate. Kanellopoulos and Wilkinson (1997) state that where there are 20 output classes (or more) it is advisable to use a second hidden layer. In such a case the second hidden layer should contain a number of nodes equal to two or three times the number of output classes.

As noted in Sarle (2000), the problem of determining the optimum number of hidden layer nodes is not an easy one since it depends in a complex way on:

- the numbers of input and output units,
- the number of training cases,
- the complexity of the function or classification to be learned,
- the amount of noise in the targets,
- the architecture,
- the type of hidden unit activation function,
- the training algorithm, and
- regularization.

While the number of inputs to the network defines the complexity of the problem, the number of output nodes determines the difficulty of separation of the classes in the feature space. Therefore, these two components of the network are vitally important and together they determine the optimum number of hidden layer nodes. Hence, most of the rules of thumb have been proposed using a function of numbers of input and output nodes.

Several strategies and heuristics have been suggested to estimate the optimum number of hidden layer nodes. However, none of these suggestions has been universally accepted or used. Note that the strategies utilised to build neural networks, namely pruning, constructive methods, and the hybrid techniques coupling both methods, are described in section 3.7 of Chapter III.

Most of the rules of thumb result from the experience of individuals using trial-and-error methodology. However, it should be mentioned that there are two rules of thumb that are exceptional since they are based on mathematical theories. The first, introduced by Hecht-Nielsen (1987), uses Kolmogorov's theorem, which states that any continuous function of  $n$  variables can be represented by the superposition of a set of  $2n+1$  univariate functions. From this theorem, he suggests that any function can be implemented in the single hidden layer neural network having  $2N_i + 1$  nodes in the single hidden layer, where  $N_i$  represents the number of input nodes. Secondly, Paola (1994) derived the formulae, shown in Table 6.2, by making the number of parameters necessary for neural networks equal to the parameters required by the maximum likelihood classifier. Whilst the parameters in neural networks are the network weights, those in the maximum likelihood classifier are the mean vectors and variance-covariance matrices for each class. Other heuristics used to determine the number of hidden layer nodes are listed in Table 6.2.

Table 6.2 Heuristics proposed by researchers to compute the optimum number of hidden layer nodes. See text for explanation.

Heuristic	Source	Optimum Nodes for the Datasets
$2N_i$ or $3N_i$	Kanellopoulos <i>et al.</i> (1997)	16 or 24
$3N_i$	Hush (1989)	24
$2N_i + 1$	Hecht-Nielsen (1987)	17
$2N_i/3$	Wang (1994b)	6
$(N_i + N_o)/2$	Ripley (1993)	8
$N_p / [r(N_i + N_o)]$	Garson (1998)	15–30
$\frac{2 + N_o \cdot N_i + \frac{1}{2} N_o (N_i^2 + N_i) - 3}{N_i + N_o}$	Paola (1994)	21

In Table 6.2, numbers of input and output layer nodes are represented by  $N_i$  and  $N_o$  respectively, and the number of training samples (or patterns) is represented by  $N_p$ . The symbol  $r$  used in Garson's (1998) formulation is a constant set by the noise level of the data. Typically,  $r$  is in the range from 5 to 10. Garson (1998) mistakenly states that  $r$  might be as high as 100 for very noisy data and as low as 2 for very clean data, whereas the reverse is the case.

The number of hidden layer nodes is also dependent upon the number of training samples available. Huang and Huang (1991) suggest that one should never use more hidden layer nodes than training samples. In fact, the number of hidden layer nodes should always be much smaller than the number of training samples, otherwise, the network can memorise the training samples, which leads to failure in classification of new and unseen data.

In neural network models, the weights are the free parameters. It is extremely important that a sufficient number of training samples is available to estimate these parameters accurately. A generally accepted guideline is to use at least five to ten times the number of training samples as free parameters (Klimasauskas, 1993; Messer and Kittler, 1998). For example, a network structure of 8-20-7 has 300 free parameters that require at least 1,500 training samples (3,000 samples would be optimal). If it is not possible to provide this number of training samples, the network will not be able to classify new data outside the training data with an acceptable level of accuracy. Since, in remote sensing studies, the volume of training data available is generally limited, this issue becomes quite important for applications. This limitation can be overcome by eliminating some irrelevant parameters from the network. Such parameters, which could be both input and hidden layer nodes, are the ones that are not contributing to the solution. It should be also noted that when the number of training samples is limited, it is expected that three-layer networks, which have a single hidden layer, perform better than four-layer networks. According to Hush (1989), 'this can be attributed to the fact that the four-layer networks provide too much flexibility. When the number of training samples is large the four-layer networks are forced to learn the same solution as the three-layer networks so their performance is about the same'.

Not only the size of the training samples but also the content of these samples is important. For example, for the same numbers of input and output nodes and training data size, many different networks would be optimal with respect to the characteristics of the training samples. Training data including a large amount of noise with similar characteristics among the classes would require more hidden layer nodes. Therefore, the heuristics considering the training data characteristics in some way should be favoured.

Another issue that should be considered is the type of strategy that will be used to end the training process. If an early stopping strategy is going to be employed, a large number of hidden layer nodes is needed to reduce the danger of arriving at a poor local minimum (Sarle, 1995). The general purpose of the project is also an important factor in defining the number of hidden layer nodes. For example, if the network is aimed to be used for feature extraction purposes, then fewer hidden layer nodes than input nodes are required.

In order to produce networks with high generalisation capabilities, node pruning techniques can be applied. A large network is trained initially, and later the least effective hidden layer nodes are eliminated. Such a methodology has the advantage of using a large network for training that prevents convergence to a local minimum and of producing a small network that has high generalisation capabilities and which is less complex and faster.

If the training error does not decrease to an acceptable level, then the number of hidden nodes should be increased. If the training error reaches an acceptable level but the classification accuracy on test data is low, then the size of the hidden layers should be reduced.

For the analysis of the effect of numbers of hidden layers and nodes on the performance of a neural network classification the two datasets described in Chapter V were employed. The network weights were initialised in the range  $[-0.5, 0.5]$ . The learning rate was set to 0.2 and reduced to 0.1 after 750 iterations. Trained networks were saved after every 2,500 iterations and tested on independent datasets. For a single hidden layer network, the number of nodes

varied from 1 to 25, and for the network with two hidden layers six network structures (5-5, 5-10, 10-10, 10-20, 16-14, 24-21) were considered. The results for both test sites are shown in Figures 6.8 and 6.9. It should be noted that both figures show the results produced at 15,000 iterations.

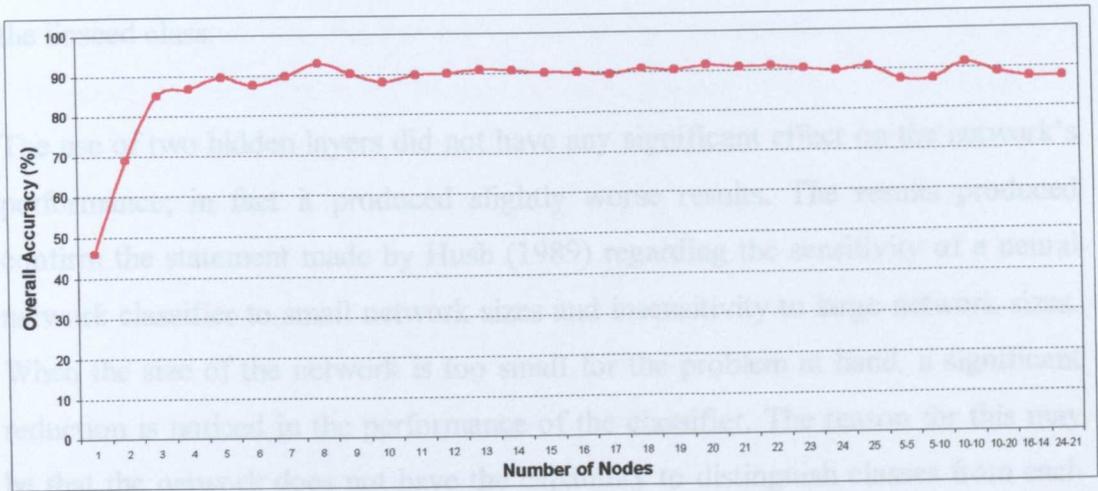


Figure 6.8 Effect of number of hidden nodes on classification accuracy for the first test site (Littleport dataset).

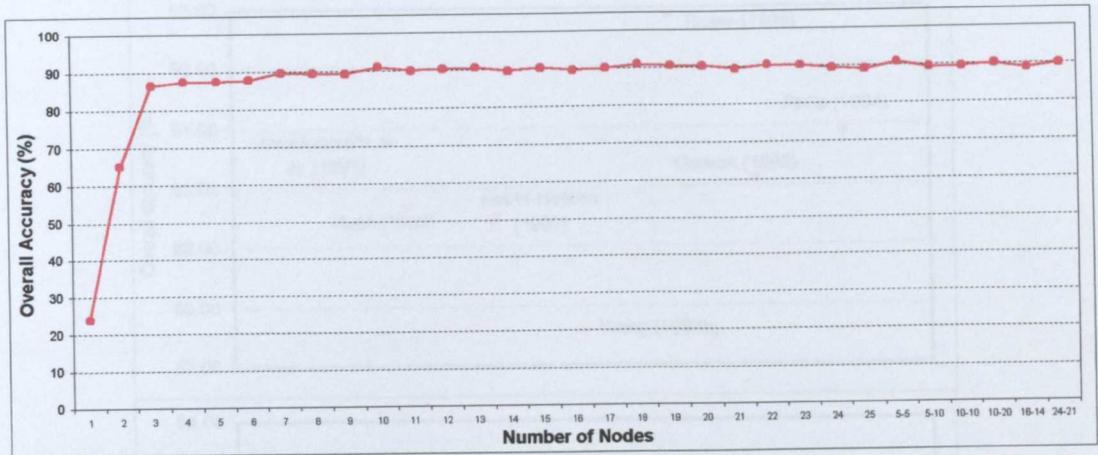


Figure 6.9 Effect of number of hidden nodes on classification accuracy for the second test site (Thetford dataset).

It is clear that three and more nodes on a single hidden layer and the two hidden layer configurations produced acceptable levels of accuracy (the variation was less than 5% overall accuracy). However, networks having smaller number of hidden layer nodes produced poor results, as low as 23% overall accuracy. Close analysis of the results reveals the nature of the failure of the small networks.

Whilst the 8-1-7 network structure could only recognise wheat and potato classes for the first dataset, it could only recognise peas and forest classes for the second dataset. Similarly, the network structure of 8-2-7 was ineffective in learning the characteristics of all the classes and therefore missed out a specific land cover type for both cases. For the first test site it was peas, and for the second site it was the linseed class.

The use of two hidden layers did not have any significant effect on the network's performance; in fact it produced slightly worse results. The results produced confirm the statement made by Hush (1989) regarding the sensitivity of a neural network classifier to small network sizes and insensitivity to large network sizes. When the size of the network is too small for the problem at hand, a significant reduction is noticed in the performance of the classifier. The reason for this may be that the network does not have the capability to distinguish classes from each other. On the other hand, when the network size is too large for the problem, the performance of the classifier stays almost unchanged.

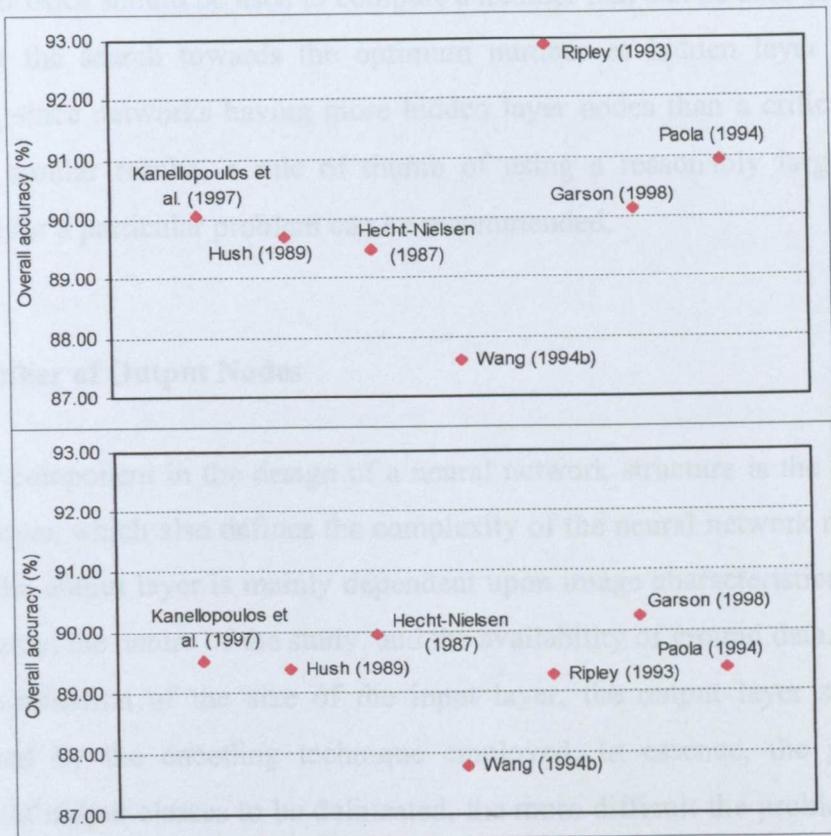


Figure 6.10 Comparison of the performances of the heuristics for the first (upper) and the second (lower) test datasets.

When the heuristics listed in Table 6.2 are compared in Figure 6.10 with respect to their effectiveness and reliability towards the determination of the optimum number of hidden layer nodes, it can be seen that all heuristics except for the one proposed by Wang (1994b) produced similar results. It should be noted that the classification problems considered in this study can be categorised as easy or moderate, therefore the lower bounds of the suggested heuristics were used. The smallest number of hidden layer nodes required seems to be 8 for both problems, which is pointed out by the heuristic given by Ripley (1993). The heuristic proposed by Hush (1989) suggests quite large number of nodes. Of the other heuristics, the one put forward by Garson (1998) can be favoured as it considers the difficulty of the problem using a noise-in-the-data coefficient. It should be also noted that the heuristic proposed by Paola (1994) also produced highly accurate results for both test datasets.

On the whole, instead of estimating the exact number of hidden layer nodes, the given heuristics should be used to compute a number that can be used as a starting point for the search towards the optimum number of hidden layer nodes. In addition, since networks having more hidden layer nodes than a critical number produce similar results, a rule of thumb of using a reasonably large network structure for a particular problem can be recommended.

#### **6.4 Number of Output Nodes**

Another component in the design of a neural network structure is the size of the output layer, which also defines the complexity of the neural network model. The size of the output layer is mainly dependent upon image characteristics, the scale of the study, the nature of the study, and the availability of ground data. Similar to the determination of the size of the input layer, the output layer size is also influenced by the encoding technique employed. In essence, the greater the number of output classes to be delineated, the more difficult the problem will be, due to the separation of input space into more specific regions. When there are the same numbers of input and output nodes, the network is called auto-associative

and performs a kind of mapping or encoding of inputs to outputs. When there are fewer output nodes than input nodes, the network model performs a type of compression of inputs into output. Such networks are used to implement principal components analysis.

## 6.5 Learning Rate and Momentum

The main disadvantage of the backpropagation learning algorithm is its slow convergence, which is largely related to the appropriateness of the learning rate chosen. The learning rate, also referred to as the step size, determines the size of the steps taken towards the global minimum of the error throughout the training process. It can be considered as the key parameter for a successful ANN application because it controls the learning process. If the learning rate is set too high, large steps will be taken, the system will be unstable, oscillating and failing to converge. If it is set too low, small steps will be taken, resulting in longer training times and a greater likelihood of becoming trapped in a local minimum, or a plateau area in the error surface. The momentum term, on the other hand, uses the previous weight configuration to determine the direction of the global minimum of the error. The learning rate with or without a momentum term is used to update the inter-node weights. A careful selection of the two parameters is often necessary for smooth convergence to a global minimum, leading to successful training.

Many configurations of the learning rate and momentum have been favoured in the literature, some of which are presented in Table 6.3. However, most of them are determined experimentally for a particular dataset or problem. As well as setting a constant learning rate value, a methodology varying the learning rate during training can be employed. For example, Swingler (1996a) suggests that starting with a large value for the learning rate ( $\sim 0.75$ ) and reducing to 0.25 and then to 0.1 as the network starts to oscillate is a good way of reaching the global minimum of the error.

In Table 6.3,  $N_p$  and  $N$  represent the number of training patterns and the total number of nodes in the network respectively, and  $C_o$  is a coefficient that is set to 10 based on the experience of corresponding researchers. In the formula given by Eaton and Olivier (1992),  $N_1, N_2, \dots, N_m$  are used to represent the sizes of  $m$  numbers of classes included in the training data.

Table 6.3 Heuristics for optimum learning rate and momentum term. Rates given in brackets are recommended by Eberhart and Dobbins (1990) for large datasets.

Learning rate	Momentum term	Source
0.01	0.00005	Paola and Schowengerdt (1997)
0.05	-	Lawrence <i>et al.</i> (1996)
0.05	0.5	Partridge and Yates (1996)
0.1	-	Haykin (1999), Gallagher <i>et al.</i> (1997)
0.1	0.3	Ardö <i>et al.</i> (1997)
0.1	0.9	Foody <i>et al.</i> (1996), Pierce <i>et al.</i> (1994)
0.15 (0.04)	0.075 (0.02)	Eberhart and Dobbins (1990)
0.2	-	Bischof <i>et al.</i> (1992)
0.2	0.6	Gong <i>et al.</i> (1996)
0.25	0.9	Swingler (1996a)
0.3	0.6	Gopal and Woodcock (1996)
0.5	0.9	Hara <i>et al.</i> (1994)
0.8	-	Staufner and Fischer (1997)
$C_o \frac{1}{N_p} \frac{1}{N}$	-	Heermann and Khazenie (1992)
$1.5/\sqrt{N_1^2 + N_2^2 + \dots + N_m^2}$	0.9	Eaton and Olivier (1992)

In addition to the rates given in Table 6.3, some sophisticated methodologies have also been developed to determine the optimum rates of the learning rate and momentum. These methods adapt the learning rate during the training process,

considering different characteristics of the error surface and error gradient. Such strategies are widely known as adaptive learning strategies.

Heermann and Khazenie (1992) propose an adaptive learning algorithm considering the training error. The algorithm increases the learning rate if the last training iteration results in a decrease in the error summed over all training patterns. Conversely, the learning rate is reduced (but not allowed to converge to zero) and the momentum term disabled if the error rises. Once the error begins to decrease again, the momentum term is included and the learning rate is increased with each good step. It is claimed that this technique speeds up the training process by a factor of 5 to 10 compared to methods using a fixed learning rate, without any loss in classification accuracy.

A number of methods have been proposed to set the learning rate and the momentum term for each weight in the network for better convergence. According to Haykin (1999), 'all neurons in the network should ideally learn at the same rates. The last layers usually have larger local gradients than the layers at the front end of the network. Hence, the learning rate parameter should be assigned to a smaller value in the last layers than in the front layers. Neurons with many inputs should have a smaller learning rate parameter than neurons with few inputs so as to maintain a similar learning time for all neurons in the network'. On the other hand, Le Cun (1993) suggests that for a given neuron, the learning rate should be inversely proportional to the square root of synaptic connections made to that neuron. A similar approach proposed by Hush and Horne (1993) sets the learning rate for each node to be inversely proportional to average magnitude of vectors feeding into the network. Kanellopoulos *et al.* (1992) state that by setting the learning rate for each layer of the network to be  $\eta/\text{number of inputs to each node in that layer}$  and setting the momentum term to zero, convergence was obtained more easily. Several attempts have been also made to adapt the learning rate according to the local curvature of the surface (Becker and Le Cun, 1988; Jacobs, 1988).

An extensive review performed by Moreira and Fiesler (1995) describes a large number of methods for learning rate and/or momentum term adaptation. They

categorise the techniques into several groups in terms of their theoretical basis, as follows:

- ◆ Based on numerical optimisation procedures using second-order information
  - Conjugate gradient
  - Quasi-Newton
  - Using a second-order calculation of the step size
- ◆ Based on Stochastic Optimisation
- ◆ Heuristic-based
  - Adaptation based on the angle between gradient direction in consecutive iterations
  - Adaptation based on the sign of the local gradient in consecutive iterations
  - Adaptation based on the evolution of the error
  - Prediction of a set of new values for the learning rate
  - Searching for zero-points of the error function instead of zero-points its derivative
  - Adaptation using the derivative of the error function in relation to the learning rate
  - Using peak values for the learning rate
- ◆ Others
  - Calculation of the optimal fixed values for the parameters before the training

Moreira and Fiesler (1995) also apply five popular techniques of optimisation to six real-world problems plus the Exclusive-OR (XOR) problem. The main conclusion that they draw from the results is that there is no clear best method among those that perform automatic parameter adaptation. Nevertheless, comparing the fixed parameter methods with the adaptive ones, considerable improvement has been made by the adaptation techniques.

It is also evident that optimum learning rate is dependent on the size of the training samples. Eaton and Olivier (1992) tested two networks having identical

topology with few training patterns (16) and with many patterns (192). They observed that different values of the learning rate produce good results for the networks. From this point they proposed a method (see Table 6.2) to compute a fixed value of learning rate that yields rapid training when coupled with a momentum term of 0.9 for a wide variety of networks.

A critical view of the use of adaptive learning rates, noted in Sarle (2000), is that many algorithms try to adapt the learning rate, but any algorithm that multiplies the learning rate by the gradient to compute the change in the weights is likely to produce erratic behaviour when the gradient changes abruptly.

After a large number of experiments performed in this study, several observations have been made regarding the nature of the learning rate and the momentum term. As a result, a strategy similar to the heuristic given by Swingler (1996a) for the learning rate setting can be recommended to reach the minimum error solution by reducing the learning rate during the course of training. In this strategy, the learning process is started with a large value of the learning parameter (i.e. 0.7) to avoid local minima and plateau areas in the error surface, and the rate was then reduced gradually from 0.3 to 0.05 to reach the global minimum of the error smoothly. Specifically, the learning rate is set to 0.7 for 1,000 epochs, then to 0.5, 0.3 and 0.2 for 500 epochs respectively, and lastly it is reduced to 0.1 and 0.05 for 250 epochs, totally 3,000 iterations. The process is more rapid than the one employing a constant learning rate of 0.2 in terms of reaching to low error level. The result is usually a line with steps. It can be observed that the error decreases sharply at the points where the learning rate is reduced. The strategy appears to perform better than the method employing a constant learning rate in terms of quickly moving towards the global minimum of the error (Figure 6.11).

Also, it has been observed that the momentum term can show erratic behaviour for small training samples. In order to portray this behaviour, a network was trained using 2,100 and 175 training samples. For this application, the learning rate was set to 0.2 and the weights in the network were initialised in the range [-0.3, 0.3]. Rates of 0.1 and 0.9 for the momentum term were applied with a constant learning rate to train the network. Training was continued for 1,000

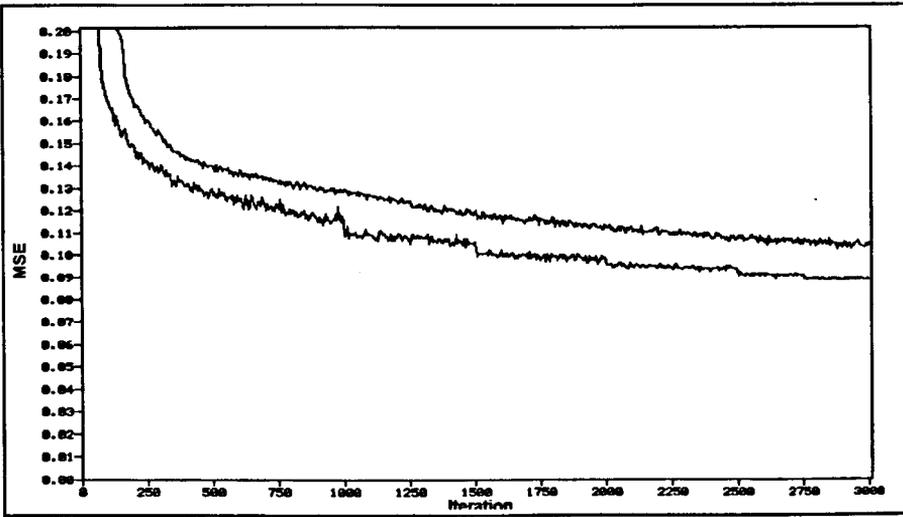


Figure 6.11 Comparison of using constant and varied learning rates. The undeviating line is resulted from the constant learning rate of 0.2 and the stepped line is a result of using 0.7, 0.5, 0.3, 0.2, 0.1 and 0.05 in order.

iterations and repeated a number of times to minimise possible bias caused by the effect of weight initialisation. The results of the processes are given in Figures 6.12 and 6.13. Note that the horizontal axes show the number of iterations and the vertical axes represent the MSE values. As can clearly be seen, for the small training dataset, a value of the momentum term of 0.9 caused unstable learning, which could result in unreliable classification performance. It can be concluded that small values of the momentum term should be employed to train small or limited number of training samples.

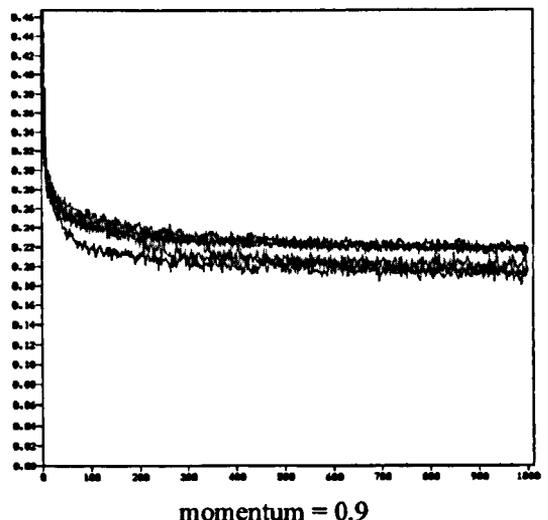
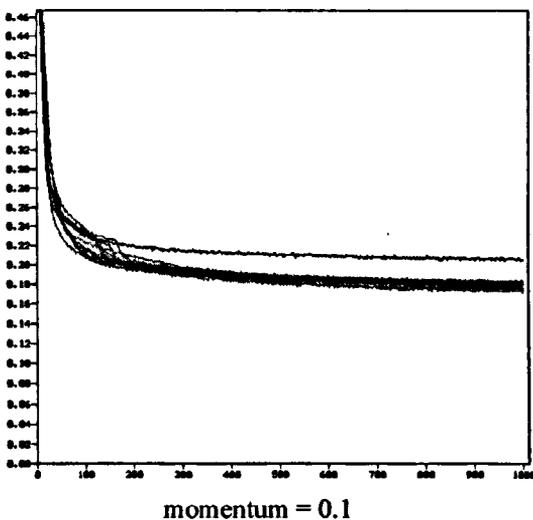
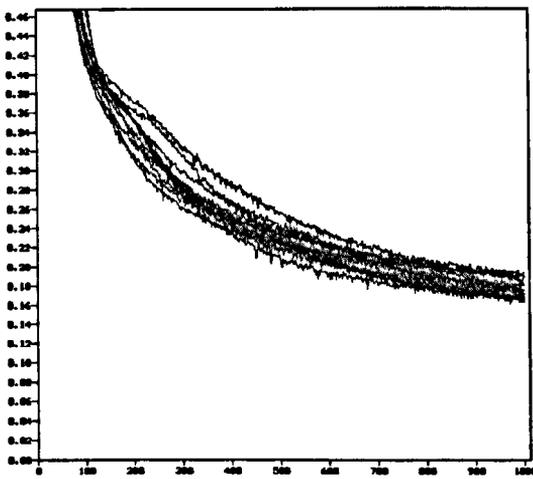
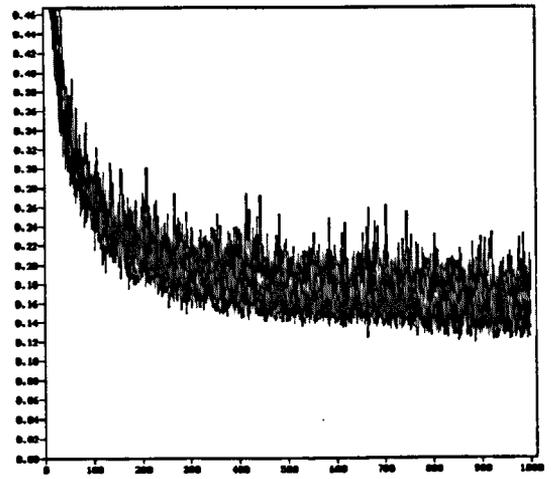


Figure 6.12 Training process for 2,100 samples at 0.1 and 0.9 momentum rates.



momentum = 0.1



momentum = 0.9

Figure 6.13 Training process for 175 samples at 0.1 and 0.9 momentum rates.

To investigate the effect of different learning rates and momentum terms on the network performance, a number of experiments was carried out employing the heuristics given in Table 6.3. These experiments can be divided into two parts. In the first part, six heuristics using only a constant learning rate in the training process were taken into consideration, and the combinations of the learning rate and the momentum term were considered in the second part. For the experiments two datasets were utilised. A network structure of 8-15-7 was trained using the backpropagation learning algorithm. The weights in the network were randomly initialised in the range  $[-0.3, 0.3]$ . All the parameters except for the learning rate and the momentum term were kept constant for all training experiments. The performance of the networks for different values of the learning rate at each 1,000-iteration is shown in Figures 6.14 and 6.15.

Several conclusions can be deduced from the Figures regarding the effect of the learning rate during the training process. It has been noticed that small learning rates produced consistent and high accurate results, whereas large learning rates appeared to cause oscillations and inconsistent results. As can be seen in Figure 6.14, the network using a learning rate of 0.2 produced the highest accuracies. However, good performances were also produced using learning rates of 0.05 and 0.1. Although the learning rate of 0.5 initially performed well, it did not maintain this performance. The opposite behaviour was observed from the use of a learning

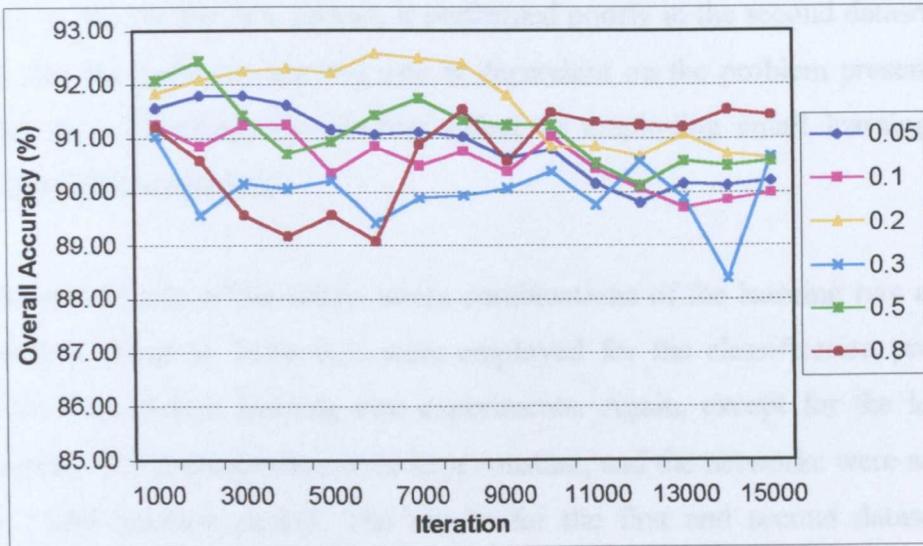


Figure 6.14 Overall accuracies produced for different learning rate configurations for the first dataset.

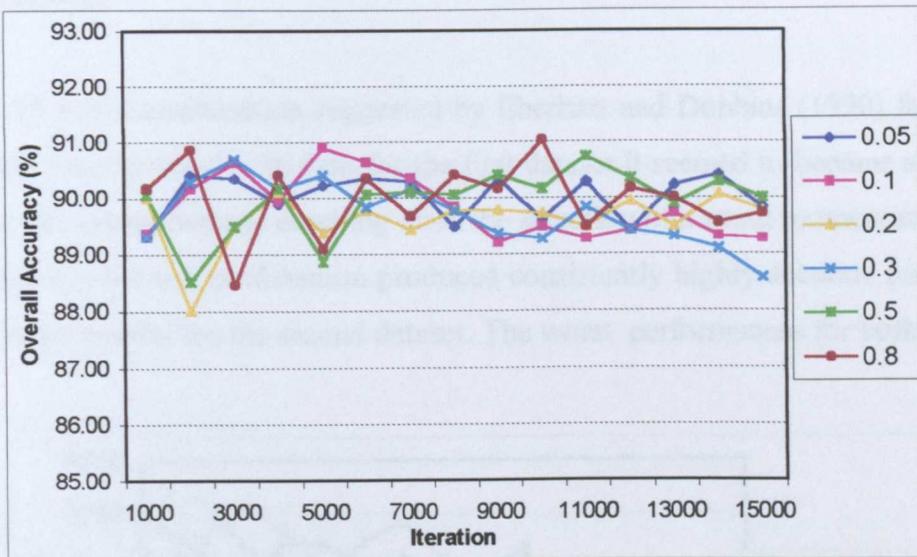


Figure 6.15 Overall accuracies produced for different learning rate configurations for the second dataset.

rate of 0.8. Unlike the situation shown in Figure 6.14, the overall accuracies produced for the second dataset, presented in Figure 6.15, are quite close to each other. However, it is possible to notice the consistency in performance produced using small learning rates. Large values for the learning rate (i.e. 0.5 and 0.8) resulted in oscillations around the global minimum, producing large deviations in the overall accuracy for nearby iterations. These oscillations sometimes led the network to work well at certain stages. Although the learning rate of 0.2 produced

the best results for the first dataset, it performed poorly in the second dataset. This shows that the optimum learning rate is dependent on the problem presented by training data. However, the positive effect of employing small learning rates should not be disregarded.

For the second part of the study, seven combinations of the learning rate and the momentum, listed in Table 6.3, were employed for the classification problems used for the constant learning rate experiments. Again, except for the learning parameters, all the parameters were kept constant, and the networks were saved at every 1,000 iteration period. The results for the first and second datasets are shown in Figures 6.16 and 6.17, respectively. Best results were overall produced by the combinations that use small learning rates, such as 0.05 and 0.1. In fact, for both cases, consistently good results were produced by the 0.05-0.5 and 0.1-0.3 combinations.

The 0.15-0.075 combination suggested by Eberhart and Dobbins (1990) failed to produce accurate results. In fact, for the first dataset it seemed to become stuck in a local minimum, perhaps resulting from the selection of a small momentum term. Whereas the 0.2-0.6 combination produced consistently highly accurate results, it was not successful for the second dataset. The worst performances for both cases

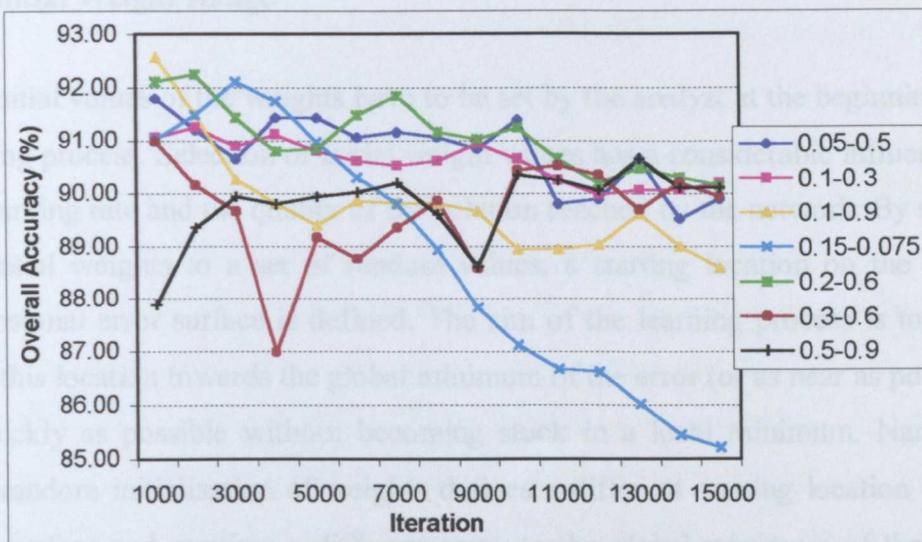


Figure 6.16 Overall accuracies produced for different configurations of the learning rate and the momentum for the first dataset.

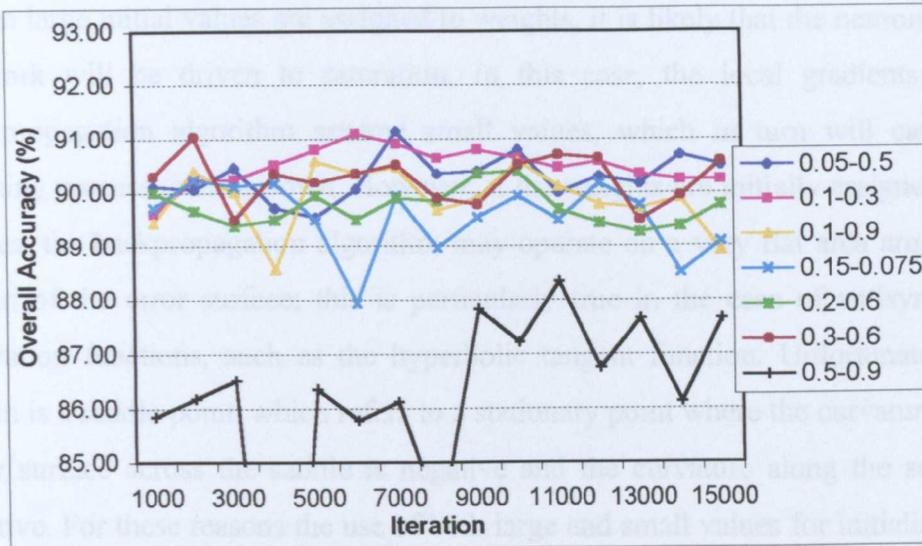


Figure 6.17 Overall accuracies produced for different configurations of the learning rate and the momentum for the second dataset.

were produced by the 0.5-0.9 combination. The reason for this could be that when such large learning rate causes oscillations, the use of large momentum term increases the effect of oscillations by extending the steps taken in faulty direction. One important observation was made: the addition of the momentum term to the training considerably slows down the learning process.

## 6.6 Initial Weight Range

The initial values of the weights have to be set by the analyst at the beginning of a learning process. Selection of initial weight values has a considerable influence on the learning rate and the quality of the solution reached by the network. By setting the initial weights to a set of random values, a starting location on the multi-dimensional error surface is defined. The aim of the learning process is to move from this location towards the global minimum of the error (or as near as possible) as quickly as possible without becoming stuck in a local minimum. Naturally, each random initialisation of weights defines a different starting location on the error surface and requires a different route to the global minimum of the error. When all weights are set to zero, no learning takes place due to the formulation provided for the backpropagation learning algorithm.

When large initial values are assigned to weights, it is likely that the neurons in the network will be driven to saturation. In this case, the local gradients in the backpropagation algorithm assume small values, which in turn will cause the learning process to slow down. However, if the weights are initially assigned small values, the backpropagation algorithm may operate on a very flat area around the origin of the error surface; this is particularly true in the case of antisymmetric activation functions, such as the hyperbolic tangent function. Unfortunately, the origin is a saddle point, which refers to a stationary point where the curvature of the error surface across the saddle is negative and the curvature along the saddle is positive. For these reasons the use of both large and small values for initialising the synaptic weights should be avoided (Haykin, 1999). It is also suggested that the mean value of the initial weights should be zero and the variance should be equal to the reciprocal of the number of synaptic connections of a neuron.

Although several investigations, which are discussed in the following section, have been carried out in order to examine the effect of different initial weight configurations, to date there is no method (or guideline) universally accepted for the determination of an optimum range. The problem is not only to determine the range for initial weights, but also to investigate the replicability of the solution when the process of random weight initialisation has been performed a number of times. It is reported that the biggest problem is the significant effect of different initial weight configurations over the same range.

Kolen and Pollack (1990) explore the effect of initial weight selection on feed-forward networks learning simple functions with a backpropagation learning algorithm. The results of their experiments show the extreme sensitivity of the backpropagation algorithm to the initial weight configuration. From this point, they suggest that when theoretical claims are made from experience regarding the power of an adaptive network to model some phenomena, the initial conditions for the network need to be precisely specified or filed in a public scientific database. Blamire (1996) and Ardö *et al.* (1997) also report significant differences in the accuracy produced by the neural networks when the weights in the networks are randomly initialised. Specifically, Blamire (1996) observes that the

overall accuracy of the classification ranged between 86 percent and 90 percent, and the Kappa coefficient varied from 0.72 to 0.80. Unfortunately, he does not provide any information about his choice of weight range. The effect was more severe for the classification problem considered by Ardö *et al.* (1997). Test data accuracy ranged from 59 percent to 70 percent when the weights were initialised 30 times between 0 and 1. From the results they conclude that the use of random initial weights makes it impossible to repeat the learning part of a neural network application. Due to such behaviour observed by researchers, it is recommended that the learning process is repeated a number of times each with different initial weights over the same range. The advantage of this proposal is that each time the search for the global minimum of the error is started from different parts of the error surface. The solution that produces the best accuracy is chosen and used for further analysis.

The effect of employing different initial weight ranges in the learning process was also investigated by Skidmore *et al.* (1997) where the network parameters were held constant, except that the starting weights were randomly adjusted by  $\pm 5$  percent. The resulting five classification maps were visually different and a large variation was noticed in training and test accuracies. While the overall accuracy on training data was ranging from 90 percent to 97 percent, that on test data was ranging between 42 percent to 55 percent. Such results seem to have negative effect on the applicability and usefulness of neural networks.

Approaches used to determine optimum initial weights can be grouped into two main categories. The first group uses different distributions for the weights. However, the second group of approaches is based on Thimm and Fiesler (1997b):

- the steepness of the sigmoidal function,
- the number of connections feeding into a neuron (called fan-in of a neuron),
- (analysis of) the dataset on which the network will be trained,
- the number of connections in the network, and
- constants that emerged from experiments.

Several sophisticated approaches based on the above criteria, described in Thimm and Fiesler (1997a), have been developed to determine the best initial weight range for a successful neural network application. Wessels and Barnard (1992) suggest two methods of weight initialisation. The first method initialises the weights in the range  $[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}]$  where  $d_{in}$  denotes the number of weights leading to a particular node. In this method it is assumed that the output of the network and the output patterns have the same variance. The second method initialises weights in such a fashion that the following conditions are met: 1) the decision boundaries of the hidden nodes should be positioned well within the region occupied by the training samples; 2) the orientations of the decision boundaries of the hidden nodes have to be as varied as possible; and 3) every part of the sample region needs at least one hidden node which is active for samples occupying that region. In their comparative study using three datasets, they found their second technique more robust than the first one in terms of generalisation performance.

In an approach similar to that adopted by Wessels and Barnard (1992), Boers and Kuiper (1992) propose that the initial weights of each node to be in the range  $[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}]$ , where  $d_{in}$  is the number of connections feeding into a neuron (or fan-in). They note that if this range is used to calculate the random initial weights, the network will always have a reasonable initial weight setting no matter what the size of the network may be. Smieja (1991), on the other hand, initialises weights using a uniform distribution and having a magnitude of  $2/d_{in}$ .

Denoeux and Lengellé (1993) introduce a technique that relies on the use of reference patterns, or prototypes, to determine initial weights. Their simulations have shown that the method yields drastic reductions in training time, and considerably improves robustness with regard to local minima. Experimental results also suggest that networks initialised with prototypes show better generalisation properties. de Castro *et al.* (1998), on the other hand, propose a method that uses a genetic algorithm to analyse the space of weights, in order to achieve good initial conditions for supervised learning. They find the proposed

method more robust and better in terms of the convergence speed. Two important conclusions they draw from the results are: (i) initialising the weights predominantly in the approximately linear part of the activation function makes the training faster and less subjective to numerical instability (ii) the weights have to be well distributed around the origin in the weight space in order to generate a broad coverage of the search space.

A comprehensive study carried out by Thimm and Fiesler (1997a) tests major sophisticated methods used for random weight initialisation using eight real-world benchmark datasets and a broad range of initial weight variances. Several conclusions are drawn from a large number of results. Firstly, the weight initialisation method (the first one) proposed by Wessels and Barnard (1992) performed best, on average. Secondly, a fixed weight variance of 0.2, which corresponds to a weight range of  $[-0.77, 0.77]$ , gave the best mean performance for all the applications tested. Finally, the experiments show that the best initial weight variance is determined by the dataset. Therefore, some reasoning on the dataset has to be included in the determination of this value.

In the remote sensing literature, the use of constant initial weight ranges, determined empirically, have been usually suggested and used for a variety of problems. However, the underlying ideas behind these choices are not usually presented. As one of the proposers, Eberhart and Dobbins (1990) state that initial weight range should be set to  $[-0.3, 0.3]$ . The justification they give for their choice is that there is no better reason than 'it works'. A list of initial weight ranges employed by researchers is shown in Table 6.4.

The heuristics given in Table 6.4 suggest quite different ranges of initial weights. Determining the most appropriate one is fundamentally related to several factors, including the network size, number of training samples and the learning parameters. The relationships between these factors and the initial weight range have not been explored in detail to date. In this study, these issues are considered and a large number of experiments have been performed to derive some ideas concerning the selection of most appropriate initial weight range prior to neural network training process.

Table 6.4 Initial weight ranges used by some researchers.

Initial Weight Range	Source
[0 , 1]	Ardö <i>et al.</i> (1997)
[-0.1 , 0.1]	Paola (1994), Gopal and Woodcock (1996), Lawrence <i>et al.</i> (1996), Bebis <i>et al.</i> (1997), Paola and Showengerdt (1997), Staufer and Fischer (1997)
[-0.15 , 0.15]	Vuurpijl (1998)
[-0.25, 0.25]	Gallagher and Downs (1997)
[-0.3 , 0.3]	Rumelhart <i>et al.</i> (1986), Eberhart and Dobbins (1990)
[-0.5 , 0.5]	Sietsma and Dow (1991), Huurneman <i>et al.</i> (1996), Partridge and Yates (1996)
$[-2/d_{in}, 2/d_{in}]$	Gallant (1993)
$[-2/\sqrt{d_{in}}, 2/\sqrt{d_{in}}]$	Smieja (1991)
$[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}]$	Boers and Kuiper (1992), Wessels and Barnard (1992)

A network structure of 8-10-7 was selected for the implementation as it was found to be adequate to produce accurate classification results. The learning rate was initially set to 0.2, and reduced to 0.1 after 750 iterations. For the Littleport dataset, 2,262 samples were used to train the networks, and 2,204 samples were used to test the network performances. For the Thetford dataset, while 2,100 samples were employed to train the networks, 1,750 samples were used to test the networks. Six initial weight ranges were employed with different configurations in terms of network size, training data size, and learning parameters. During the training, networks were saved after 2,500 iterations to allow the observation of the trend in the network performance. The results of the process for the datasets are presented in Figures 6.18 and 6.19, respectively. It should be noted that whilst the horizontal axes represent the number of iterations performed, the vertical axes show the overall accuracy achieved. In the figures, the component defined as *original* represents the results produced when the above conditions were set. The line called *pattern* was produced for the datasets having only 350 patterns for training. On the other hand, the *network* line shows the results when the network size was reduced to 8-4-7 (4 hidden layer nodes as opposed to 10). Moreover, four

configurations were established to investigate the effect of learning parameters. While 0.05 line represents the results for the learning rate of 0.05, lines for 0.05, 0.5 and others are given for the combinations of the learning rate and the momentum term.

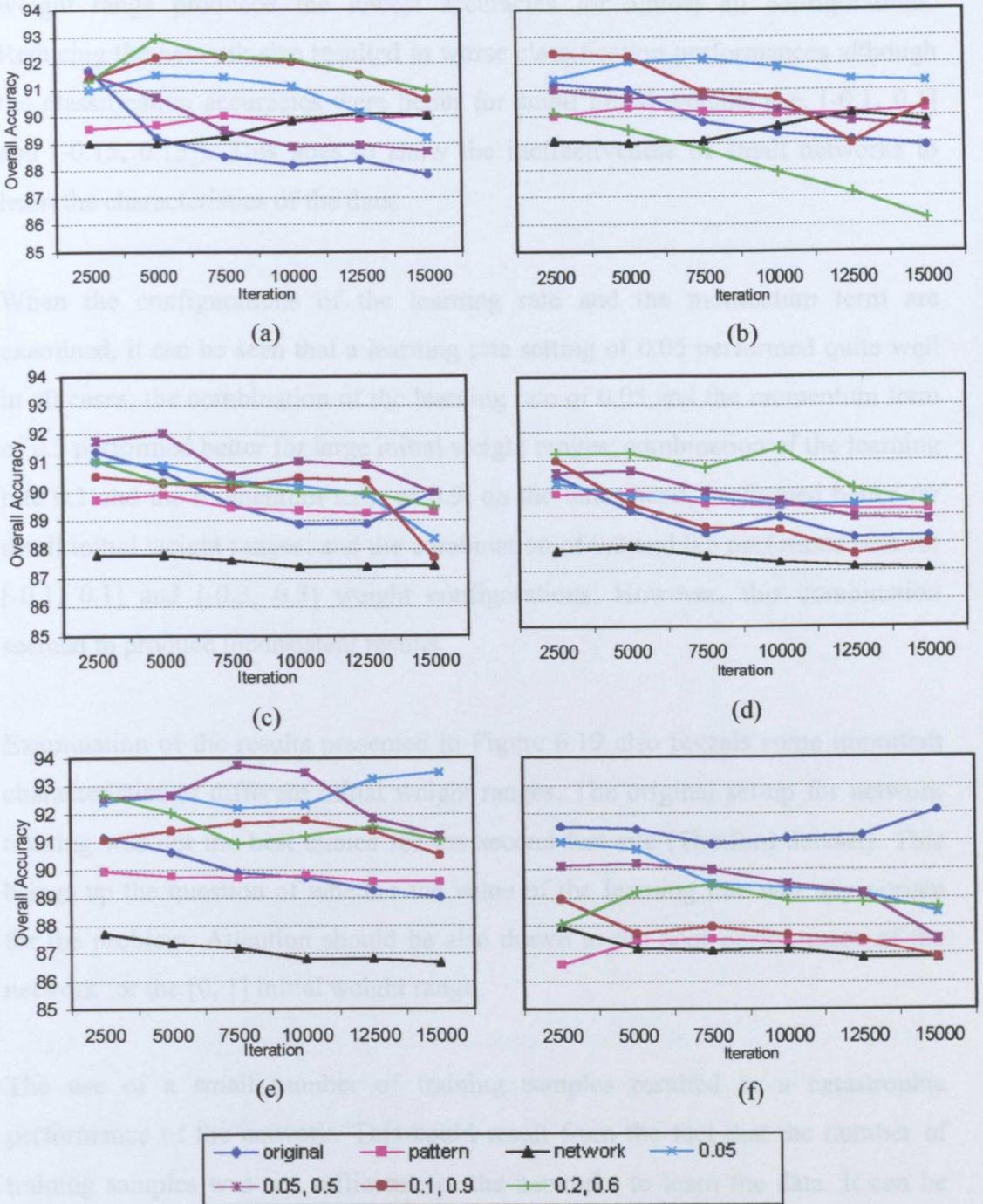


Figure 6.18 Variations in the overall accuracy depending on the initial weight range for the Littleport dataset. Initial weight ranges are (a) [-0.1, 0.1], (b) [-0.15, 0.15], (c) [-0.25, 0.25], (d) [-0.3, 0.3], (e) [-0.5, 0.5], (f) [0, 1].

Several conclusions can be drawn from Figure 6.18. First, the original set-up for the network and learning could not perform optimally for all cases, but only for the range  $[0, 1]$ . Reducing the number of training samples did not cause serious problems except for the  $[0, 1]$  weight range. In fact, in the use of this particular weight range produced the lowest accuracies for almost all configurations. Reducing the network size resulted in worse classification performances although the classification accuracies were better for small initial weights (i.e.  $[-0.1, 0.1]$  and  $[-0.15, 0.15]$ ). This goes to show the ineffectiveness of small networks to learn the characteristics of the data.

When the configurations of the learning rate and the momentum term are examined, it can be seen that a learning rate setting of 0.05 performed quite well in all cases; the combination of the learning rate of 0.05 and the momentum term of 0.5 performed better for large initial weight ranges; combination of the learning rate 0.1 and the momentum term of 0.9, on the other hand, performed better for small initial weight ranges; and the combination of 0.2 and 0.6 performed best for  $[-0.1, 0.1]$  and  $[-0.3, 0.3]$  weight configurations. However, this combination seemed to produce inconsistent results.

Examination of the results presented in Figure 6.19 also reveals some important characteristics of different initial weight ranges. The original set-up for network training was not the best choice for the second test site (Thetford dataset). This brings up the question of whether the value of the learning rate was appropriate for the problem. Attention should be also drawn to the poor performance of the network for the  $[0, 1]$  initial weight range.

The use of a small number of training samples resulted in a catastrophic performance of the network. This could result from the fact that the number of training samples was not sufficient for the networks to learn the data. It can be also related to the selection of patterns in terms of their representativeness. Similar to the results reported for the first test site, reducing the network size resulted in lower classification accuracies. The results verify once again the importance of network size on the network performance.

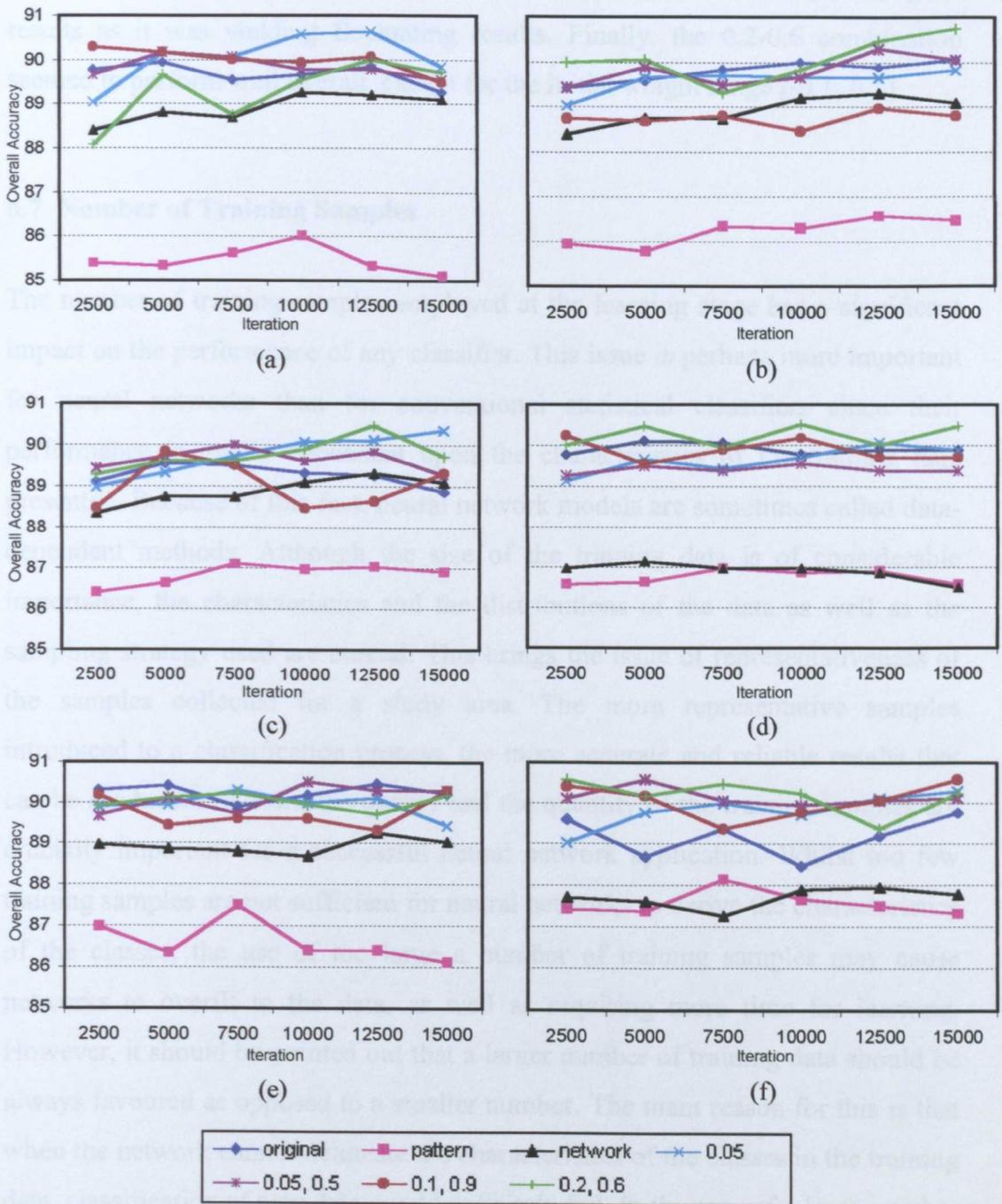


Figure 6.19 Variations in the overall accuracy depending on the initial weight range for the Thetford dataset. Initial weight ranges are (a) [-0.1, 0.1], (b) [-0.15, 0.15], (c) [-0.25, 0.25], (d) [-0.3, 0.3], (e) [-0.5, 0.5], (f) [0, 1].

The effect of combinations of the learning rate and the momentum term in conjunction with different initial weight ranges over the network performance can also be examined from Figure 6.19. Whilst a learning rate of 0.05 performed well in all cases, a learning rate 0.05 and a momentum term 0.5 combination performed

slightly less well. The 0.1-0.9 combination did not produce consistently good results as it was yielding fluctuating results. Finally, the 0.2-0.6 combination seemed to perform well overall, except for the initial weight range [-0.1, 0.1].

## **6.7 Number of Training Samples**

The number of training samples employed at the learning stage has a significant impact on the performance of any classifier. This issue is perhaps more important for neural networks than for conventional statistical classifiers since their performance is totally dependent upon the characteristics of the training data presented. Because of this fact, neural network models are sometimes called data-dependent methods. Although the size of the training data is of considerable importance, the characteristics and the distributions of the data as well as the sampling strategy used are crucial. This brings the issue of representativeness of the samples collected for a study area. The more representative samples introduced to a classification process, the more accurate and reliable results that can be produced. In short, the quality and the quantity of the training samples are crucially important for a successful neural network application. Whilst too few training samples are not sufficient for neural networks to derive the characteristics of the classes, the use of too large a number of training samples may cause networks to overfit to the data, as well as requiring more time for learning. However, it should be pointed out that a larger number of training data should be always favoured as opposed to a smaller number. The main reason for this is that when the network cannot delineate the characteristics of the classes in the training data, classification of new data would definitely fail. In the case of a large number of training samples, the outcome is only a degree of reduction in the performance. Hush and Horne (1993) state that the more training we have, the more incorrect functions we are able to reject, and the more likely we are to find the correct function.

As discussed earlier, using more features (i.e. spectral bands) requires more training samples. This relationship can be thought of as a linear one. As in most remote sensing studies, the size of training samples is limited, the design of a

neural network is partly based on the number of training samples. However, in most cases the question is how many training samples are required to produce optimal (or near-optimal) classification results. The answer to this question is difficult as it depends on many factors, including the difficulty of the problem, training data characteristics, and the neural network structure. In a study where the error surfaces for Multilayer Perceptrons (MLP) are analysed, Hush *et al.* (1992) observe that when the number of training samples is small, the error surface includes stair steps, one for each training sample. When the number of training samples is increased, the surface becomes smoother as the steps smear together. Stauffer and Fischer (1997) report that their experimental results clearly indicate that the generalisation performance measured in terms of total classification accuracy generally increases with increasing training set size. Moreover, the surface appears more complex when there is overlap from different classes. Similar results are also reported by Ahmad and Tesauro (1989). They find that, for a fixed network size, the failure rate decreases exponentially with the size of the training set. In addition, the number of patterns required to achieve a fixed performance level was shown to increase linearly with the network size.

A study performed by Zhuang *et al.* (1994) investigates the number of training samples, in terms of the percentage of the study area used (10.36 km<sup>2</sup>), required by neural networks to classify six features using a Landsat TM scene. They conclude that using approximately 5-10 percent of the image data was adequate to train the neural network. This conclusion seems to be incomplete and misleading since it is fully dependent on the size of the study area, and does not consider the difficulty of the problem defined by the numbers of inputs and outputs.

There have been several attempts in the literature to estimate the optimum number for training samples in relation to the network size and the accuracy level desired. The heuristic, proposed by Klimasauskas (1993) and noted earlier, suggesting using five training samples for each weight in the network can be applied to determine the number of training samples needed. In this case, size of the training data is estimated for a considered network. It is also worth noting that for conventional statistical classifiers at least  $30p$  pixels per class, where  $p$  is the number of features should be used (Mather, 1999a). This rule can also be applied

to neural networks as they are considered to be better in terms of handling the small training datasets (Hepner *et al.*, 1990; Blamire, 1994 and Foody, 1995).

In the performance analysis of neural networks for classification problems, Hush (1989) observes from a series of experiments that at least  $30N_i(N_i + 1)$  training samples are required. In order to achieve near optimal performance he recommends to use  $60N_i(N_i + 1)$  training samples. According to this formula, a considerable number of training samples are required. For example, for a network having ten input bands 6,600 training patterns are necessary to produce near-optimal results. Another weakness of the suggested idea is that the sizes of other elements of the network (i.e. the output and hidden layers) are not considered.

Garson (1998) presents several rules of thumb in order to determine the optimum number of training samples to produce acceptably accurate results. Specifically, a liberal rule of thumb is that the number of training samples should be at least 10 times the number of inputs. A conservative rule of thumb is that the number of training samples should be at least 10 times the number of input and middle layer neurons in the network. Another rule of thumb is to use 30 times as many input patterns as network weights to avoid overfitting.

An approach to the problem of determining the optimum number of training samples is to consider the generalisation error of the network, which is defined as the difference between the generalisation on the training data and the generalisation on the actual problem. In many cases, it is found that the difference between the two generalisations can be bounded, and by increasing the number of training samples this bound can be made arbitrarily small. This bound can be established when the number of training samples exceeds the *Vapnik-Chervonenkis Dimension* (VC dimension). Whilst Hush and Horne (1993) define the VC dimension as a measure of the capability of the system, Sontag (1998) describes it as a quantity which characterises the difficulty of distribution-independent learning. The VC dimension of a one-hidden-layer network with full connectivity between the layers is in the range (Fu, 1994):

$$2\lceil N_h/2 \rceil N_i \leq \text{VC dimension} \leq 2N_w \ln(N_n) \quad (6.1)$$

where  $\lfloor \cdot \rfloor$  is the *floor* operator that returns the largest integer less than its argument,  $N_h$  is the number of hidden units,  $N_i$  is the number of input units,  $N_w$  is the total number of weights in the network, and  $N_n$  is the total number of nodes in the network. The upper bound holds no matter what the number of layers and the connectivity are. As a rule of thumb, the number of weights can give a rough estimate of the VC dimension. The above statement assumes that the network uses a hard-limiting activation function. In the case of the sigmoid activation function, Sontag (1989) suggests that the VC dimension is at least twice as large as the one estimated for a hard-limiting activation function. Baum and Haussler (1989) propose that if an accuracy level of 90% is desired, the number of training samples should be about 10 times the VC dimension, or the number of weights in the network.

In order to investigate the effect of number of training samples on the performance, a number of training sample sizes, including the ones produced by the heuristics mentioned above, were considered for two real-world datasets. A network structure of 8-13-7, having 195 weights in total, was found to be adequate for both problems. The learning rate was set to 0.2 and the network weights were randomly initialised in the range [-0.3, 0.3]. While the test datasets for both sites included 150 samples for each class type, the training sample sizes varied between 250 and 5,850. In addition to the numbers estimated from heuristics, shown below, training sample sizes were determined for certain intervals. In the process of sample selection, after randomly selecting the test samples from the image, training samples were randomly selected from remaining pixels having ground truth information attached.

$5 \times N_w = 975$	Klimasauskas (1993)
$30p = 1680$	Mather (1999a)
$10 \times N_w = 1950$	Baum and Haussler (1989)
$30 \times N_i \times (N_i + 1) = 2160$	Hush (1998) [at least]
$60 \times N_i \times (N_i + 1) = 4320$	Hush (1998) [optimal]
$30 \times N_w = 5850$	Garson (1998)

During the training process of 10,000-iteration period, solutions reached at every 1,000-iteration were recorded and later assessed using the test datasets. From these solutions the one that yields the highest accuracy in terms of overall accuracy and the Kappa coefficient was chosen to form Figure 6.20, in which the results are given for the two test sites. For both cases, a gradual increase trend can be initially observed starting from the least number of training samples employed. However, after certain number of training samples, no significant improvement in classification accuracy is observed. For the first classification problem 2,160 training samples appear to produce a higher level of accuracy compared to experiments using fewer samples. Although this number was not apparent for the second dataset, 1,500 samples appeared to be critical for the network's performance. The heuristic that was proposed by Klimasauskas (1993) suggests an insufficiently small number of samples, and the one presented by Garson (1998) suggests an excessively large number of samples. Of the other heuristics, the ones that were recommended by Baum and Haussler (1989), and Hush (1998) indicate numbers that are close to the optimum numbers. The most important result is that the effect of number of training samples on the classification accuracy produced is not as severe as expected. This may be due to the fine selection of representative samples for the classes under consideration.

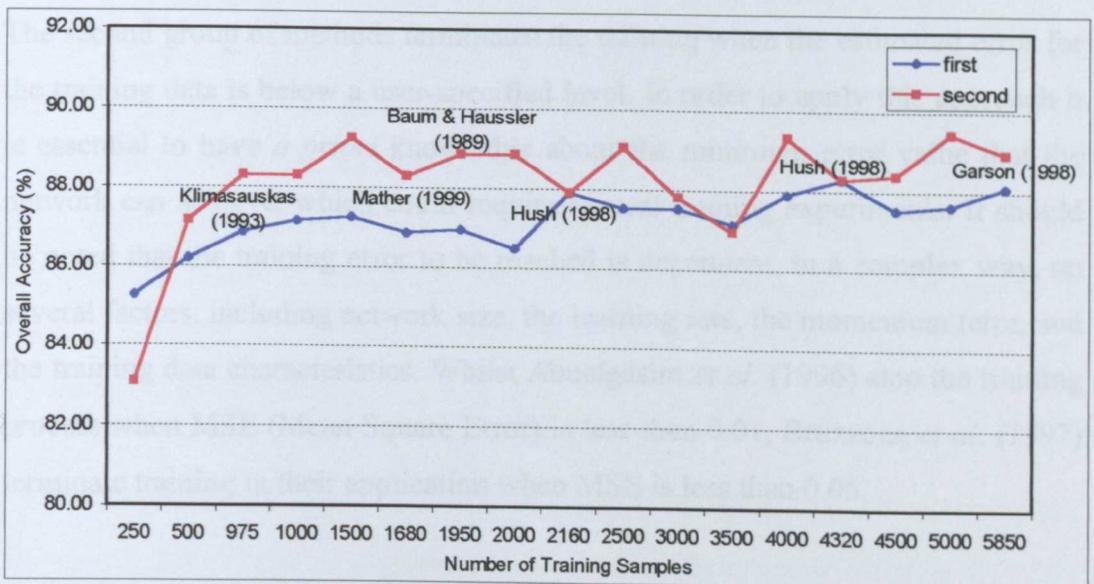


Figure 6.20 Effect of number of training samples on classification accuracy for two datasets.

## 6.8 Stopping Criterion for the Training Process

As it is generally impossible to train neural networks for real-world problems until they classify all training samples correctly, indicating zero training error, a stopping criterion has to be established. One of the difficulties in the use of neural networks is to determine the point at which the learning process is to be terminated before overfitting occurs. Underlying relationships in the training data are usually determined in early stages of the learning process. As training continues, the network tends to fit to the noise rather than the data structure. Several suggestions have been made to help determine the point at which the learning process should stop. The first group of suggestions is based on the magnitude of the gradient of the error. The learning algorithm is stopped when the magnitude of the gradient is small, assuming that the gradient will be zero at the minimum of the error. However, there is a danger that the magnitude of the gradient will also be small around local minima and plateau areas of the error surface. Therefore, a careful design of this strategy is essential to reach the global minimum of the error. Such a strategy is employed by Hara *et al.* (1994) in that a network is trained until the total RMS (Root Mean Square) error remained constant to at least three decimal digits.

The second group of methods terminates the training when the estimated error for the training data is below a user-specified level. In order to apply this approach it is essential to have *a priori* knowledge about the minimum error value that the network can achieve, which could require several training experiments. It should be noted that the training error to be reached is dependent, in a complex way, on several factors, including network size, the learning rate, the momentum term, and the training data characteristics. Whilst Abuelgasim *et al.* (1996) stop the training process when MSE (Mean Square Error) is less than 0.01, Bruzzone *et al.* (1997) terminate training in their application when MSE is less than 0.05.

A simple approach to the problem is to train the network a pre-defined number of times, hoping that the network will reach the global minimum of the error. Whilst fewer iterations than the required number do not guarantee a sufficient level of

learning, too many iterations can cause network to overfit to the data. Even if the network reaches the global minimum after a long training period, the performance of the network may not be as satisfactory as stopping the training just before the global minimum of the error, as reported by Wang (1994a). Paola and Schowengerdt (1994), Thackrah *et al.* (1999) and Gong (1996) employed this strategy to terminate the training process.

A more appropriate way of stopping the learning is to employ a validation dataset to monitor the generalisation capability of the network at certain defined points in the training process. This is known as cross-validation. In practice, the learning process is terminated when the error estimated for the validation dataset starts to rise. It is assumed that the network tested on a validation dataset will perform equally well on the test data. During the learning, the performance of the network tends to increase on the training data, whereas its performance on the validation data increases up to a point, where the network starts to overfit the training data and the generalisation capability starts to decrease. The main advantage of this approach is that it does not suffer from the effects of network size and the choice of values of the learning parameters. However, there are three drawbacks to cross-validation. Firstly, it is computationally more demanding, and therefore requires more time. Secondly, in addition to training and test datasets, a validation dataset has to be prepared. This could be a potential problem for the cases in which only a limited number of samples is available. Finally, it can be misleading to stop the learning considering the first rise in the error on validation data since the error usually continues to decrease after the first rise. Therefore, determining the best point to stop using cross-validation is not straightforward, as it requires careful design of the learning process. Several researchers, including Kanellopoulos *et al.* (1992), Blamire (1994), and Blamire and Mineter (1995), employed cross-validation in their studies.

The strategies described above are investigated in this study to determine the best stopping epoch. Training and validation datasets were formed for both test sites. For the first test site, the training dataset included 1,750 samples, and the validation dataset contained 1,120 samples. For the second test site, 1,750 and 980 samples were taken for the training and the validation datasets, respectively. In

addition, 4,000 samples were used for both sites to test the performance of the trained networks. All datasets were randomly selected from the images of the test sites. The network structure of 8-20-7 was chosen for the study. For the standard backpropagation learning, weight values were initialised in the range  $[-0.3, 0.3]$  and the learning rate was set to 0.2 for all experiments carried out. The experiments were implemented through specific configuration files written for *batchman* program provided by the SNNS software.

For the first and the simplest method, it was found that 5,000 iterations were sufficient for the network to learn the characteristics of the data. Therefore, training processes for both datasets were terminated when 5,000 iterations were completed. The resulting networks were saved and later assessed on test datasets. The second method of stopping the learning process was based on terminating the process when reaching to a pre-defined MSE level for the training datasets. By considering the difficulty of the problems, 0.13 and 0.09 MSE levels were set for the first and second datasets, respectively. For the last method, a validation dataset was employed to determine the best epoch at which the network can perform best on the validation dataset, assuming that at this point that network has the best generalisation capabilities.

It was observed that stopping the training process when the error on the validation dataset starts to increase could be misleading, since slight fluctuations in error during training are common. For the problems considered here, the first rises were observed at 100 and 180 iterations for the first and second datasets, respectively. Such a small number of iterations would not be sufficient for the network to identify the patterns inherent in the datasets. There are two major reasons for this behaviour. Firstly, the training and validation datasets do not represent exactly the same characteristics of the problem. Secondly, oscillations around minima in the error surface could be encountered for a short period of time. Therefore, a careful design and special set-up is required, which is implemented in this study through configuration files. In the methodology adopted here, MSE level for the validation data was checked every 20 iterations. At the end, if any improvement over 0.001 was achieved by the network for the validation data, the network was saved and the MSE level was set as a threshold for the next error rates. The reason for

setting an improvement parameter (0.001 in this case) was to determine the epochs from which significant improvements were attained. If no improvement was achieved, the training process was continued for another 500 iterations to ensure that the global minimum of the error was found. If in any stage of the 500-iteration period a significant improvement encountered, the network was saved again and the new threshold was set for the MSE level. The results of applying the three methods with above considerations are listed in Table 6.5.

Table 6.5 Results of the three major training termination methods for two datasets. MSE values in brackets are computed for the validation datasets.

Stopping Criterion	First Test Site				Second Test Site			
	Iteration	MSE	Overall	Kappa	Iteration	MSE	Overall	Kappa
Fixed Iteration	5000	0.0864	86.52	0.8416	5000	0.0799	89.95	0.8797
MSE on Training Data	1860	0.1297	85.75	0.8337	2900	0.0898	89.18	0.8710
Using Validation Data	1140	0.1432 (0.155)	86.88	0.8457	980	0.1155 (0.137)	89.13	0.8702

Although the results produced are quite close to each other, comparisons can be made regarding their characteristics and reliabilities of the methods. The use of a fixed number of iterations produced slightly more accurate results for the second dataset, and the least accurate results for the first dataset. Even though the improvements in the classification accuracy were not substantial, the results show the unstable nature of the method. As well as being extremely difficult to determine the number of iterations required prior to any experiment, there is no guarantee for preventing underfitting or overfitting.

Determination of the optimum number of iterations from the error level achieved on a training dataset is always biased towards the characteristics of the dataset. The error level that can be achieved by a network for a training dataset usually decreases with the number of iterations. However, it should be borne in mind that

the error level for the validation and test datasets can decrease up to a level and then starts to increase, ignoring the fact that some small fluctuations should be expected in the error level.

It appears that the most sophisticated method of the three is the method employing the validation dataset in the training for monitoring the generalisation capabilities during the process of learning. Using this method, the training processes for both cases were terminated in early stages. A smaller number of iterations also produced good results. In particular, for the first dataset the best performance was achieved with the overall accuracy of 86.88%. The classification accuracy produced for the second dataset was also comparable to others produced. Since the number of iterations is small, the generalisation capabilities of the networks are expected to be greater, and less time is required for training.

## 6.9 Output Encoding

Output encoding is another issue that must be considered prior to the training process. With the output encoding the real world features or classes are represented in the network in a special way. Encoding techniques for input and output information are described in section 3.6 of Chapter III. However, output encoding techniques suggested by researchers have not been fully investigated and compared in terms of the classification accuracy produced. The conventional way of representing the classes in neural networks is to allocate one output node for each land cover type, and assign 1 to the node that corresponds to a particular class and 0 to the nodes that represent other classes. For example, if there are four classes to be classified, a code of 0 1 0 0 is used to represent the class 2 in the network. The advantage of this approach is that the values of the output nodes calculated for test data can be interpreted as *a posterior* probabilities of membership since they are in the 0-1 range. It is argued that the use of such encoding for outputs may be problematic as the values of 0 and 1 can only be produced by the network when the weights have infinitively large positive and negative values. The standard sigmoid activation function can only yield these

extreme values for the inputs of  $\pm\infty$ . Therefore, ranges inside these extreme values are recommended. Some of these recommendations are as follows:

0.01 0.99 0.01 0.01	Fitch <i>et al.</i> (1991) and Thrackrah <i>et al.</i> (1999)
0.10 0.90 0.10 0.10	Paola and Schowengerdt (1995a, 1997), Pierce <i>et al.</i> (1994) and Skidmore <i>et al.</i> (1997)
0.003 0.99 0.003 0.003	Gong (1996)

It is claimed by Gong (1996) that using [0.003 0.99 0.003 0.003] encoding approach as opposed to [0 1 0 0] encoding also speeds up the training process. The main drawback of the truncated ranges is that the results can no longer be interpreted as *a posterior* probabilities (percentage values) for the class membership. However, they can be easily transformed to the 0-1 range.

In order to investigate whether the network performance is significantly affected by the encoding approach employed in network training, the encoding approaches described above are employed in two classification problems. For the experiments the learning rate was set to 0.2 and the weights were initialised in the range [-0.3, 0.3]. A network structure of 8-15-7 was found to be sufficient to learn the characteristics of both problems. During the training, networks were saved at 1,000 iteration intervals so as to observe the change in the classification accuracy and eliminate the bias that could arise when interpreting the results produced at a certain stage of the process. The results for the first and the second datasets are shown in Figures 6.21 and 6.22.

For the first dataset, the [0.1 0.9 0.1] encoding approach performing best at 1,000-iteration displayed a distinct trend, whereas the other approaches yielded similar results. The classification accuracy achieved by the [0.1 0.9 0.1] set decreased sharply, indicating that the network reached to the global minimum of the error earlier compared to other approaches and therefore lost its generalisation capabilities with more training. The other approaches led to networks to reach the global minimum of the error around 8,000 iterations. This indicates that using the encoding approach [0.1 0.9 0.1] may speed up the learning process.

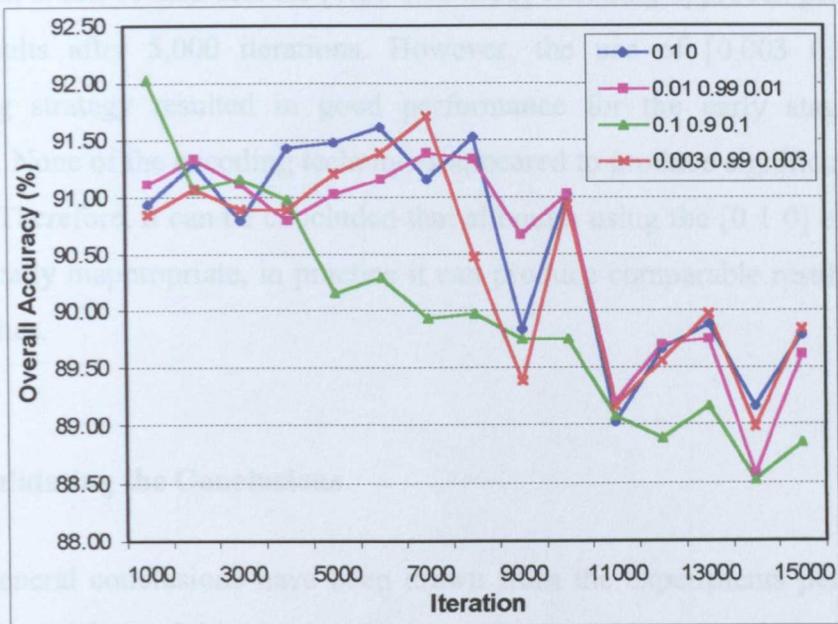


Figure 6.21 The effect of employing different output encoding methods on the classification accuracy for the first dataset.

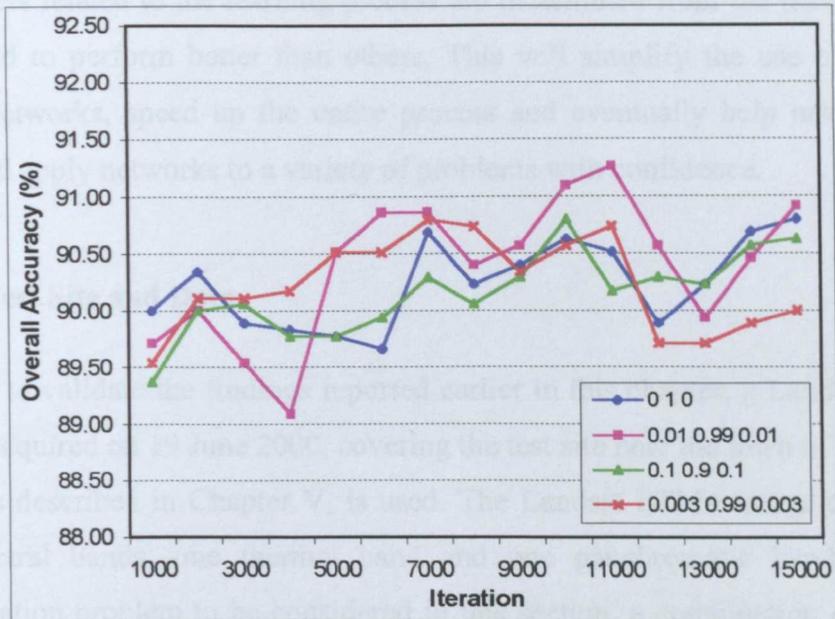


Figure 6.22 The effect of employing different output encoding methods on the classification accuracy for the second dataset.

The results for the second dataset, on the other hand, suggest different conclusions. Firstly, the classification problem appeared to be more difficult, therefore the performance of the networks increases with increased number of

iterations. It can be said that the [0.01 0.99 0.01] encoding approach produced the best results after 5,000 iterations. However, the use of [0.003 0.99 0.003] encoding strategy resulted in good performance for the early stages of the training. None of the encoding techniques appeared to produce significantly better results. Therefore, it can be concluded that although using the [0 1 0] encoding is theoretically inappropriate, in practice it can produce comparable results to other approaches.

## **6.10 Validating the Conclusions**

Some general conclusions have been drawn from the experiments performed in the earlier sections of this chapter. However, it was noticed that there was a need for validation using a new set of data. It is essential that the findings reported should be proved to be valid and producing high-accurate results for other datasets. In the classification of new datasets, network structure and the parameters related to the learning process are determined from the heuristics that are found to perform better than others. This will simplify the use of artificial neural networks, speed up the entire process and eventually help new users to build and apply networks to a variety of problems with confidence.

### **6.10.1 Test Site and Data**

In order to validate the findings reported earlier in this chapter, a Landsat ETM+ image, acquired on 19 June 2000, covering the test site near the town of Littleport, which is described in Chapter V, is used. The Landsat ETM+ image consists of six spectral bands, one thermal band and one panchromatic band. For the classification problem to be considered in this section, a combination of spectral bands (excluding thermal and panchromatic bands) is formed. The test site covers approximately 44.16 km<sup>2</sup> of rich agricultural fields. The classification problem involved the identification of six land cover classes; namely, onion, wheat, sugar beat, potato, lettuce and peas that cover the bulk of the area of interest.

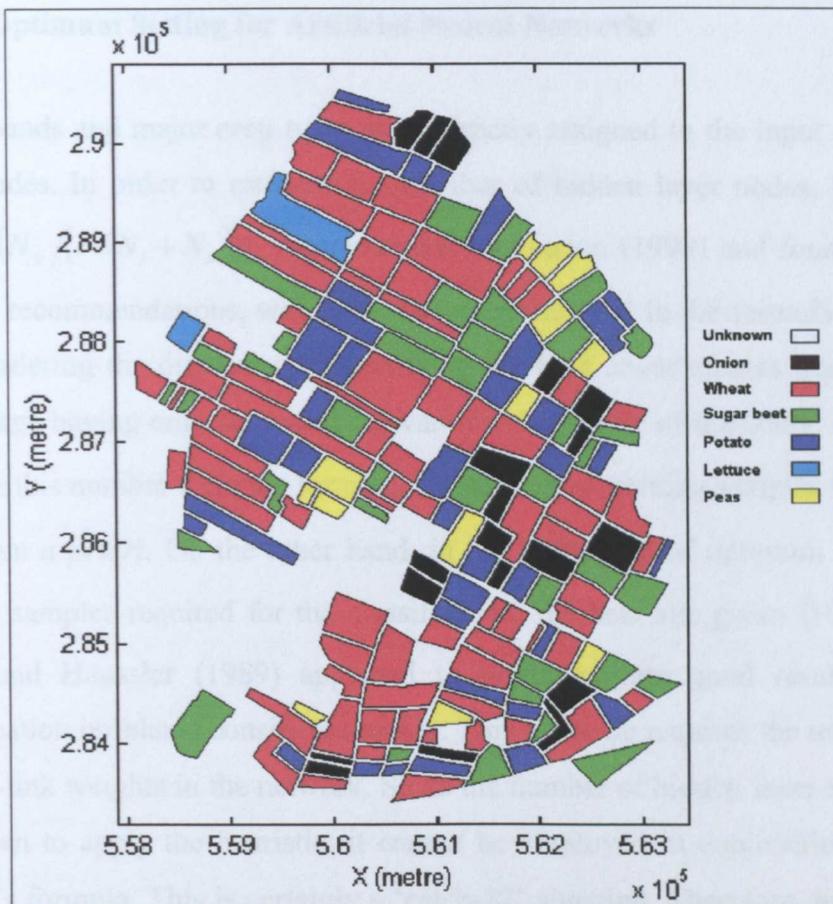


Figure 6.23 Ground reference data for Littleport site for the crop season of 2000.

The image was registered to the Ordnance Survey (OS) of Great Britain's National Grid using the ERDAS Imagine image processing software (version 8.4) by applying a first-order polynomial transformation. The RMSE value estimated for image transformation was less than one pixel. In the resampling process of the co-registration stage, the spatial resolution (i.e. pixel size) of the image was reduced to 20 metres. Ground truth information (Figure 6.23) was collected from the farmers and their representatives by research staff of the School of Geography, Nottingham University. Field boundaries were digitised from 1:25,000 OS maps, published in 1987, and the class labels were later assigned to those fields. In order to disregard the mixed pixels located at the field boundaries, a buffer zone of 20 metres corresponding to the dimensions of a pixel, was constructed and the areas in this zone were labelled as unknown. A 286-pixel by 386-pixel portion of the image covering the area of interest was extracted and used for further stages.

## 6.10.2 Optimum Setting for Artificial Neural Networks

Image bands and major crop types were directly assigned to the input and output layer nodes. In order to estimate the number of hidden layer nodes, the rule of thumb  $(N_p / [r \cdot (N_i + N_o)])$ , recommended by Garson (1998) and found superior to other recommendations, was used. The coefficient 'r' in the formula was set to 10 considering the difficulty of identifying six land cover classes from a single date image having only six bands relevant for the nature of the study. In order to estimate this number from the formula, the number of training samples ( $N_p$ ) must be known *a priori*. On the other hand, in the estimation of optimum number of training samples required for the classification, the heuristic given  $(10 \times N_w)$  by Baum and Haussler (1989) appeared to produce quite good results for the classification problems considered earlier. This heuristic requires the total number of inter-link weights in the network. Since the number of hidden layer nodes must be known to apply the heuristic, it cannot be employed in conjunction with the Garson's formula. This is certainly a 'catch-22' situation. Therefore, the heuristic  $(60 \times N_i \times (N_i + 1))$  suggested by Hush (1998) for estimating the optimal number of training samples is favoured. This heuristic suggests slightly greater numbers than the one proposed by Baum and Haussler (1989).

For setting up the parameters employed the learning process, the initial weight range was set to  $[-0.25, 0.25]$ , as proposed by Gallagher and Downs (1997). It was observed from the experiments reported earlier in this chapter that a learning rate of 0.2 usually performs well in cases where no momentum term is added to the process, as suggested by Bischof *et al.* (1992). Where a momentum term is employed, learning rates of 0.1 and 0.2 performed well together with values of momentum of 0.5 and 0.6. As the results suggest, one of the combinations of these rates should be employed in ANN studies. Another important factor affecting the performance of ANNs is the stopping criterion, which is used to terminate the learning process. It was found that the use of a validation dataset for this purpose results in acceptably good classification performance and shorter training period. Therefore, 75 samples for each output class (450 samples in total) were selected randomly from the areas of the image where ground reference data are available.

Determination of this number is made on the grounds of experience. However, around 100 samples for each class or a 80:20 rate for the training and validation datasets can be used, depending on the availability of the ground reference data. It should be noted that the use of more validation samples can provide more accurate testing with the drawback of a slower training process. Finally, for the encoding of the output classes the conventional way, [0 1 0] form, is employed since no superiority was observed for the use of other recommendations. All these considerations were implemented to build the network and define the learning process. The configuration of the network and the learning process using the above considerations is shown in Table 6.6.

Table 6.6 Optimum setting of network structure and learning parameters.

<b>Parameters</b>	<b>Choice</b>
Number of input nodes	6
Number of output nodes	6
Number of hidden nodes	21
Initial weight range	[-0.25, 0.25]
Learning rate without momentum	0.2
Learning rate with momentum	0.1 or 0.2
Momentum term	0.5 or 0.6
Stopping criterion for learning process	Validation set of 450 samples
Number of training samples	2,520
Output encoding scheme	[0 1 0]

A network structure of 6-21-6 was formed and trained with the parameters listed in Table 6.6 using a training set of 2,520 pixels and a validation set of 450 pixels, both of which are randomly selected. The training process was repeated five times to apply the combinations of the learning rate and momentum term. During these operations, all the parameters except for the learning rate and the momentum term were kept constant. Trained networks were saved and their performances were evaluated using a test dataset including 4,000 randomly selected pixels. The results for the combinations are shown in Table 6.7.

From the experiments performed, the longest training period (4,300 iterations) was required for the one employing learning rate without the momentum term, which is in fact an expected behaviour. The shortest training processes were, however, attained from the combinations of 0.1-0.6 and 0.2-0.5 where the first value shows the learning rate and the second indicates the momentum term. As can be seen from Table 6.7, the lowest MSE values for training and validation sets were achieved by the configuration employing learning rate of 0.2 without momentum. This also led the network to produce the highest classification accuracy (overall accuracy of 85.78% and Kappa coefficient of 0.8237). Other combinations also produced similar results. The variation in overall accuracy was 1%, which can be considered insignificant.

Table 6.7 ANN results obtained from the configurations given in Table 6.6. Abbreviations of ‘lr’ and ‘m’ are used to represent the learning rate and momentum term, respectively.

	lr: 0.2	lr: 0.1 m: 0.5	lr: 0.1 m: 0.6	lr: 0.2 m: 0.5	lr: 0.2 m: 0.6
Iteration	4300	2660	1820	1820	1940
MSE on tr.	0.210073	0.223087	0.225862	0.225488	0.227824
MSE on val.	0.187221	0.195389	0.199553	0.194959	0.193532
Overall (%)	85.78	85.28	84.98	84.85	84.78
Kappa	0.8237	0.8175	0.8138	0.8122	0.8115

In order to verify the results obtained from neural networks and make sound comparisons, the same training and test datasets were employed in the classification process using the maximum likelihood classifier, which is the most sophisticated statistical classification technique. This technique produced classification results with overall accuracy of 81.77% and Kappa coefficient of 0.7740. It is clear that artificial neural networks could identify the crops with around 4% more accuracy than the maximum likelihood classifier. Such a difference can be regarded as considerable.

It is likely that a slightly better classification performance could be produced by means of making some adjustments to the network structure and the parameters used in neural network classification. However, the results produced in this experiment are promising since the aim here is to identify a set of rules that can provide high-accuracy classification results in most cases without the need to consider the many factors involved in the determination of the network structure and the learning parameters.

### 6.10.3 A Worst-Case Scenario for ANN Design and Use

During the experiments described here, a number of heuristics and personal choices reported by researchers were found ineffective and misleading. When they are used together, performance of the networks can be affected severely, producing even worse results than simple statistical classifiers.

For the determination of number of hidden layer nodes, Wang (1994b) used the formula  $2 \times N_i / 3$ , which suggests very small numbers. The initial weight range of  $[0, 1]$  used by Ardö *et al.* (1997) appears to produce the worst in the experiments. A learning rate of 0.8 without the use of a momentum term, suggested by Staufer and Fischer (1997) showed an abrupt behaviour. Hara *et al.* (1994) use the combination of 0.5 and 0.9 for the learning rate and momentum. This combination produced the worst results in the experiments. The training process can be stopped after a pre-defined number of iterations instead of checking the change in the error using a validation dataset. This methodology is extremely simple and potentially troublesome for the learning process, as it is prediction-based. For the determination of the number of training samples required for appropriate learning, the heuristic  $(5 \times N_w)$ , proposed by Klimasauskas (1993), was found to suggest too few samples, which are not sufficient for the network to learn the characteristics of the data. Finally, as there was no clear indication for the worst performing output encoding scheme recommended, the conventional scheme was employed. All these considerations were also put into practice, resulting in the configuration shown in Table 6.8.

Table 6.8 Configuration of the network and the learning algorithm for the worst-case scenario.

Parameters	Choice
Number of input nodes	6
Number of output nodes	6
Number of hidden nodes	4
Initial weight range	[0, 1]
Learning rate without momentum	0.8
Learning rate with momentum	0.5
Momentum term	0.9
Stopping criterion for learning process	5,000 iterations
Number of training samples	240
Output encoding scheme	[0 1 0]

A 6-4-6 network structure was trained with the parameters listed in Table 6.8. Training was performed with the training data 5,000 times in an iterative way using the backpropagation learning algorithm. The training process was carried out for the two learning rate and momentum combinations (i.e. 0.8 and 0.5-0.9) and the resulting networks were saved. These networks were later tested using 4,000 randomly selected pixels. The results are presented in Table 6.9.

Table 6.9 ANN results obtained for the configurations given in Table 6.8.

	Learning rate: 0.8	Learning rate: 0.5 Momentum: 0.9
Iteration	5000	5000
MSE on tr.	0.255463	0.290819
Overall (%)	75.00	72.52
Kappa	0.6844	0.6576

As can be seen from Table 6.9, the training process could not reach a good solution in terms of MSE and classification accuracy. Overall classification accuracies of 75.00% and 72.52% that are considerably less than those produced by the network using optimum settings. The results are also inferior to the results produced by the maximum likelihood classifier (MLC). This behaviour of ANNs clearly indicates that the classification accuracies that can be achieved through their use can differ significantly, depending on the selection of network structure and parameters related to learning process. However, the MLC method can produce similar results on the condition that there are sufficient numbers of samples to estimate the variance-covariance matrix accurately and that the frequency distributions of the classes are approximately Gaussian.

### **6.11 Summary**

The major issues that have been reported to be of primary importance for the performance of artificial neural networks are investigated in this chapter. Special attention is paid to the components of the network structure (i.e. input, hidden and output layers) and the learning parameters (i.e. initial weight range, the learning rate and the momentum term). The strategies for output encoding and stopping the training process at the most appropriate point are also studied. In addition, the effect of the size of samples employed in the training process is explored. The main purpose of this chapter is to provide both theoretical knowledge available to date by referring to recent studies, and heuristics developed by researchers as a result of their experience. The heuristics are usually combined in tables and their effectiveness has been evaluated using real world datasets. The heuristics found to be superior are then applied to a new dataset to verify the findings.

Although a number of conclusions and observations can be reported from the results of the experiments, they will not be enumerated here in order to avoid repetition. Instead, they will be given in the next chapter, Chapter VII, since this particular chapter is intended to present some guidelines that will be largely extracted from the results given in this chapter. Thus, one of the main objectives

of this study, which is to prepare some guidelines for new users of artificial neural networks, is achieved. It is hoped that such guidelines derived from a vast amount of experiments will be useful and beneficial for a wide variety of artificial neural network applications in remote sensing.

## CHAPTER VII

### CONCLUSIONS

#### 7.1 Introduction

Knowledge of the nature and spatial distribution of land cover types is a prerequisite for many regional to global scale studies. Production of such information is mainly through classification using remotely sensed imagery. Although a variety of statistical classification techniques have been developed and used to identify land cover types, there has been a need for more sophisticated and robust methods due to the restrictions of the statistical approaches, particularly regarding the frequency distribution of the data. Artificial neural networks have great potential in pattern recognition, and have recently been employed in a diversity of applications in the remote sensing field. It should be noted that choices of features and training data have as much influence as the classifier on classification results. An unsophisticated classification technique can give a good solution to a well-specified problem in terms of the scale of the problem relative to spatial resolution, the selection of an appropriate number of classes, the characteristics of the features, and the training data. This study is mainly concerned with the application of neural networks to land cover classification, and concentrates on their behaviour and the impact of individual parameters.

This chapter summarises the results presented in previous chapters, and elaborates the conclusions that are later considered to form guidelines for designing and

using neural networks efficiently. Within the thesis, the issues reported to have significant effect on network performance are thoroughly investigated. The general aim of this research was to understand the behaviour of artificial neural networks and thus make some important suggestions for future work. However, the ultimate goal was to make a contribution towards increasing the popularity of neural networks in the remote sensing field in terms of simplifying their design and application, and ensuring that they consistently produce reliable results. It is believed that they can be thus considered as one of the basic and standard tools in remote sensing studies.

## **7.2 Summary of the Thesis**

The theory of classification and a number of classification methods used to classify remotely sensed images are described and their classification abilities are examined in Chapter II. The calculation and representation of classification accuracy is also discussed, and specific attention is paid to the use of accuracy maps showing the variations in the classification accuracy in spatial domain.

Since this thesis is mainly concerned with the use of artificial neural networks for classification of land cover features, a specific chapter (Chapter III) is devoted to a discussion of the theory of artificial neural networks, specifically the feed-forward neural network model (also known as the multilayer perceptron) that is employed in the analyses performed in this research. In addition, a critical assessment of the problems encountered in their use is carried out. The following chapters concentrate on the investigation of these problems with the aim of understanding the effects of the most important issues involved in the design and use of neural networks.

In order to understand the effects of major factors having significant effect on the performance of neural networks, scientific visualisation techniques, described in Chapter IV, are employed. A variety of techniques that are found useful for this purpose and a number of analyses performed on the training data and the classification results are combined in a toolkit (Appendix A) that is created in the MATLAB software package.

For the investigation of the use of neural networks for high-dimensional image data, a particular chapter (Chapter V) is devoted to feature selection methods. This issue has assumed more importance with the launch of new satellite sensors providing information in large number of spectral bands and the availability of image data from many sources. Landsat TM and ETM, SPOT HRV and SIR-C SAR images are used to study this issue. In the search of most effective bands for a particular problem, genetic algorithms are shown to be one of the most promising methods. It is shown that number of input features can be reduced using feature selection techniques without significantly affecting accuracy. It should be noted that the number of features should be sufficient to correctly represent the problem. The findings regarding feature selection methods are presented in section 5.9 of Chapter V.

The most important factors and issues having impact on network performance outlined in early chapters are examined, and heuristics (or rules of thumb) are presented in Chapter VI. Their effectiveness is evaluated using two real-world datasets. The conclusions derived from the results are tested on an independent classification problem. Satisfactory results are produced in comparison with the results produced both by the maximum likelihood classifier (MLC), and the neural networks designed with the worst-case scenario that is constructed from the worst performing parameters. The improvements achieved by the optimum setting are more than 4% for MLC and 10% for the neural network designed with the worst-case scenario parameters.

### **7.3 Conclusions**

The conclusions reached from the experiments in this study using the two datasets described in Chapter V are generally presented in the conclusions section of each chapter. However, the most important conclusions matching the primary aims of this study are collectively presented here for the convenience of the reader.

- Although transformed measures (transformed divergence and the Jeffries-Matusita distance) produced better results than their counterparts

(divergence and the Bhattacharyya distance) for the solutions attained by the GA, the solutions attained using SFS based on divergence measure yielded the best performances for both datasets.

- Of the feature selection techniques employed, Hotelling's  $T^2$  appears to perform better than the others in terms of the classification accuracy produced. Wilks'  $\Lambda$  criterion, on the other hand, produced largely varying results considering the datasets used.
- The genetic algorithm generally reaches a better solution than the sequential forward selection method in terms of the separability measures considered. However, it should be noted that this does not guarantee more accurate classification results.
- In the use of node pruning methods for input layer nodes, the Noncontributing Units method is found more effective than the Skeletonization method.
- It is observed that a non-linear relationship exists between the number of input nodes and the training time required, contrary to the claims made that this is a linear relationship. The results suggest that using some particular band combinations helps the network better recognise patterns in datasets and therefore speeds up the learning process.
- It is found that there is no significant benefit of employing more inputs than a specific number. It can be concluded that the performance of the network is insensitive to large sizes of the input layer. Unfortunately, there is no way, except for evaluating different sizes of input layer, to determine the critical number, smaller than which causes neural network to lose its power and therefore produce less accurate results.
- Although a large number of heuristics have been recommended in the literature to estimate the number of hidden layer nodes, only a few of them are found to be applicable to the problems considered in this research study. The one suggested by Garson (1998) ( $N_p / [r \cdot (N_i + N_o)]$ ) is found to be superior to other heuristics. One of

the reasons for the superiority of this particular heuristic is that it has a constant defined by the user relating to the noise level in the data and the difficulty of the problem. This introduces a flexibility and robustness to the determination of number of hidden layer nodes.

- Assessment of the effect of the number of hidden layer nodes on the performance shows the extreme sensitivity of neural networks to small network sizes and insensitivity to large network sizes. In other words, whilst classification accuracy stayed almost the same for large networks, a significant reduction in the classification accuracy is noticed when the size of the network is too small for the problem under consideration.
- A number of combinations have been suggested for the learning rate parameter and momentum term to accelerate the learning process and reach the global minimum of the error. However, it is found that some of these suggestions result in failure when they are employed. As a result of an extensive number of investigations, it is concluded that a learning rate of 0.2 where there is no momentum term employed, or a learning rate of 0.1 or 0.2 with a momentum term of 0.5 or 0.6 can lead the networks to produce accurate classification results in most cases.
- Evaluation of six initial weight ranges under different conditions in terms of network size, number of training patterns, and learning rate-momentum values shows that small ranges of initial weights ( $[-0.1, 0.1]$ ,  $[-0.15, 0.15]$  and  $[-0.25, 0.25]$ ) produced better results than large ranges. However, the differences in classification accuracy were usually small. Any of the small ranges can be chosen, but the range  $[-0.25, 0.25]$ , suggested by Gallagher and Downs (1997), is favoured in this research since it usually maintains the level of accuracy under most conditions.
- Heuristics proposed to estimate the optimum number of training samples were compared using two datasets involving the classification of seven land cover classes for both cases. The results show that the heuristics proposed by Baum and Haussler ( $10 \times N_{\#}$ ) and Hush (1998) ( $30$  or  $60 \times N_i \times (N_i + 1)$ ) are good choices.

- It is noticed that ANNs can perform well for small training samples. In the cases considered, the lower bound is appeared to be 975 samples in total, indicated by the heuristic  $(5 \times N_w)$  suggested by Klimasauskas (1993). However, for the same problems a statistical classifier would need at least 1,680 patterns to compute the variance-covariance matrix accurately, according to the heuristic  $(30 \times p)$  given by Mather (1999).
- In the comparison of strategies used to terminate the learning process, it is found that using a validation dataset helped to detect the best stopping point in terms of the classification accuracy produced. With the use of a validation dataset, the training processes were also terminated in the relatively early stages of the learning process.
- Several suggested output encoding strategies were compared to the conventional output encoding scheme [0 1 0] in that 1 is assigned to the node corresponding to a particular class and 0 to the nodes that represent other classes. The results produced were comparable to each other, suggesting no superiority for a particular scheme. Therefore, the conventional scheme [0 1 0] is recommended, although it is theoretically inappropriate due to the impracticality of transfer functions to produce these extreme values. Another reason to favour this scheme is that the results produced using this particular strategy can be interpreted as *a posteriori* probabilities of class membership.
- Close examination of the learning process using animations created from lower dimensional (2 and 3) representations reveals an important fact that neural networks can, in fact, learn the major characteristics of the datasets quite quickly (in about several hundred iterations). After that, they attempt to identify the mixed and atypical pixels.

#### 7.4 Guidelines for the Effective Use of Artificial Neural Networks

The conclusions produced and the experience gained during this study can be used to form a number of guidelines that can greatly facilitate the process of design and use of artificial neural networks. These guidelines are particularly useful for

defining the network structure and configuring the learning algorithm. It should be noted that they are valid for similar datasets and classification problems to those used in this study. Some suggestions can be also made for post-processing to improve the generalisation capabilities of networks. The list of the guidelines is given as follows:

- ◆ *Use feature selection techniques (e.g. Hotelling's  $T^2$  together with genetic algorithm or divergence with sequential forward selection) if there are large numbers of input features available.*
- ◆ *Estimate the number of hidden layer nodes required using the expression  $N_p / [r \cdot (N_i + N_o)]$ .*
- ◆ *Define the number of output layer nodes by considering the nature of the problem and the availability of ground reference data.*
- ◆ *Select training samples randomly, between  $30 \times N_i \times (N_i + 1)$  and  $60 \times N_i \times (N_i + 1)$  in number, depending on the difficulty of the problem under consideration.*
- ◆ *Set the initial weights to a small range (e.g. [-0.15, 0.15] or [-0.25, 0.25]) that has a mean value of 0.*
- ◆ *Set the learning rate to 0.2 for the standard backpropagation algorithm and to either 0.1 or 0.2 for backpropagation with momentum in conjunction with the momentum term of 0.5 or 0.6.*
- ◆ *Employ a validation dataset to terminate the training process. The validation dataset may include around 50-100 samples for each class.*
- ◆ *Use the output encoding scheme of [0 1 0] to represent output classes.*
- ◆ *Use a shuffling mechanism for the learning process to present the inputs to the network in a randomly defined order.*
- ◆ *To improve the generalisation capability of a trained network, employ inter-connection (e.g. Optimum Brain Surgeon) or node pruning methods (e.g. Noncontributing Units).*

## 7.5 Future Work and Recommendations

Based on the research carried out in this study, there is considerable potential for future work in extending the investigations to new datasets, particularly hyperspectral image data, and to other neural network models and learning parameters. In addition, more testing is needed to evaluate the applicability of guidelines to other datasets to be able to make claims about their robustness. It is recommended that the research reported in Chapter V involving feature selection should be extended to hyperspectral datasets, such as CASI (Compact Airborne Spectrographer Imagery) and MODIS (Moderate Resolution Imaging Spectroradiometer) so as to validate the effectiveness of the conclusions reached through multispectral, multitemporal and multisensor datasets obtained from SPOT HRV, Landsat TM and SIR-C SAR satellite images. It could be thus possible to determine the most effective bands for the nature of the output classes attempted to be identified.

In order to improve and extend the investigations reported in Chapter VI, in addition to constant learning rates, the use of adaptive learning rate strategies should be examined and their results should be compared to those produced by their counterparts. Also, the effect of employing different transfer functions, such as the sigmoid and tangent hyperbolic function, in the learning process, which is also reported to have significant effect on neural network performance, needs investigating. However, it should be noted that as a standard selection, the sigmoid activation function is used in all experiments performed in this study. As this study is limited to feed-forward artificial neural networks learning problems with the backpropagation learning algorithm, it could be also beneficial to investigate the effects of the network structure and the learning parameters on other ANN models, including SOM and LVQ, with the aim of deriving some general conclusions that can be used to construct some guidelines for users in the use of these particular network models.

## 7.6 Final Remarks

Artificial neural networks can be used for many investigation purposes. However, the focus in this study is on land cover classification using satellite image data. It is the case that artificial neural networks are more robust than conventional statistical classifiers. Therefore, they are of great importance for remote sensing studies. The research reported here aims to strengthen their importance by providing extensive analyses on the effect of network structure and learning parameters, as well as by presenting new ways to visualise and understand the data and the results produced by neural networks. It is believed that neural networks will continue to maintain their importance and validity for pattern recognition problems in the future despite the advent of new and sophisticated methods, such as decision trees and genetic algorithms. A current trend is to incorporate fuzziness into the classification procedure with the aim of producing more reliable and accurate information. It should be also stated that as their nature, the implementation of artificial neural networks requires parallel processing. If they are implemented on a massively parallel computing system, the computational cost, which is recognised as one of the biggest drawbacks of the technique, could be reduced significantly.

It is hoped that this study makes some contributions to the understanding of the role of neural networks in remote sensing studies, and will be beneficial for their design and use. By applying the suggestions made in this research, more accurate classification results and shorter training times can be produced. Finally, by evaluating the impact of the choices of network architecture, of initial weight values, and of parameter values, it is hoped that users of artificial neural networks will have a clearer idea of the way these networks function, so that they are no longer considered to be 'black-boxes'.

## REFERENCES

- Abuelgasim, A. A., Gopal, S., Irons, J. R. and Strahler, A. H. (1996) Classification of ASAS multiangle and multispectral measurements using artificial neural networks. *Remote Sensing of Environment*, **57**, 79-87.
- Aha, D. W. and Bankert, R. L. (1995) A comparative evaluation of sequential feature selection algorithms. *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, USA, 1-7.
- Ahmad, S. and Tesauro, G. (1989) Scaling and generalization in neural networks: a case study. In *Advances in Neural Information Processing Systems I*, edited by D. S. Touretzky. (San Mateo, CA: Morgan Kaufmann), 160-168.
- Alpaydin, E. (1991) GAL: Networks that grow when they learn and shrink when they forget. *International Journal of Pattern Recognition and Artificial Intelligence*, **8**, 391-414.
- Alpaydin, E. and Gurgen, F. (1998) Comparison of statistical and neural classifiers and their applications to optical character recognition and speech classification. *Neural Network Systems Techniques and Applications*, **5**, 61-88.
- Alt, M. (1990) *Exploring Hyperspace: A Non-mathematical Explanation of Multivariate Analysis*. (London: McGraw-Hill).
- Altun, H. (1998) Evaluation of neural learning in a MLP NN for an acoustic-to-articulatory mapping problem using different training pattern vector characteristics. PhD Thesis, The University of Nottingham, Nottingham, UK.
- Andrews, D. F. (1972) Plots of high-dimensional data. *Biometrics*, **28**, 125-136.
- Ardö, J., Pilesjö, P. and Skidmore, A. (1997) Neural networks, multitemporal Landsat Thematic Mapper data and topographic data to classify forest damages in the Czech Republic. *Canadian Journal of Remote Sensing*, **23**, 217-229.
- Atkinson, P. M. and Tatnall, A. R. L. (1997) Neural networks in remote sensing. *International Journal of Remote Sensing*, **18**, 699-709.
- Augusteijn, M. F., Clemens, L. E. and Shaw, K. A. (1995) Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier. *IEEE Transactions on Geoscience and Remote Sensing*, **33**, 616-625.
- Azimi-Sadjadi, M. R., Ghaloum, S. and Zoughi, R. (1993) Terrain classification in SAR images using principal component analysis and neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, **31**, 511-515.

- Bailey, T. C. and Gatrell, A. C. (1995)** *Interactive Spatial Data Analysis*. (Essex: Longman).
- Bateson, A. and Curtiss, B. (1996)** A method for manual endmember selection and spectral unmixing. *Remote Sensing of Environment*, **55**, 229-243.
- Baum, E. B. and Haussler, D. (1989)** What size net gives valid generalization? In *Advances in Neural Information Processing Systems I*, edited by D. S. Touretzky. (San Mateo, CA: Morgan Kaufmann), 81-90.
- Bebis, G. and Georgiopoulos, M. (1994)** Optimal feed-forward neural network architectures. *IEEE Potentials*, October, 27-31.
- Bebis, G., Georgiopoulos, M. and Kasparis, T. (1997)** Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing*, **17**, 167-194.
- Becker, S. and Le Cun, Y. (1988)** Improving the convergence of back-propagation learning with second order methods. *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA, USA, (London: Morgan Kaufmann) 29-37.
- Belward, A. S. and De Hoyos, A. (1987)** A comparison of supervised maximum likelihood and decision tree classification for crop cover estimation from multitemporal LANDSAT MSS data. *International Journal of Remote Sensing*, **8**, 229-235.
- Benediktsson, J. A., Swain, P. H. and Ersoy, O. K. (1990)** Neural network approaches versus statistical methods in classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, **28**, 540-551.
- Bischof, H., Schneider, W. and Pinz, A. J. (1992)** Multispectral classification of Landsat-images using neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, **30**, 482-489.
- Bishop, C. M. (1995)** *Neural Networks for Pattern Recognition*. (Oxford: Clarendon Press).
- Blamire, P. A. (1994)** An investigation into the identification and classification of urban areas from remotely sensed satellite data using neural networks. MSc Dissertation, University of Edinburgh, Edinburgh, UK.
- Blamire, P. A. (1996)** The influence of relative sample size in training artificial neural networks. *International Journal of Remote Sensing*, **17**, 223-230.
- Blamire, P. A. and Mineter, M. J. (1995)** On the identification of urban areas from Landsat TM data using artificial neural networks. *Proceedings of the 21st Annual*

*Conference of The Remote Sensing Society*, Southampton, UK, (Nottingham: The Remote Sensing Society) 50-57.

**Boers, E. J. W. and Kuiper, H. (1992)** Biological metaphors and the design of modular artificial neural networks. MSc Dissertation, Leiden University, Leiden, The Netherlands.

**Bostock, R. T. J. (1994)** The impact of architecture on the performance of artificial neural networks. PhD Thesis, The University of Aston, Birmingham, UK.

**Bruzzone, L., Conese, C., Maselli, F. and Roli, F. (1997)** Multisource classification of complex rural areas by statistical and neural-network approaches. *Photogrammetric Engineering and Remote Sensing*, **63**, 523-533.

**Campbell, J. B. (1987)** *Introduction to Remote Sensing*. (London: The Guilford Press).

**Castellano, G., Fanelli, A. M. and Pelillo, M. (1997)** An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, **8**, 519-531.

**Chatfield, C. and Collins, A. J. (1980)** *Introduction to Multivariate Analysis*. (London: Chapman and Hall).

**Chen, K. S., Huang, W. P., Tsay, D. H. and Amar, F. (1996)** Classification of multifrequency polarimetric SAR imagery using a dynamic learning neural network. *IEEE Transactions on Geoscience and Remote Sensing*, **34**, 814-820.

**Chen, K. S., Kao, W. L. and Tzeng, Y. C. (1995a)** Retrieval of surface parameters using dynamic learning neural network. *International Journal of Remote Sensing*, **16**, 801-809.

**Chen, K. S., Tzeng, Y. C., Chen, C. F. and Kao, W. L. (1995b)** Land-cover classification of multispectral imagery using a dynamic learning neural network. *Photogrammetric Engineering and Remote Sensing*, **61**, 403-408.

**Chernoff, H. (1973)** The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, **68**, 361-368.

**Chester, D. L. (1990)** Why two hidden layers are better than one. *Proceedings of the International Joint Conference on Neural Networks*, Washington, USA, 265-268.

**Chipperfield, A. (1997)** Introduction to genetic algorithms. In *Genetic Algorithms in Engineering Systems*, edited by A. M. S. Zalzal and P. J. Fleming. (London: The Institution of Electrical Engineers), 1-45.

- Civco, D. L. and Waug, Y. (1994)** Classification of multispectral, multitemporal, multisource spatial data using artificial neural networks. *Proceedings of 1994 Annual ASPRS/ACSM Convention*, Reno, NV. USA, 123-133.
- Clark, C. and Canas, A. (1995)** Spectral identification by artificial neural network and genetic algorithm. *International Journal of Remote Sensing*, **16**, 2255-2275.
- Cole, J. P. and King, C. A. M. (1968)** *Quantitative Geography: Techniques and Theories in Geography*. (London: John Wiley & Sons).
- Comon, P. (1994)** Independent component analysis, a new concept? *Signal Processing*, **36**, 287-314.
- Congalton, R. G. (1991)** A review of assessing the accuracy of classifications of remotely sensed data. *Remote Sensing of Environment*, **37**, 35-46.
- Congalton, R. G. and Green, K. (1999)** *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*. (London: Lewis Publishers).
- Cybenko, G. (1989)** Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303-314.
- Davis, L. (1987)** *Genetic Algorithms and Simulated Annealing*. (London: Morgan Kaufmann).
- Dawson, M. S., Amar, F., Rawat, V., Fung, A. K. and Manry, M. T. (1994)** Classification of remote sensing data using fast learning neural networks and topology selection algorithms. *Proceedings of the IEEE Geoscience and Remote Sensing Symposium (IGARSS'94)*, Pasadena, CA, USA, (Piscataway, NJ: IEEE) (CD-ROM).
- De Backer, S., Naud, A. and Scheunders, P. (1998)** Non-linear dimensionality reduction techniques for unsupervised feature extraction. *Pattern Recognition Letters*, **19**, 711-720.
- de Castro, L. N., Iyoda, E. M., Von Zuben, F. J. and Gudwin, R. R. (1998)** Feedforward neural network initialization: an evolutionary approach. *Proceedings of the 5th Brazilian Symposium on Neural Networks*, Belo Horizonte, Brazil, (Piscataway, NJ: IEEE Computer Society Press) 43-48.
- Denoeux, T. and Lengellé, R. (1993)** Initializing back propagation networks with prototypes. *Neural Networks*, **6**, 351-363.
- Dickson, S., Thomas, B. T. and Goddard, P. (1997)** Using neural networks to automatically detect brain tumours in MR images. *International Journal of Neural Systems*, **8**, 91-99.

- Dreyer, P. (1993)** Classification of land cover using optimized neural nets on SPOT data. *Photogrammetric Engineering and Remote Sensing*, **59**, 617-621.
- Duguay, C. R. and Peddle, D. R. (1996)** Comparison of evidential reasoning and neural network approaches in a multi-source classification of Alpine Tundra vegetation. *Canadian Journal of Remote Sensing*, **22**, 433-440.
- Dutra, L. V. and Huber, R. (1999)** Feature extraction and selection for ERS-1/2 InSAR classification. *International Journal of Remote Sensing*, **20**, 993-1016.
- Eaton, H. A. C. and Olivier, T. L. (1992)** Learning coefficient dependence on training set size. *Neural Networks*, **5**, 283-288.
- Eberhart, R. C. and Dobbins, R. W. (1990)** *Neural Network PC Tools - A Practical Guide*. (London: Academic Press, Inc.).
- Erwin, E., Obermayer, K. and Schulten, K. (1992a)** Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, **67**, 47-55.
- Erwin, E., Obermayer, K. and Schulten, K. (1992b)** Self-organizing maps: stationary states, metastability and convergence rate. *Biological Cybernetics*, **67**, 35-45.
- Estes, J. E., Sailer, C. and Tinney, L. R. (1986)** Applications of artificial intelligence techniques to remote sensing. *The Professional Geographer*, **38**, 133-141.
- Everitt, B. S. (1978)** *Graphical Techniques for Multivariate Data*. (London: Heinemann Educational Books).
- Everitt, B. S. and Dunn, G. (1991)** *Applied multivariate data analysis*. (London: Edward Arnold).
- Everitt, B. S. and Nicholls, P. (1975)** Visual techniques for representing multivariate data. *The Statistician*, **24**, 37-49.
- Fahlman, S. E. (1988)** An empirical study of learning speed in back-propagation networks. Technical Report: *CMU-CS-88-162*, Computer Science Department, Carnegie-Mellon University, Pittsburgh, USA.
- Fahlman, S. E. and Lebiere, C. (1990)** The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky. (San Mateo, CA: Morgan Kaufmann), 524-532.
- Fitch, J. P., Lehman, S. K., Dowla, F. U., Lu, S. Y., Johansson, E. M. and Goodman, D. M. (1991)** Ship wake-detection procedure using conjugate gradient

- trained artificial neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, **29**, 718-726.
- Foody, G. M. (1995)** Using prior knowledge in artificial neural network classification with a minimal training set. *International Journal of Remote Sensing*, **16**, 301-312.
- Foody, G. M. (1996)** Approaches for the production and evaluation of fuzzy land cover classifications from remotely-sensed data. *International Journal of Remote Sensing*, **17**, 1317-1340.
- Foody, G. M. (1997)** Land cover mapping from remotely sensed data with a neural network: accommodating fuzziness. In *Nerocomputation in Remote Sensing Data Analysis*, edited by I. Kanellopoulos, G. G. Wilkinson, F. Roli and J. Austin. (Berlin: Springer), 28-37.
- Foody, G. M. (1999)** Image classification with a neural network: from completely-crisp to fully-fuzzy situations. In *Advances in Remote Sensing and GIS Analysis*, edited by P. M. Atkinson and N. J. Tate. (New York: John Wiley & Sons), 17-38.
- Foody, G. M. and Arora, M. K. (1997)** An evaluation of some factors affecting the accuracy of classification by an artificial neural network. *International Journal of Remote Sensing*, **18**, 799-810.
- Foody, G. M. and Curran, P. J. (1994)** Scale and environmental remote sensing. In *Environmental Remote Sensing from Regional to Global Scales*, edited by G. M. Foody and P. J. Curran. (London: John Wiley & Sons), 223-232.
- Foody, G. M., Lucas, R. M., Curran, P. J. and Honzak, M. (1996)** Estimation of the areal extent of land cover classes that only occur at a sub-pixel level. *Canadian Journal of Remote Sensing*, **22**, 428-432.
- Fu, K. S. (1982)** Application of pattern recognition to remote sensing. In *Applications of Pattern Recognition*, edited by K. S. Fu. (Boca Raton, FL: CRC Press), Chapter 4.
- Fu, L. M. (1994)** *Neural Networks in Computer Intelligence*. (London: McGraw-Hill).
- Fung, T. and LeDrew, E. (1987)** Application of principal component analysis to change detection. *Photogrammetric Engineering and Remote Sensing*, **53**, 1649-1658.
- Gallagher, M. and Downs, T. (1997)** Visualisation of learning in neural networks using principal component analysis. *Proceedings of International Conference on Computational Intelligence and Multimedia Applications*, edited by B. Varma and X. Yao, Gold Coast, Australia, 327-331.
- Gallant, S. I. (1993)** *Neural Network Learning and Expert Systems*. (London: The MIT Press).

- Garson, G. D. (1998)** *Neural Networks: An Introductory Guide for Social Scientists*. (London: SAGE Publications).
- Giles, C. L., Lawrence, S. and Tsoi, A. C. (1997)** Rule inference for financial prediction using recurrent neural networks. *Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, Piscataway, NJ, USA, 253-259.
- Gnanadesikan, R. (1977)** *Methods for Statistical Data Analysis of Multivariate Observations*. (New York: Wiley).
- Gong, P. (1996)** Integrated analysis of spatial data from multiple sources: using evidential reasoning and artificial neural network techniques for geological mapping. *Photogrammetric Engineering and Remote Sensing*, **62**, 513-523.
- Gong, P., Pu, R. and Chen, J. (1996)** Mapping ecological land systems and classification uncertainties from digital elevation and forest-cover data using neural networks. *Photogrammetric Engineering and Remote Sensing*, **62**, 1249-1260.
- Goodenough, D. G., Narendra, P. M. and O'Neill, K. (1978)** Feature subset selection in remote sensing. *Canadian Journal of Remote Sensing*, **4**, 143-148.
- Gopal, S. and Woodcock, C. (1996)** Remote sensing of forest change using artificial neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, **34**, 398-403.
- Gordon, A. D. (1981)** *Classification*. (London: Chapman & Hall).
- Grimm, G. and Yarnold, P. R. (1997)** *Reading and Understanding Multivariate Statistics*. (Washington: The American Psychological Association).
- Gurney, C. M. and Townshend, J. R. G. (1983)** The use of contextual information in the classification of remotely sensed data. *Photogrammetric Engineering and Remote Sensing*, **49**, 55-64.
- Hand, D. J. (1981)** Branch and bound in statistical data analysis. *The Statistician*, **30**, 1-13.
- Hand, D. J. (1997)** *Construction and Assessment of Classification Rules*. (New York: John Wiley & Sons).
- Hara, Y., Atkins, R. G., Yueh, S. H., Shin, R. T. and Kong, J. A. (1994)** Application of neural networks to radar image classification. *IEEE Transactions on Geoscience and Remote Sensing*, **32**, 100-109.
- Haralick, R. M. (1982)** Image texture survey. In *Handbook of Statistics*, edited by P. R. Krishnaiah and L. N. Kanal. (Oxford: North-Holland Publishing Company), 399-415.

- Haralick, R. M., Shanmugam, K. and Dinstein, I. (1973)** Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, **3**, 610-621.
- Hartigan, J. A. (1975)** *Clustering Algorithms*. (New York: Wiley).
- Harvey, D. (1969)** *Explanation in Geography*. (London: Edward Arnold).
- Hassibi, B. and Stork, D. G. (1993)** Second order derivatives for network pruning: optimal brain surgeon. In *Advances in Neural Information Processing Systems 5*, edited by S. J. Hanson, J. D. Cowan and C. L. Giles. (San Mateo, CA: Morgan Kaufmann), 164-171.
- Haykin, S. (1999)** *Neural Networks: A Comprehensive Foundation*. Second Edition. (New Jersey: Prentice Hall).
- Hecht-Nielsen, R. (1987)** Kolmogorov's mapping neural network existence theorem. *Proceedings of the First IEEE International Conference on Neural Networks*, San Diego, CA, USA, 11-14.
- Heermann, P. D. and Khazenie, N. (1992)** Classification of multispectral remote sensing data using a back-propagation neural network. *IEEE Transactions on Geoscience and Remote Sensing*, **30**, 81-88.
- Hennebert, J., Hasler, M. and Dedieu, H. (1994)** Neural networks in speech recognition. *MicroComputer School 94*, Sedmihorky, Czech Republic, 23-40.
- Hepner, G. F., Logan, T., Ritter, N. and Bryant, N. (1990)** Artificial neural network classification using a minimal training set: comparison to conventional supervised classification. *Photogrammetric Engineering and Remote Sensing*, **56**, 469-473.
- Hewitson, B. C. and Crane, R. G. (1994)** *Neural Nets: Applications in Geography*. (London: Kluwer Academic Publishers).
- Hirosawa, Y., Marsh, S. E. and Kliman, D. H. (1996)** Application of standardised principal component analysis to land-cover characterisation using multitemporal AVHRR data. *Remote Sensing of Environment*, **58**, 267-281.
- Hirose, Y., Yamashita, K. and Hijiya, S. (1991)** Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, **4**, 61-66.
- Holland, J. H. (1992)** Genetic algorithms. *Scientific American*, **267**, 44-50.
- Huang, S. C. and Huang, Y. F. (1991)** Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks*, **2**, 47-55.

- Hughes, G. F. (1968)** On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, **14**, 55-63.
- Hush, D. R. (1989)** Classification with neural networks: a performance analysis. *Proceedings of the IEEE International Conference on Systems Engineering*, Dayton, Ohio, USA, 277-280.
- Hush, D. R., Horne, B. and Salas, J. M. (1992)** Error surfaces for multilayer perceptrons. *IEEE Transactions on Systems, Man and Cybernetics*, **22**, 1152-1161.
- Hush, D. R. and Horne, B. G. (1993)** Progress in supervised neural networks. *IEEE Signal Processing Magazine*, January, 8-39.
- Hutchinson, C. F. (1982)** Techniques for combining Landsat and ancillary data for digital classification improvement. *Photogrammetric Engineering and Remote Sensing*, **48**, 123-130.
- Huurneman, G., Gens, R. and Broekema, L. (1996)** Thematic information extraction in a neural network classification of multi-sensor data including microwave phase information. *International Archives of Photogrammetry and Remote Sensing*, **XXXI**, 170-175.
- Jacobs, R. A. (1988)** Increased rates of convergence through learning rate adaptation. *Neural Networks*, **1**, 295-308.
- Jain, A. and Zongker, D. (1997)** Feature selection: evaluation, application and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 153-158.
- James, M. (1985)** *Classification Algorithms*. (London: Collins).
- Jensen, J. R. (1996)** *Introductory Digital Image Processing - A Remote Sensing Perspective*. (London: Prentice Hall).
- Jimenez, L. and Landgrebe, D. (1998)** Supervised classification in high-dimensional space: geometrical, statistical and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man and Cybernetics*, **28**, 39-44.
- Joost, M. and Schiffmann, W. (1998)** Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, **6**, 117-126.
- Kailath, T. (1967)** The Divergence and Bhattacharyya Distance measures in signal selection. *IEEE Transactions on Communication Technology*, **15**, 52-60.
- Kanellopoulos, I., Varfis, A., Wilkinson, G. G. and Mégier, J. (1992)** Land-cover discrimination in SPOT HRV imagery using an artificial neural network – a 20-class experiment. *International Journal of Remote Sensing*, **13**, 917-924.

- Kanellopoulos, I. and Wilkinson, G. G. (1997)** Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, **18**, 711-725.
- Karnin, E. D. (1990)** A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, **1**, 239-242.
- Kaski, S. (1997)** Data exploration using self-organizing maps. *Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82, Espoo, Finland*, Published by the Finnish Academy of Technology.
- Kaski, S., Kangas, J. and Kohonen, T. (1998)** Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, **1**, 102-350.
- Kavzoglu, T. (1999)** Determining optimum structure for artificial neural networks. *Proceedings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society*, Cardiff, UK, 675-682.
- Kavzoglu, T. and Mather, P. M. (1999)** Pruning artificial neural networks: an example using land cover classification of multi-sensor images. *International Journal of Remote Sensing*, **20**, 2787-2803.
- Kemsley, D. H., Martinez, T. R. and Campbell, D. M. (1992)** A survey of neural network research and fielded applications. *International Journal of Neural Networks: Research and Applications*, **2**, 123-133.
- Klimasauskas, C. C. (1993)** Applying neural networks. In *Neural Networks in Finance and Investing*, edited by R. R. Trippi and E. Turban. (Cambridge: Probus), 47-72.
- Kolen, J. F. and Pollack, J. B. (1990)** Back propagation is sensitive to initial conditions. *Complex Systems*, **4**, 269-280.
- Kramer, M. A. (1991)** Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers Journal*, **37**, 233-243.
- Kröse, B. and Van Der Smagt, P. (1996)** *An Introduction to Neural Networks*. (Amsterdam: University of Amsterdam).
- Kuscu, I. and Thornton, C. (1994)** Design of artificial neural networks using genetic algorithms: review and prospect. Research Paper, *School of Cognitive and Computing Sciences CSRP 319*, University of Sussex, UK.
- Landgrebe, D. (2000)** Information extraction principles and methods for multispectral and hyperspectral image data. In *Information Processing for Remote Sensing*, edited by C. H. Chen. (New Jersey USA: World Scientific Publishing Co.), Chapter 1.

- Lawrence, S., Giles, C. L. and Tsoi, A. C. (1996)** What size neural network gives optimal generalization? convergence properties of backpropagation. Technical Report: *UMIA CS-TR-96-22 and CS-TR-3617*, Institute for Advanced Computer Studies, University of Maryland, USA.
- Le Cun, Y. (1993)** *Efficient learning and second order methods*, Tutorial presented at Neural Information Processing Systems 5 (San Mateo, CA: Morgan Kaufmann).
- Le Cun, Y., Denker, J. S. and Solla, S. A. (1990)** Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretsky. (San Mateo, CA: Morgan Kaufmann), 598-605.
- Leray, P. and Gallinari, P. (1998)** Feature selection with neural networks. *LIP6 Research Reports*, University of Paris, France (<http://www.lip6.fr/reports/lip6.1998.012.html>).
- Levin, A. U., Leen, T. K. and Moody, J. E. (1994)** Fast pruning using principal components. In *Advances in Neural Information Processing 6*, edited by J. Cowan, G. Tesauro and J. Alspector. (San Mateo, CA: Morgan Kaufmann), 35-42.
- Levin, S. A. (1992)** The problem of pattern and scale in ecology. *Ecology*, **73**, 1943-1967.
- Lillesand, T. M. and Kiefer, R. W. (1994)** *Remote Sensing and Image Interpretation*. (New York: John Wiley & Sons).
- Lippmann, R. P. (1987)** An introduction to computing with neural nets. *IEEE ASSP Magazine*, April, 4-22.
- Lo, Z., Yu, Y. and Bavarian, B. (1993)** Analysis of the convergence properties of topology preserving neural networks. *IEEE Transactions on Neural Networks*, **4**, 207-220.
- Lunetta, R. S., Congalton, R. G., Fenstermaker, L. K., Jensen, J. R., McGwire, K. C. and Tinney, L. R. (1991)** Remote sensing and geographical information system data integration: error sources and research issues. *Photogrammetric Engineering and Remote Sensing*, **57**, 677-687.
- Man, K. F., Tang, K. S. and Kwong, S. (1999)** *Genetic Algorithms*. (London: Springer-Verlag).
- Mao, J. and Jain, A. K. (1995)** Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, **6**, 296-317.
- Mao, J., Mohiuddin, K. and Jain, A. K. (1994)** Parsimonious network design and feature selection through node pruning. *Proceedings of 12th International Conference on Pattern Recognition*, Jerusalem, Israel, 622-624.

- Marceau, D. J. (1999)** The scale issue in the social and natural sciences. *Canadian Journal of Remote Sensing*, **25**, 347-356.
- Marceau, D. J. and Hay, G. J. (1999)** Remote sensing contributions to the scale issue. *Canadian Journal of Remote Sensing*, **25**, 357-366.
- Mather, P. M. (1999a)** *Computer Processing of Remotely-Sensed Images: An Introduction*. Second Edition. (Chichester: John Wiley).
- Mather, P. M. (1999b)** Land cover classification revisited. In *Advances in Remote Sensing and GIS Analysis*, edited by P. M. Atkinson and N. J. Tate. (New York: John Wiley & Sons), 7-16.
- Mather, P. M., Tso, B. and Koch, M. (1998)** An evaluation of Landsat TM spectral data and SAR-derived textural information for lithological discrimination in the Red Sea Hills, Sudan. *International Journal of Remote Sensing*, **19**, 587-604.
- Matthews, P. and McWhirter, N. D. (1995)** *The New Guinness Book of Records 1996*. (Middlesex: Guinness Publishing Ltd).
- Mausel, P. W., Kramber, W. J. and Lee, J. K. (1990)** Optimum band selection for supervised classification of multispectral data. *Photogrammetric Engineering and Remote Sensing*, **56**, 55-60.
- Messer, K. and Kittler, J. (1997)** A comparison of colour texture attributes selected by statistical feature selection and neural network methods. *Pattern Recognition Letters*, **18**, 1241-1246.
- Messer, K. and Kittler, J. (1998)** Choosing an optimal neural network size to aid search through a large image database. *Proceedings of the Ninth British Machine Vision Conference (BMVC98)*, University of Southampton, UK, 235-244.
- Miller, J. J. and Wegman, E. J. (1991)** Construction of line densities for Parallel Coordinate Plots. In *Computing and Graphics in Statistics*, edited by A. Buja and P. Tukey. (New York: Springer-Verlag), 107-123.
- Monostori, L. and Barschdorff, D. (1992)** Artificial neural networks in intelligent manufacturing. *Robotics and Computer-Integrated Manufacturing*, **9**, 421-437.
- Moreira, M. and Fiesler, E. (1995)** Neural networks with adaptive learning rate and momentum term. IDIAP Technical Report: 95-04, available at <ftp://ftp.idiap.ch/pub/techreports/95-04.ps.Z>.
- Mozer, M. C. and Smolensky, P. (1989a)** Skeletonization: a technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems I*, edited by D. S. Touretzky. (San Mateo, CA: Morgan Kaufmann), 107-115.

- Mozer, M. C. and Smolensky, P. (1989b)** Using relevance to reduce network size automatically. *Connection Science*, **1**, 3-16.
- Niemann, H. and Weiss, J. (1979)** A fast-converging algorithm for nonlinear mapping of high-dimensional data to a plane. *IEEE Transactions on Computers*, **28**, 142-147.
- Oja, E. (1995a)** PCA, ICA and nonlinear Hebbian learning. *Proceedings of the International Conference on Artificial Neural Networks*, Paris, France, 89-94.
- Oja, E. (1995b)** Principal component analysis. In *The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib. (London: The MIT Press), 753-756.
- Oja, E. and Karhunen, J. (1995)** Signal separation by nonlinear Hebbian learning. In *Computational Intelligence - A Dynamic System Perspective*, edited by M. Palaniswami, Y. A. Hikiouzel, R. MarksII, D. Fogel and T. Fukuda. (New York: IEEE Press), 83-97.
- Oja, E., Karhunen, J., Wang, L. and Vigario, R. (1995)** Principal and independent components in neural networks - recent developments. *Proceedings of the VII Italian Workshop on Neural Networks*, Italy, 16-35.
- Openshaw, S. (1994)** Neuroclassification of spatial data. In *Neural Nets: Applications in Geography*, edited by B. C. Hewitson and R. G. Crane. (London: Kluwer Academic Publishers), 53-70.
- Opitz, D. W. (1997)** The effective size of a neural network: a principal component approach. *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, USA, 263-271.
- Overall, J. E. and Klett, C. J. (1972)** *Applied Multivariate Analysis*. (London: McGraw-Hill).
- Pankhurst, R. J. (1991)** *Practical Taxonomic Computing*. (Cambridge: Cambridge University Press).
- Pao, Y. (1989)** *Adaptive Pattern Recognition and Neural Networks*. (New York: Addison-Wesley).
- Paola, J. D. (1994)** Neural network classification of multispectral imagery. MSc Thesis, The University of Arizona, USA.
- Paola, J. D. and Schowengerdt, R. A. (1994)** Comparisons of neural networks to standard techniques for image classification and correlation. *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'94)*, Pasadena, USA, 1404-1406.

- Paola, J. D. and Schowengerdt, R. A. (1995a)** A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, **33**, 981-996.
- Paola, J. D. and Schowengerdt, R. A. (1995b)** A review and analysis of backpropagation neural networks for classification of remotely-sensed multi-spectral imagery. *International Journal of Remote Sensing*, **16**, 3033-3058.
- Paola, J. D. and Schowengerdt, R. A. (1997)** The effect of neural-network structure on a multispectral land-use/land-cover classification. *Photogrammetric Engineering and Remote Sensing*, **63**, 535-544.
- Partridge, D. and Yates, W. B. (1996)** Replicability of neural computing experiments. *Complex Systems*, **10**, 257-281.
- Pedersen, M. W., Hansen, L. K. and Larsen, J. (1995)** Pruning with generalization based weight saliencies:  $\gamma$ OBD,  $\gamma$ OBS. *Proceedings of Neural Information Processing Systems 1995*, Denver, CO, USA, 521-528.
- Pekalska, E.** The multidimensional scaling problem, (1998), Delft University of Technology, Netherlands, available at <http://www.ph.tn.tudelft.nl/~ela/mds/node1.html> (10 Mar. 1999).
- Picchiotti, A., Casacchia, R. and Salvatori, R. (1997)** Multitemporal principal component analysis of spectral and spatial features of the Venice Lagoon. *International Journal of Remote Sensing*, **18**, 183-196.
- Pierce, L. E., Sarabandi, K. and Ulaby, F. T. (1994)** Application of an artificial neural network in canopy scattering inversion. *International Journal of Remote Sensing*, **15**, 3263-3270.
- Plumbley, M. D. (1991)** On information theory and unsupervised neural networks. Technical Report: *CUED/F-INFENG/TR.78*, Engineering Department of Cambridge University, UK.
- Pollack, J. B. (1989)** Connectionism: past, present and future. *Artificial Intelligence Review*, **3**, 3-20.
- Pudil, P., Novovicovo, J. and Kittler, J. (1994)** Floating search methods in feature selection. *Pattern Recognition Letters*, **15**, 1119-1125.
- Pykett, C. E. (1978)** Improving the efficiency of Sammon's Nonlinear Mapping by using clustering archetypes. *Electronics Letters*, **14**, 799-800.
- Reed, R. (1993)** Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, **4**, 740-747.

- Riedmiller, M. and Braun, H. (1993)** A direct adaptive method for faster backpropagation learning: the RPROP algorithm. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, USA, 586-591.
- Ripley, B. D. (1996)** *Pattern Recognition and Neural Networks*. (Cambridge: Cambridge University Press).
- Ripley, B. D. (1993)** Statistical aspects of neural networks. In *Networks and Chaos - Statistical and Probabilistic Aspects*, edited by O. E. Barndorff-Nielsen, J. L. Jensen and W. S. Kendall. (London: Chapman & Hall), 40-123.
- Rumelhart, D. E., McClelland, J. L. and PDP Group (1986)** *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vol.1 Foundations*. (London: The MIT Press).
- Sammon, J. W. (1969)** A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, **18**, 401-409.
- Sarle, W. S. (1995)** Stopped training and other remedies for overfitting. *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, Cary, NC, USA, 352-360.
- Sarle, W. S. Neural network FAQ, (2000)**, (<ftp://ftp.sas.com/pub/neural/FAQ.html>).
- Scott, M. J. J., Niranjani, M. and Prager, R. W. (1998)** Parcel: feature subset selection in variable cost domains. Technical Report: *CUED/FINFENG/TR.323*, Engineering Department of Cambridge University, UK, ([ftp://svr-ftp.eng.cam.ac.uk/pub/reports/Scott\\_tr323.ps.gz](ftp://svr-ftp.eng.cam.ac.uk/pub/reports/Scott_tr323.ps.gz)).
- Sharma, K. M. S. and Sarkar, A. (1998)** A modified contextual classification technique for remote sensing data. *Photogrammetric Engineering and Remote Sensing*, **64**, 273-280.
- Siedlecki, W. and Sklansky, J. (1988)** On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, **2**, 197-220.
- Sietsma, J. and Dow, R. J. F. (1988)** Neural net pruning - why and how. *Proceedings of IEEE International Conference on Neural Networks*, San Diego, USA, 325-333.
- Sietsma, J. and Dow, R. J. F. (1991)** Creating artificial neural networks that generalize. *Neural Networks*, **4**, 67-79.
- Skidmore, A. K., Turner, B. J., Brinkhof, W. and Knowles, E. (1997)** Performance of a neural network: mapping forests using GIS and remotely sensed data. *Photogrammetric Engineering and Remote Sensing*, **63**, 501-514.

- Smieja, F. J. (1991)** Hyperplane "spin" dynamics, network plasticity and back-propagation learning. Technical Report: 634, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Germany.
- Sneath, P. H. A. and Sokal, R. R. (1973)** *Numerical Taxonomy*. (San Francisco: W.H. Freeman).
- Sontag, E. D. (1989)** Sigmoids distinguish better than Heavisides. *Neural Computation*, 1, 470-472.
- Sontag, E. D. (1998)** VC dimension of neural networks. In *Neural Networks and Machine Learning*, edited by C. M. Bishop. (Berlin: Springer-Verlag), 69-95.
- Stauffer, P. and Fischer, M. M. (1997)** Spectral pattern recognition by a two-layer perceptron: effects of training set size. In *Neurocomputation in Remote Sensing Data Analysis*, edited by I. Kanellououlos, G. G. Wilkinson, F. Roli and J. Austin. (London: Springer), 105-116.
- Story, M. and Congalton, R. G. (1986)** Accuracy assessment: a user's perspective. *Photogrammetric Engineering and Remote Sensing*, 52, 397-399.
- Sutton, J. C. (1992)** Manufacturing applications of neural networks for the 90s. *Proceedings of the ASME Manufacturing International*, New York, USA, 177-189.
- Swain, P. H. and Davis, S. M. (1978)** *Remote Sensing: The Quantitative Approach*. (New York: McGraw-Hill).
- Swain, P. H. and King, R. C. (1973)** Two effective feature selection criteria for multispectral remote sensing. *Proceedings of the 1st International Joint Conference on Pattern Recognition*, Washington, USA, IEEE 73 CHO821-9 C, 536-540.
- Swingler, K. (1996a)** *Applying Neural Networks - A Practical Guide*. (London: Academic Press).
- Swingler, K. (1996b)** Financial prediction: some pointers, pitfalls and common errors. *Neural Computing and Applications*, 4, 192-197.
- Thackrah, G., Barnsley, M. and Pan, P. (1999)** Merging land cover classifications from artificial neural networks using the Sugeno fuzzy integral. *Proceedings of the 25th Annual Conference and Exhibition of the Remote Sensing Society*, Cardiff, UK, 71-78.
- Thimm, G. and Fiesler, E. (1997a)** High-order and multilayer perceptron initialization. *IEEE Transactions on Neural Networks*, 8, 349-359.
- Thimm, G. and Fiesler, E. (1997b)** Optimal setting of weights, learning rate and gain. IDIAP Technical Report: 97-04, (<ftp://ftp.idiap.ch/pub/reports/1997/tr97-04.ps.gz>).

- Thomas, I. L., Benning, V. M. and Ching, N. P. (1987a)** *Classification of Remotely Sensed Images*. (London: Adam Hilger).
- Thomas, I. L., Ching, N. P., Benning, V. M. and D'Aguanno, J. A. (1987b)** A review of multi-channel indices of class separability. *International Journal of Remote Sensing*, **8**, 331-350.
- Tso, B. (1997)** An investigation of alternative strategies for incorporating spectral, textural and contextual information in remote sensing image classification. PhD Thesis, The University of Nottingham, Nottingham, UK.
- Tso, B. and Mather, P. M. (1999)** Crop discrimination using multi-temporal SAR imagery. *International Journal of Remote Sensing*, **20**, 2443-2460.
- Tucker, C. J. (1979)** Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, **10**, 127-150.
- Vieira, C. A. O. and Mather, P. M. (1999)** Visualising the spatial accuracy of classified images. *Proceedings of the 25th Annual Conference and Exhibition of the Remote Sensing Society*, Cardiff, UK, 897-903.
- Vuurpijl, L. (1998)** Platforms for artificial neural networks. PhD Thesis, The University of Nijmegen, The Netherlands.
- Wang, C. (1994a)** A theory of generalisation in learning machines with neural network application. PhD Thesis, The University of Pennsylvania, USA.
- Wang, F. (1990a)** Fuzzy supervised classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, **28**, 194-201.
- Wang, F. (1990b)** Improving remote sensing image analysis through fuzzy information representation. *Photogrammetric Engineering and Remote Sensing*, **56**, 1163-1169.
- Wang, F. (1994b)** The use of artificial neural networks in a geographical information system for agricultural land-suitability assessment. *Environment and Planning A*, **26**, 265-284.
- Wang, Z., Di Massimo, C., Tham, M. T. and Morris, A. J. (1994)** A procedure for determining the topology of multilayer feedforward neural networks. *Neural Networks*, **7**, 291-300.
- Wegman, E. J. (1990)** Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, **85**, 664-675.
- Wegman, E. J. and Luo, Q. (1997)** High dimensional clustering using Parallel Coordinates and the Grand Tour. *Computing Science and Statistics*, **28**, 352-360.

- Werbos, P. J. (1995)** Backpropagation: basics and new developments. In *The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib. (London: The MIT Press), 134-139.
- Wessels, L. F. A. and Barnard, E. (1992)** Avoiding false local minima by proper initialization of connections. *IEEE Transactions on Neural Networks*, **3**, 899-905.
- Weszka, J. S., Dyer, C. R. and Rosenfeld, A. (1976)** A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man and Cybernetics*, **6**, 269-285.
- Whitley, D. (1995)** Genetic algorithms and neural networks. In *Genetic Algorithms in Engineering and Computer Science*, edited by G. Winter, J. Periaux, M. Galan and P. Cuesta. (New York: John Wiley & Sons), 203-216.
- Wilkinsn, G. G. (1997)** Open questions in neurocomputing for Earth observation. In *Neurocomputation in Remote Sensing Data Analysis*, edited by I. Kanellopoulos, G. G. Wilkinson, F. Roli and J. Austin. (London: Springer), 3-13.
- Wilson, J. D. (1992)** A comparison of procedures for classifying remotely-sensed data using simulated data sets. *International Journal of Remote Sensing*, **13**, 365-386.
- Woodcock, C. E. and Strahler, A. H. (1987)** The factor of scale in remote sensing. *Remote Sensing of Environment*, **21**, 311-332.
- Yang, J. and Honavar, V. (1998)** Feature subset selection using a genetic algorithm. In *Feature Extraction, Construction and Subset Selection: A Data Mining Perspective*, edited by H. Motoda and H. Liu. (New York: Kluwer), 117-136.
- Zell, A., Mamier, G., Vogt, M., Mache, N., Hübner, R., Döring, S., Herrmann, K., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeorgiou, A., Posselt, D., Schreiner, T., Kett, B., Clemente, G. and Wieland, J.** Stuttgart neural network simulator (SNNS): User Manual (version 4.1), (1999), University of Stuttgart (<http://www-ra.informatik.uni-tuebingen.de/SNNS/UserManual/UserManual.html>) (21 Nov. 2000).
- Zhou, J. and Civco, D. L. (1996)** Using genetic learning neural networks for spatial decision making in GIS. *Photogrammetric Engineering and Remote Sensing*, **62**, 1287-1295.
- Zhuang, X., Engel, B. A., Lozano-Garcia, D. F., Fernandez, R. N. and Johannsen, C. J. (1994)** Optimization of training data required for neuro-classification. *International Journal of Remote Sensing*, **15**, 3271-3277.
- Zongker, D. and Jain, A. (1996)** Algorithms for feature selection: an evaluation. *Proceedings of the 13th IEEE International Conference on Pattern Recognition*, Vienna, Austria, 18-22.

# VISUALISATION TOOLKIT FOR ANALYSING ARTIFICIAL NEURAL NETWORKS

## A.1 Introduction

This appendix is a guide to the visualisation toolkit, which is contained on the CD-ROM accompanying this thesis. The toolkit and its application to some specific problems are mentioned in previous chapters, where relevant. It is recommended that before using the toolkit this appendix should be read thoroughly to understand the use of the techniques available in the toolkit. Several sample datasets are provided with the toolkit to help new users to practice operations that will be fully discussed in following sections.

The toolkit basically provides the following facilities: sampling images to create datasets (pattern files) for training, validation and testing; graphical analysis of the datasets through Parallel Coordinate Display and Andrews' Plots; reducing the dimensionality of datasets using Sammon's Nonlinear Mapping algorithm, and feature selection methods including separability indices (the Divergence, the Transformed Divergence measures, the Bhattacharyya distance, and the Jeffries-Matusita distance), statistical tests (Wilks'  $\Lambda$  and Hotelling's  $T^2$ ) and Mahalanobis Distance classifier; batch (configuration) file creation for Stuttgart Neural Network Simulator (SNNS) batch mode processing; running SNNS in normal and batch modes; testing the trained neural networks; visualising data in two or three dimensions; creating new pattern files by eliminating some pixels using the interactive tools available; preparing GIF animations to display the whole training

process; analysing network weights using a line graph; assessing individual class accuracies together with overall accuracy using a histogram; converting ANN results to IDRISI image files; accuracy assessment of results files using contingency matrices; visualising the result of the classification of a test image in terms of the degree of output activation levels, and image classification using the Mahalanobis distance (MDC) or the maximum likelihood (ML) classifiers.

The toolkit is written for PCs running the Microsoft Windows 95 operating system. It is developed using MATLAB (version 5.3), which is a powerful, comprehensive, and easy-to-use environment for performing technical computations. MATLAB integrates computation, data analysis, visualisation, and programming in a flexible, open, environment. Other versions of MATLAB may demonstrate unexpected behaviour, particularly while running some of the visualisation procedures. Therefore, no guarantee is given that the software will run on other versions of MATLAB. Most of the facilities involving data analysis and visualisation tasks and the structure of the toolkit, including menus and buttons, are provided by the routines written in MATLAB, which have file extensions of 'm'. However, some of the menu items performing essential calculations and analyses are accomplished through C++ programs, which are compiled in Turbo C (version 3.0).

The primary aim of developing this toolkit is to analyse the characteristics of the data and the neural networks using scientific visualisation techniques. Unfortunately, there is currently no available comprehensive software or toolkit to perform all the tasks considered in this thesis. The toolkit has been developed specifically to meet the objectives drawn up for this study.

The general structure of the toolkit is described in subsequent sections, with an emphasis on menu items performing various data analyses, assessments, and visualisation tasks. In addition to the default menu headings in MATLAB, there are five menu headings covering all the facilities in the toolkit: *Data Analysis* includes the analyses that are performed on datasets prior to neural network

classification; *Classification* contains menu items related to network training and testing, and displaying the results of ANN classifications; *Evaluate* mainly consists of utilities to analyse the result of network training in various ways; the *Utilities* section includes several routines to help the user to add information into figures; the *Save* menu is to save new pattern files and database files, which are used to store and retrieve all information related to the pattern datasets. It should be noted that special attention was paid to make the menu items and program dialogue simple and clear to users, bearing in mind the difficulty of understanding ANN terminology and wide variety of techniques employed in this study. All menu items available in the toolkit are shown in Figure A.1.

## **A.2 Installing and Running the Toolkit**

All the programs and sample files are provided with the CD-ROM attached to this thesis. In order to install the toolkit to a desktop, all the files with exact directory structure must be copied to the hard disc. Note that programs needed for genetic algorithm applications are copied to a separate directory called *gademo*. It is recommended that all sample files provided and new data files to be created should be saved into separate directories to avoid the possibility of deleting files accidentally.

Directories containing the files for the toolkit and Animagic GIF Animator must be registered in MATLAB using the command 'addpath'. The use of the command can be displayed by the MATLAB command 'help addpath'. As a result, specified directories are added to the search path, and any file inside one of these directories is automatically retrieved. The directories added to the list can be displayed using the command 'path'. Once these steps are completed, the toolkit is ready to run from any directory, in which all new files will be saved, using the command 'phd'. Note that some help can be obtained from the text files (Readme.txt) available in the CD-ROM.

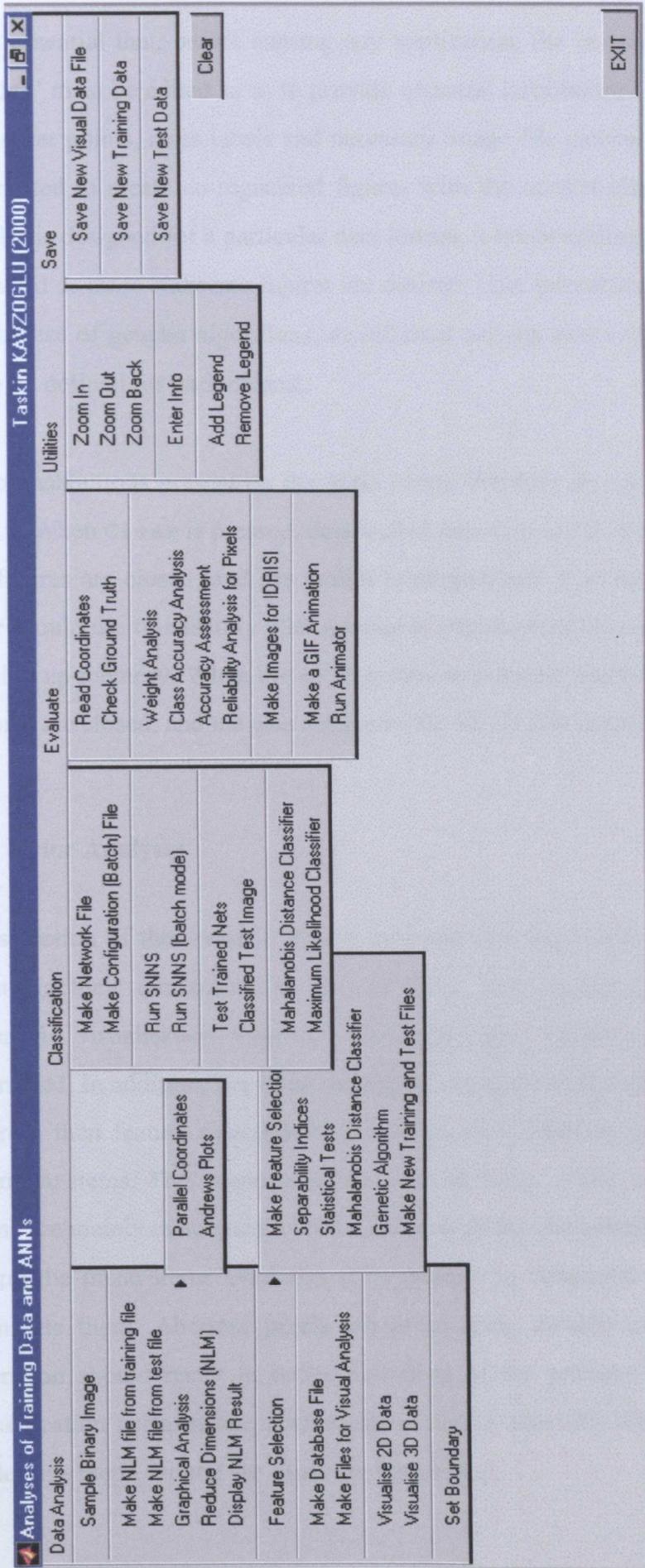


Figure A.1 Menus of the toolkit covering all the analyses.

It is essential that, before starting any application, the header section of the file 'phd.m' must be edited so as to provide essential information, such as coordinates of corner points, class labels and necessary image file names. The information is later used to create co-registered figures with the correct class labels. Since the toolkit is designed for a particular data format, it needs editing if different datasets are used or more elaborate figures are desired. User interaction is mostly required in the use of genetic algorithms, as different penalty terms and parameter values may be defined for each dataset.

Two pushbuttons present on the Main Menu Window are labelled as `Clear` and `EXIT`. When `Clear` is pressed, data loaded into memory in MATLAB are deleted, all figures are closed, and the toolkit is reinitialised. It is recommended that the user should use this facility after a series of experiments to speed up MATLAB by freeing up memory. When the `EXIT` button is selected, memory is cleared and all figures are closed, and the user returns to the MATLAB command line.

### **A.3 Prior Analyses**

This section of the menu is mainly for analysing the characteristics of datasets, reducing their dimensions to two or three, and visualising them. Using the scientific visualisation tools provided, individual pixels and clusters can be examined. In addition, if a large number of spectral bands available from different sources then feature selection methods can be utilised by selecting appropriate submenu items. This menu section is named `Data Analysis` since the menu items are mainly concerned with the analysis of the characteristics of the datasets. Using the menu items available, it is possible to determine aberrant pixels and eliminate them. Aberrant pixels are those lying outside of the clusters. This operation should result in better definition of the problem at hand and better classification performance from artificial neural networks. Operations performed under this menu section are shown in Figure A.2.

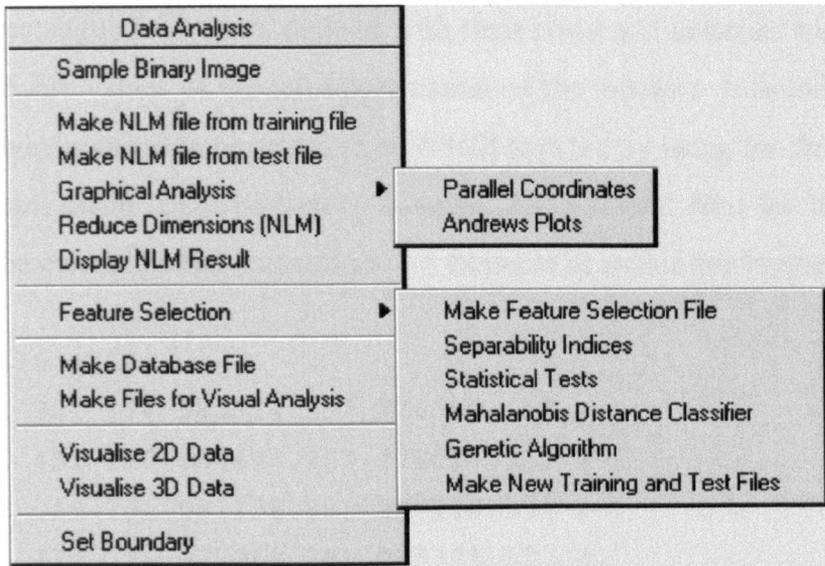
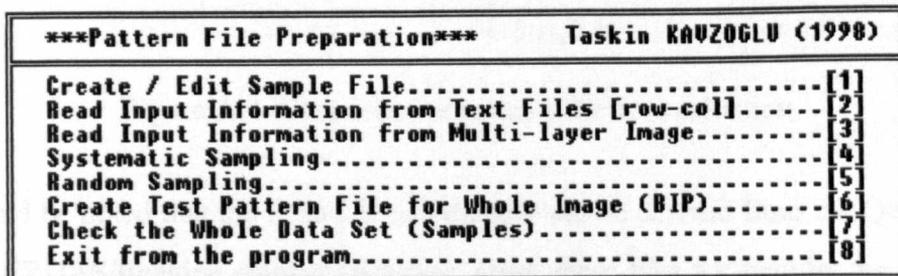


Figure A.2 Items listed under Data Analysis menu heading.

### A.3.1 Preparing Pattern Files

Preparation of pattern files for ANN, MDC and ML classifications is the first step that should be carried out with care, as the accuracy and the quality of all subsequent analyses are highly dependent on the characteristics of the data contained in these files. This procedure is selected by choosing Data Analysis from the Main Window and clicking on Sample Binary Image. Once this operation is selected, a new menu opens in a MS-DOS window with eight options (Figure A.3).



Please ENTER your choice...:

Figure A.3 Menu for the program used for training and test pattern file creation.

Using the MS-DOS menu window, two types of sampling strategies can be performed on images. The first, and simpler one is to select the pixels from user-

defined rectangular windows defined with their rows and columns together with the coordinate values of the left-upper corner of the window. Information for all selected windows has to be stored in an ASCII text file by using the first option of the program menu (Create/Edit Sample File). Text files for this purpose can also be create in other text editors. An example of such a file is given below:

```
5 280 475 6 24
275 152 301 174 580514.007 276552.410 4
441 91 451 100 584497.257 278017.160 7
301 122 316 136 581136.882 277273.535 3
169 205 175 219 577968.882 275281.535 6
327 121 339 136 581760.882 277297.535 1
```

The first line, 5 280 475 6 24, shows the number of samples, number of rows and columns in the image, number of bands and resolution in metres, respectively. The second line, 275 152 301 174 580514.007 276552.410 4, represents the number of rows and columns in a rectangular area, and class number respectively. The idea of definition of rectangular areas is shown in Figure A.4, which illustrates the second line of the text file given above.

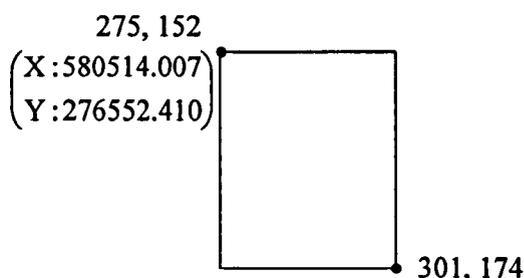


Figure A.4 Illustration of rectangular area selection.

It should be noted that the X and Y coordinates can be derived from the Query Box in the ERDAS/Imagine software package. Once these files are read by the program by selecting the second option from the menu, all the required information from the image is read using the specified row-and-column of rectangles. Coordinate values of each pixel are also estimated, and all the information is stored in an array that is later used in the sampling process. Note that the images used in this operation are required to be in BSQ (Band Sequential) image format.

The second strategy to produce samples (Read Information from Multi-Layer Image) is to use an image having ground truth information as the first layer. For this option, the image selected must be in BIP (Band Interlaced by Pixel) image format, to allow the retrieval of all band values for each pixel in a sequence without reading all the layers (or bands). In this procedure, there is no need to describe the areas for the land cover classes. Some inputs, specifically coordinates of the image corners, resolution of the image, and the number of bands that the image contains, are requested by the program to be used in computing the coordinates of each pixel selected. While reading the image, the program randomly searches for pixels having class labels, reads pixel values for each image band, and writes them into an array. In order to avoid the selection of an excessive number of patterns for a particular class, the program calculates the number of patterns for each class and tries to select approximately equal number of patterns for each class. Lastly, the coordinates of each pixel are estimated and added into the array.

The program offers two options for sampling the data loaded into memory: systematic sampling and random sampling. In systematic sampling, which is a simple procedure, every  $n^{\text{th}}$  data item in the array is chosen to form a training data file. Alternatively, random sampling, which is a more sophisticated procedure, can be used. Two options are made available: The total number of patterns to be selected from the image and the number of patterns that will be selected for each class. The total number of patterns selected, resulting from the initial random pixel selection process, is displayed and the number of patterns required for the pattern dataset is requested. Then, the user-defined number of pixels is randomly selected from the pixels placed in the array. The second option lists the number of patterns for each class, and asks for the number of patterns that the user wishes to sample. Note that this number must be equal or less than the smallest number of patterns selected for any individual class. Next, the program randomly selects patterns in equal numbers for each class. Once the sampling is completed, it is possible to save the training and testing pattern files in SNNS format. Information about the selected pixels in terms of their locations and DN (Digital Number) values for each band can be displayed by choosing menu option 7.

### A.3.2 Graphical Analysis

This process can be initiated by selecting Data Analysis|Graphical Analysis from the Main Menu. In this section, two popular graphical analysis techniques, parallel coordinates and Andrews' plots, are available to visualise the multi-dimensional data in their original dimensions. Whilst parallel coordinate plots represent each data point with a broken line providing that each dimension is characterised by an axis and all axes are parallel to each other, Andrews' plots convert each multi-dimensional sample data point into an orthogonal sinusoidal function. The underlying theory behind these techniques can be found in sections 4.2.2 and 4.2.3 in Chapter IV.

### A.3.3 Reducing the Dimensions of the Dataset

In order to visualise multi-dimensional datasets on a computer screen, it is essential to reduce the number of dimensions to two or three. For this purpose Sammon's Nonlinear Mapping algorithm (NLM), discussed in section 4.3.2.2 in Chapter IV, is selected and utilised in this study. The fundamental idea behind the method is to iteratively search for new dimensions by preserving the distance between the points as much as possible. Before running the process, pattern files must be converted to a specific format that the NLM program requires. Therefore, for training data files Data Analysis|Make NLM from training file and for test data files Data Analysis|Make NLM from test file submenu routines are executed from the Main Menu. Thus, an ASCII text file, which is in the format shown below, is created.

```
7 6
0.435294 0.282353 0.333333 0.556863 0.854902 0.858824
0.282353 0.222353 0.333333 0.576471 0.870588 0.874510
0.317647 0.215686 0.341176 0.529412 0.870588 0.878431
0.337255 0.200000 0.341176 0.529412 0.862745 0.866667
0.384314 0.192157 0.360784 0.549020 0.870588 0.874510
0.368627 0.227451 0.376471 0.549020 0.862745 0.874510
0.407843 0.247059 0.509804 0.539804 0.870588 0.882353
```

The C++ program to perform Sammon's Nonlinear Mapping algorithm is converted from a Fortran program written by Prof. Paul Mather. The inputs required by the program are the number of dimensions onto which the data are to be projected, and the 'magic factor', which is the step length used in the minimisation program. An error to stop criterion has to be also set to end the iterative process. The program is limited to use 5,000 sample patterns with 25 dimensions. Although the original dimensions of the data can be reduced to any dimensions lower than the original, two or three dimensions must be chosen to enable the toolkit to visualise and analyse the data. The output of this operation is a list of new coordinates in new dimensions. An example output file is:

0.316	-1.154	0.699
0.174	-1.183	0.784
0.214	-1.144	0.802
0.227	-1.136	0.784
0.268	-1.131	0.779
0.269	-1.132	0.762
0.430	-1.062	0.757

where the first column corresponds to X, the second column to Y and the third column to Z coordinates.

It is also possible to reduce the dimensions of the datasets by using feature selection techniques before processing with neural networks. By using this module, the user can determine the least effective image bands in distinguishing between classes and eliminate them so as to produce smaller neural networks. All the techniques that are discussed in Chapter V, devoted to feature selection, can be applied to datasets using the toolkit menus. After selecting `Data Analysis | Feature Selection` from the Main Window, further selections can be made on the type of separability measure and the search algorithm to be used. The first step in the process is to create files in the suitable format. This can be done by selecting `Make New Training and Test Files` from feature selection menu. The methods based on Sequential Forward Selection (SFS) search algorithm are shown in the submenu as `Separability Indices`, `Statistical Tests`, and `Mahalanobis Distance Classifier`, while all separability measures for Genetic Algorithm search method are initiated from

Genetic Algorithm menu item. For all operations, the first question is whether the data are already loaded into memory, or need to be loaded from a file. The second question is how many bands for best band combination are required. Once the program has run, the best subset band combination and detailed information about that solution in terms of separability measures is displayed. The solutions attained can then be used to form new training and test files for neural network processing. This operation is optional at the end of all processes except for the GA application. These files can be also generated at later stages by calling Make New Training and Test Files item from the menu.

The Genetic Algorithm menu item runs a MATLAB program called gademo, originally developed by Ron Shaffer from the Naval Research Laboratory, Washington, USA, and available as a free demo program on <http://chemdiv-www.nrl.navy.mil/6110/sensors/chemometrics/gademo.html>. Version 1.2 of the demonstration program has been adapted, and many changes and additions have been made in order to apply feature selection techniques to satellite image data. Once the program is called, the following menu appears:

```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  Binary-coded Genetic Algorithm Demo
  Ron Shaffer --> adapted by Taskin Kavzoglu
  School of Geography, The University of Nottingham
  Version 1.2
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  Current GA Configuration
1  Chromosome Length           24
2  Population Size             20
3  Number of Generations       10
4  Mutation Rate               0.001
5  Crossover Rate              0.600
6  Crossover Type:             Single
7  Elitist Operator:           On
8  Evaluation Function:         Divergence
9  Gray Coding:                 Off
10 Number of Bands in Subset    8
11 Start GA Optimization (Full Printout)
12 Start GA Optimization (Minimal Printout)
13 Quit GA Demo
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Enter Option (1-13) >

```

Chromosome Length shows the number of genes present on the chromosome. The default set for this parameter is 24, corresponding to the number of bands available for the first test site. Population Size is the number of chromosomes in the population. Although larger populations increase the amount of variation existing in the population, they require more fitness function evaluations and this leads to a considerable increase in computer time. Number of Generations shows the maximum number of generations to be produced. Mutation Rate, Crossover Rate and Crossover Type are the basic parameters of genetic algorithms, which are discussed in detail in section 5.7.3 of Chapter V. The Elitist Operator determines whether the best chromosome for each population is moved to the next generation unchanged. If the elitist operator is turned on, the best fitness score from one population to the next will never decrease. The default setting of the parameter is On.

The Evaluation Function shows the fitness function to be used for assessing the performances of chromosomes. Once item 8 is selected from the menu, a list of evaluation functions is displayed. These are the same separability measures that are used by SFS algorithm. Gray Coding is used to convert binary chromosomes to real valued variables. The Number of Bands in Subset shows the number of bands that will be selected for the best subset solution. In the performance evaluation stage, this value is utilised to penalise the chromosome solutions that include more bands than required. The default is set to 8, which corresponds to best eight-band subset solutions. Optimization with Full Printout allows the program to display each chromosome with its performance and some basic statistics about the generations produced. Optimization with Minimal Printout, on the other hand, displays only the statistics for each generation.

Finally, Quit is used to end the process. After the genetic algorithm process is completed, two figures are displayed. The first figure displays the fitness scores of the best chromosomes and the mean for each generation, while the second figure plots the number of times each gene selected (a value of 1) in a histogram.

### A.3.4 Visualising and Cleaning the Data

Prior to visualising datasets, a database file that includes all dataset-related information must be created. Database files are loaded into memory at the beginning of visual and quantitative analyses before being used in further analyses. A database file basically contains DN values of pixels for image bands, X and Y coordinates of the pixels, two and three-dimensional coordinates produced using the NLM algorithm, and class labels of pixels. Creating a database file is an easy and automated procedure, which is run by the `Make Database File` menu item.

Visualising and cleaning datasets are essential tools proposed in this study. After reducing the dimensions of the data to two or three, using Sammon's Nonlinear Mapping algorithm (NLM), it is possible to visualise the result of the process by selecting `Data Analysis|Display NLM Result` from the Main Window. Depending on the dimensions of the data, the display uses either a two or three-dimensional coordinate system. However, the main visualisation operation is carried out after an initial ANN classification is performed and class labels of pixels are assigned. Thus, clusters for existing classes can be easily observed. The `Make Files for Visual Analysis` menu item starts the process of combining the results of NLM and ANN classification. Once a file is created for visual analysis, it can be displayed using either `Visualise 2D` or `Visualise 3D`, depending on the data format. When the data are displayed in a figure, several facilities, including zooming and adding legends, can be initiated using the submenu items under the `Utilities` menu heading. Exclusive to 3-D visualisations several additional facilities are available. These include rotating all the axes using the mouse, rotating X and Z axes by specific amount using azimuth and elevation slide bars, and having a tour around the data by pressing `Data Tour` button. If the `Data Tour` button is clicked, an information window appears with the message of `To stop the process Click on figure`. Once you click on `OK`, this information window will disappear, and the Z-axis is rotated between  $-90$  and  $90$  degrees with 5 degree increments at constant elevation of 30 degrees. If at any point of the process the mouse is clicked on the figure, the process will stop at that azimuth and elevation

configuration. It is also possible to switch to two-dimensional views of XY, YZ and XZ by clicking the appropriate button available on the right side of the figure. An example view of a 3-D view is given in Figure A.5.

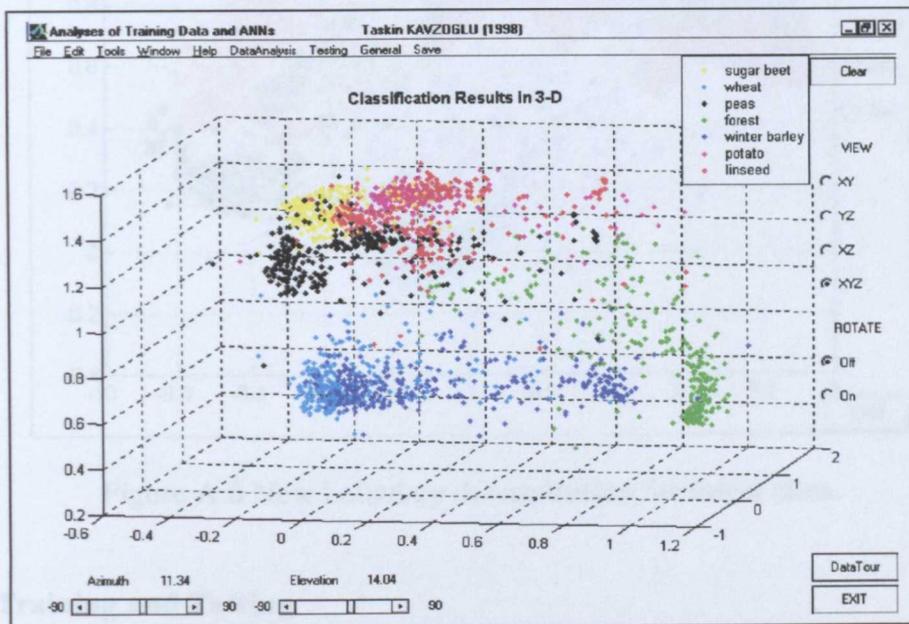


Figure A.5 Visualising classification results in three-dimensional form.

Another facility that is available in the toolkit is to define new boundaries for the classes, and thus create new classes. This operation is invaluable for studies where a general class, such as forest, includes several subclasses such as coniferous and deciduous forests, and the analyst wants to separate the subclasses from each other. The Data Analysis|Set Boundary menu item is used for this operation. Once a boundary is drawn around a group of pixels, a label must be assigned to the new class. Note that new boundary settings can only be performed on two-dimensional views. If a 3-D view is on, the user is asked to switch to one of the 2-D views. A view of the procedure of new boundary definition is shown in Figure A.6. When all the new classes are defined by drawing boundaries around pixels, new pattern and visualisation files can be created from the Save menu items. It should be noted that only the pixels located inside the boundaries are assigned to new classes. If a pixel lies outside all boundaries, that pixel will not be placed in any of the new files created.

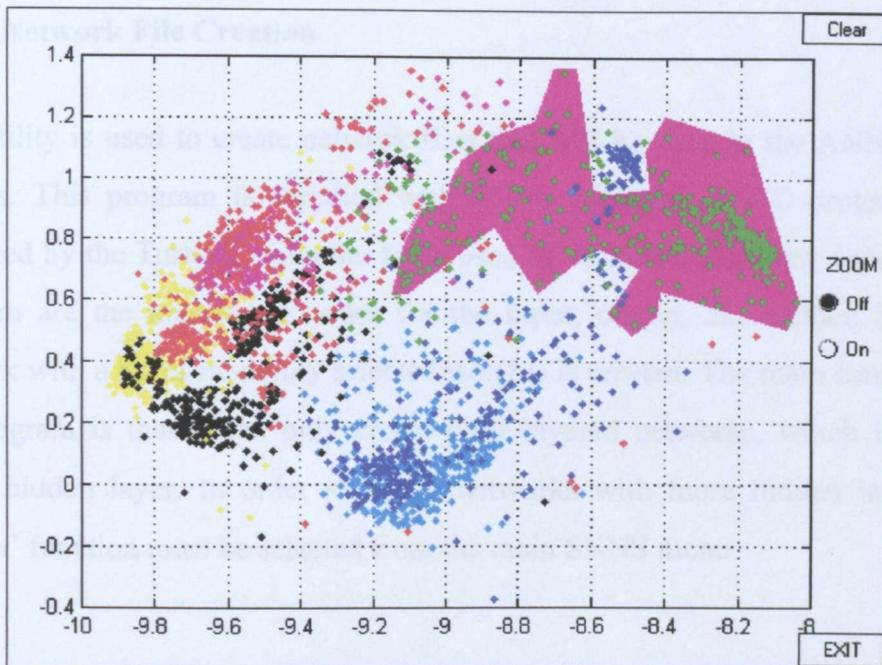


Figure A.6 New boundary determination for forest class.

#### A.4 Training and Testing

In this section of the menu, creation of files necessary for neural network training using SNNS software, producing results from trained networks, and image classifications using the Mahalanobis distance and Maximum likelihood classifiers are carried out. Results of the operations, namely Classified Test Image, Mahalanobis Distance Classifier and Maximum Likelihood Classifier are classified images of a test site. The submenu items of this menu are shown in Figure A.7.

Classification
Make Network File
Make Configuration (Batch) File
Run SNNS
Run SNNS (batch mode)
Test Trained Nets
Classified Test Image
Mahalanobis Distance Classifier
Maximum Likelihood Classifier

Figure A.7 Items listed under the Classification menu heading.

#### **A.4.1 Network File Creation**

This utility is used to create network files that will be used in the ANN training process. This program is supplied with SNNS software as a C program. It is compiled by the Turbo C compiler to be used in desktop computers. Inputs to the program are the number of nodes for the input, output, and hidden layers. A network with a set of randomly selected weights is created. The main limitation of the program is that it can only create three-layered networks, which include a single hidden layer. In order to create networks with more hidden layers, the 'Bignet' function must be selected from the main SNNS menu.

#### **A.4.2 Preparation and Running Batch Files**

Instead of running the SNNS software directly and setting up all the parameters in the program's control panel, it is possible to create a configuration (or batch) file including all settings by choosing the submenu item `Make Configuration (Batch) File` from the `Classification` menu heading. Once this item is chosen, filenames and parameters for network training are requested via command line input, and the configuration file is automatically created in the format that is required by SNNS. Specifically, configuration files consist of learning parameters, input files (network and training pattern files), and some options such as shuffling the patterns. It is also possible to include a validation dataset in the training process. In addition, the learning rate can be reduced after a certain number of iteration determined by the analyst.

Configuration files are run from the `Run SNNS (batch mode)` menu item. Information about the training error and the number of cycles is displayed in an MS-DOS shell. Running SNNS in batch mode gives the advantage of fast processing, easy and guided selection of options and parameters. The SNNS batch processing language is called 'Batchman', which is described in Chapter 13 of the SNNS Manual (Zell *et al.*, 1999). The format of a sample configuration file is as follows:

```

loadNet("net1.net")
loadPattern("train.pat")
setInitFunc("Randomize_Weights",0.3,-0.3)
initNet()
setLearnFunc("Std_Backpropagation",0.2)
setUpdateFunc("Topological_Order")
setShuffle(TRUE)
while SSE > 0.1 and CYCLES < 15000 do
  for i:=1 to 150 do
    if CYCLES mod 150 == 0 then
      print("cycles = ",CYCLES, " SSE = ",SSE," MSE = ",MSE) endif
      trainNet()
    endif
  endfor
  saveNet("lnet" + CYCLES div 150 + ".net")
  saveResult("lres" + CYCLES div 150 + ".res",1,PAT,TRUE,FALSE,"create")
endwhile

print("Cycles trained: ", CYCLES)
print("Training stopped at error(SSE): ", SSE)

```

### A.4.3 Training and Testing Artificial Neural Networks

The toolkit provides facilities to train and test artificial neural networks. In addition to training the networks in batch mode using configuration files, described above, network training can be performed by running SNNS software from **Classification|Run SNNS**. This menu item calls the Windows version of SNNS, available from <http://www-ra.informatik.uni-tuebingen.de/SNNS/>. As it requires a local server to run the software, a server program, specifically eXcursion (version 2.1), is installed and run beforehand. Although the Windows version of SNNS provides the same features as the UNIX version, the UNIX version of the software is found to be faster and more convenient to use. For the use of SNNS, please refer to its on-line manual available.

In order to test the trained neural networks using test pattern files, the **Classification|Test Trained Nets** menu item is clicked from the Main Menu. One or more results files are produced by this option. In the case of a single network file, a short configuration file, which is saved as 'bman1.cfg', is written to generate a single results file without initialising the SNNS software. The names of network and results files, together with the test pattern file, are requested on the MATLAB command line. These filenames are used to create the configuration file, a sample of which is given below:

```
loadPattern("test_elv.pat")
loadNet("ewrl.net")
saveResult("ewrl.res", 1, PAT, TRUE, FALSE, "create")
```

If a series of networks is tested, then test pattern file name and name tags of network and results files (in this case 1ewrl1 for network and ewr for results files) are entered in command line. From the information provided, a configuration file similar to the one shown below is created. Note that the configuration is written into the default file name of 'bman2.cfg'.

```
loadPattern("test_elv.pat")
loadNet("1ewrl11.net")
saveResult("ewr1.res", 1, PAT, TRUE, FALSE, "create")

loadNet("1ewrl12.net")
saveResult("ewr2.res", 1, PAT, TRUE, FALSE, "create")

loadNet("1ewrl13.net")
saveResult("ewr3.res", 1, PAT, TRUE, FALSE, "create")
```

#### **A.4.4 Displaying Classification Results for a Test Image**

Another facility provided by the toolkit is the visualisation of the results of ANN classifications using MATLAB functions. It is important that at the end of a project to produce a classification map of whole area of concern. This facility is provided to the users by the toolkit. This module is launched by selecting the Classification|Classified Test Image from the Main Menu Window. Several questions, including the names of the results file and the log file to store the results of the process, and the threshold for class membership evaluation, are asked by the program. The next question is whether the results are to be displayed as a grey-scale image or a colour tone image. In a grey-scaled display, whilst the pixels having a highest activation level less than the threshold value are set to a grey level of 100, those pixels having a highest activation level greater than the threshold are stretched between 100 and 300 so that value of 300 is allocated the pixels having activation levels of 1.0. Thus, grey level of 100 indicate that the ANN output is below a user-defined threshold level, while values of 101-300 indicate ANN outputs that exceed the threshold level.

Colour tone images are created by both looking at the class memberships and the highest activation levels of the pixels. Class memberships are determined from the location (or position) of the output nodes having the highest activation levels. Once this stage is completed, the next step is to find out how effectively the pixel is classified by the ANN. Therefore, the range defined by the threshold and the highest possible activation value, which is 1.0, is divided into four equal segments. For each class, four tones of a specific colour are assigned to pixels so that the darkest tone shows the highest probability of membership. It should be noted that the pixels having a highest activation rate less than the threshold set by the user are left unclassified, and in both images they are shown in white. At this point, the number of columns and rows must be provided to allow the visualisation of the resulting image. Outputs of both operations are written to log files defined by the user. Since they are in ASCII text format, they can be viewed in any editor. Images created from this process are presented in Chapter V of the thesis.

#### **A.4.5 Classifying Images Using the Mahalanobis Distance and Maximum Likelihood Classifiers**

The Mahalanobis distance (MDC) and maximum likelihood (ML) classifiers are the two most popular conventional statistical classifiers used by researchers for classifying satellite images. The underlying theories of MDC and ML are given in section 5.6 of Chapter V and section 2.4.2.3 of Chapter II, respectively. These techniques are made available in the toolkit so as to compare the results of the techniques with those produced by ANNs. Whilst classification based on the Mahalanobis distance classifier can be performed using the `Classification|Mahalanobis Distance Classifier` function, classification using the maximum likelihood classifier can be carried out from the `Classification|Maximum Likelihood Classifier` menu item. After the selection of the classifier, following enquires made by the programs are the same. Two options are available at this stage of the programs, namely 'Calculate Classification Accuracy for Test Files' and 'Classify Test Images Using Training Data'. By selecting the

first option an accuracy assessment is performed using particular training and test pattern files. Classification of the test images including whole test site can be carried out by selecting the second option. Note that the test data must be created for the whole test image in SNNS pattern file format. The next operation is to load the training and test data from memory or from files, depending on the user's choice. According to the response received, the data are loaded and an appropriate program is run to classify pixels. Basically, the characteristics of the training data are derived and later applied to classify test pixels or the image. For the test image, the result of the programs is a classified image similar to those produced by the ANN classifier.

### A.5 Posterior Analyses

The Evaluate menu heading provides operations for analysing the result of ANN classification. Results are assessed by several different methods, such as accuracy assessment, class accuracy analysis, and reliability analysis. In addition, results are converted to a format suitable for input to IDRISI GIS and image processing software. It is also possible to analyse changes in the network weights and the class memberships during the training process. All functions in the menu are shown in Figure A.8.

Evaluate
Read Coordinates Check Ground Truth
Weight Analysis Class Accuracy Analysis Accuracy Assessment Reliability Analysis for Pixels
Make Images for IDRISI
Make a GIF Animation Run Animator

Figure A.8 Items listed under the Evaluate menu heading.

### A.5.1 Analysing Individual Pixels

The first menu item (Read Coordinates) is used to start the process of analysing individual pixels in terms of location on the image and DN values in each band compared to other members of the class. This utility is particularly useful in the search for and elimination of mixed (if required) or atypical pixels. When the user clicks on the image, the closest point is searched for and its coordinates are displayed on the screen when found (Figure A.9). After that, for each press of the enter key, the location of the pixel corresponding to the closest point is indicated on ground truth and satellite images, which must be in TIF image format and defined in the heading section of main program 'phd.m'. Figures A.10 and A.11 show the location of a pixel selected on a ground truth image and a satellite image, respectively. The characteristics of the pixel are also analysed by means of histograms that portray the pixel values for the class to which the pixel belongs, and the position of the selected pixel on the histogram (Figure A.12). This helps to identify the pixels that lie away from others in the same class. If the analyst finds that the pixel is atypical and should be removed, then he/she can click on the delete button on the figure. Pixels marked for deletion are later excluded from the new files being saved.

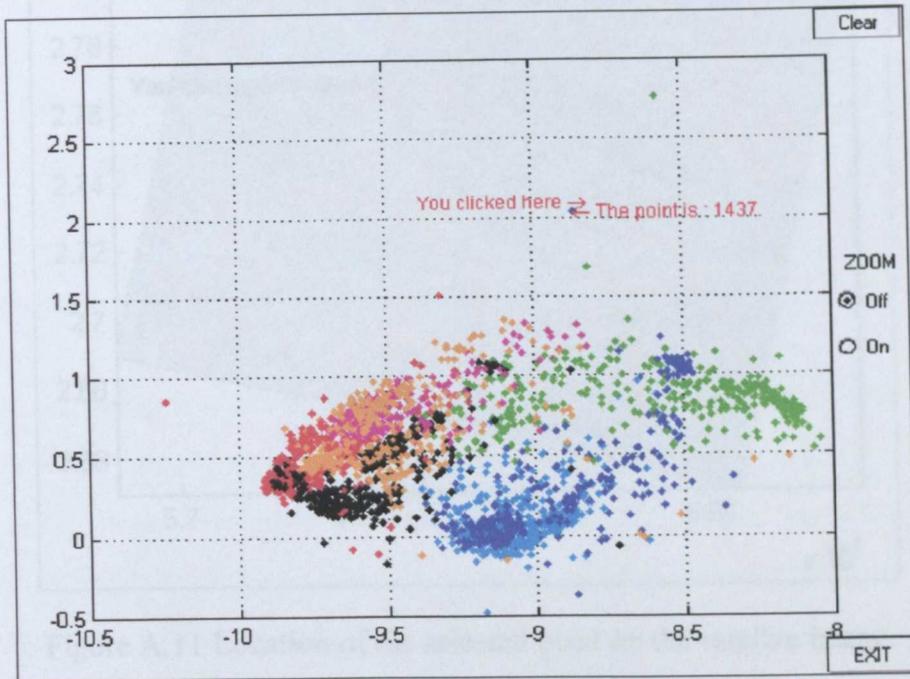


Figure A.9 Locating the pixel closest to the clicked point on the figure.

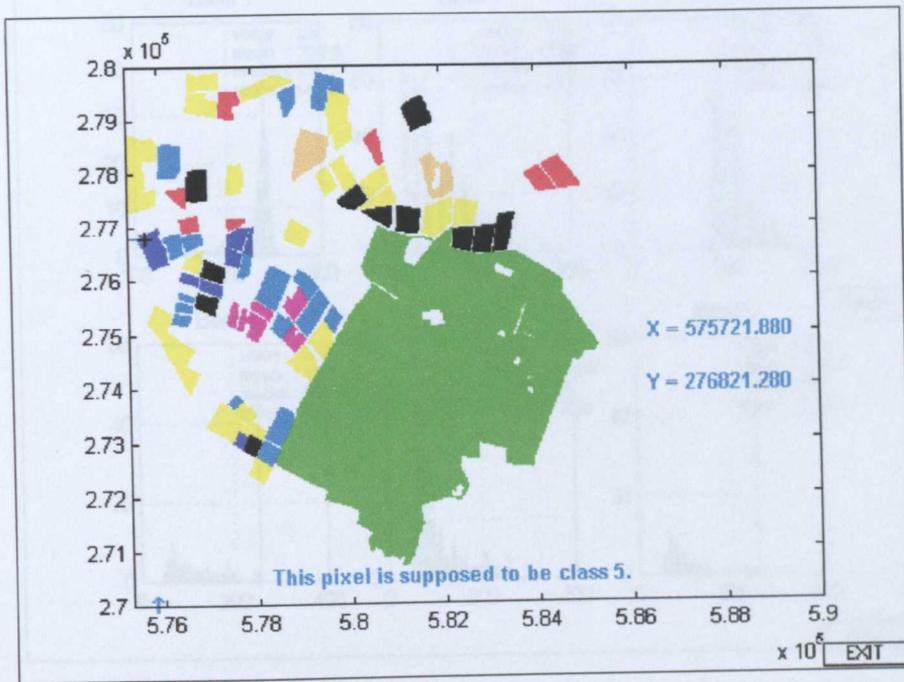


Figure A.10 Location of the selected pixel on ground truth image.

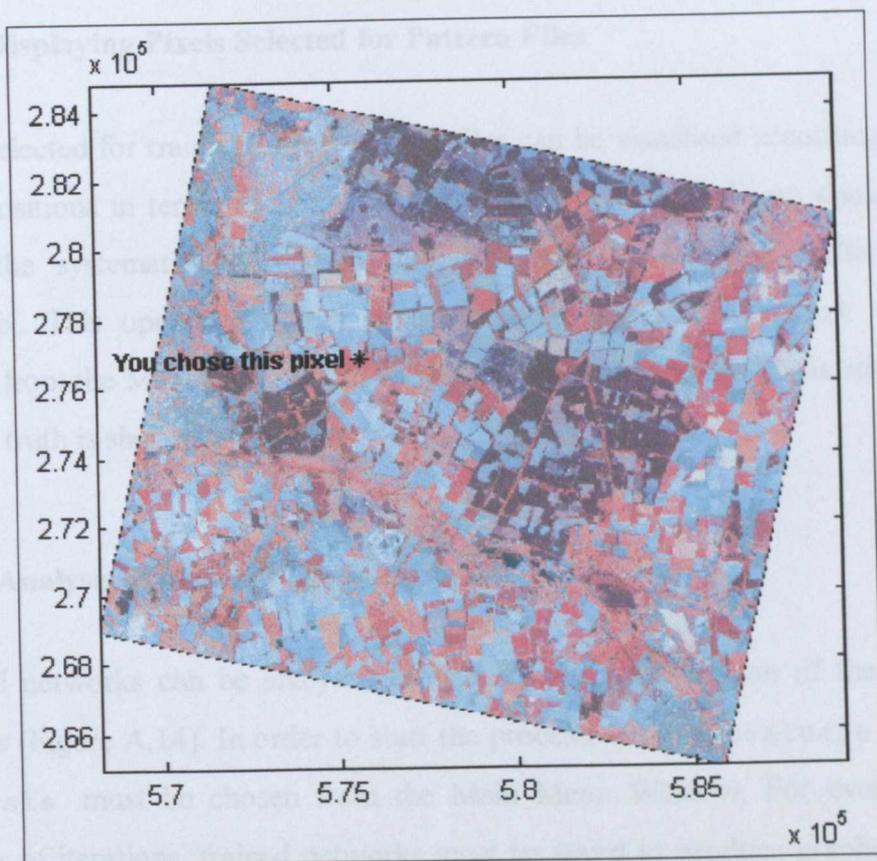


Figure A.11 Location of the selected pixel on the satellite image.

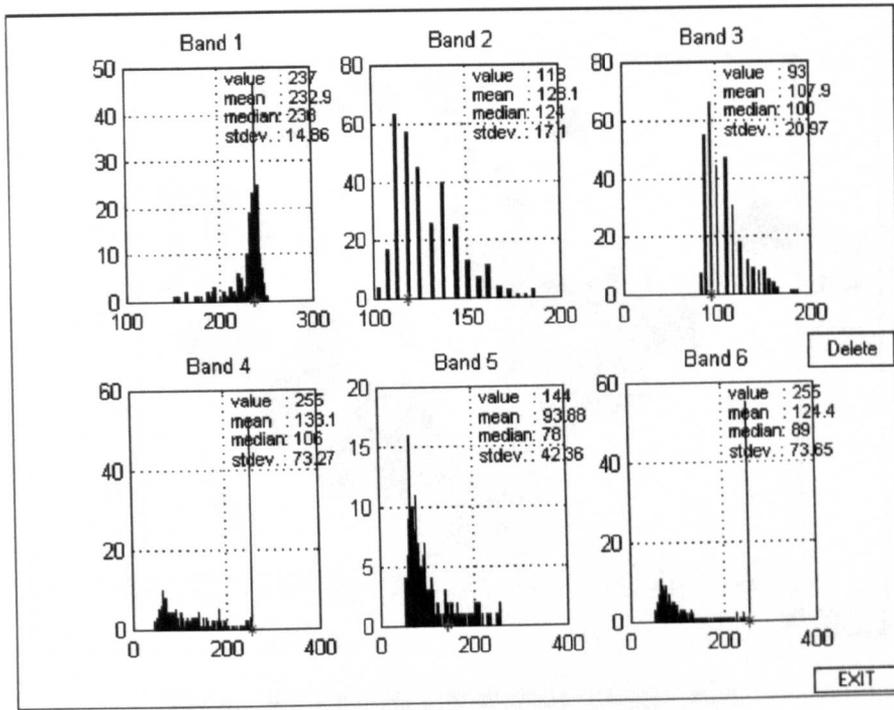


Figure A.12 Analysing the characteristics of the selected pixel using histograms.

### A.5.2 Displaying Pixels Selected for Pattern Files

Pixels selected for training or test pattern files can be visualised according to their exact positions in terms of X and Y coordinates. These pixels are chosen using either the systematic or random sampling strategy discussed earlier in this appendix. This operation is run by clicking the Evaluate|Check Ground Truth from the Main Menu Window. An example showing the pixels selected for ground truth is shown in Figure A.13.

### A.5.3 Analysis of Network Weights

Trained networks can be analysed using a graph representation of the network weights (Figure A.14). In order to start the process, the item Evaluate|Weight Analysis must be chosen from the Main Menu Window. For every certain number of iterations, trained networks must be saved to produce graphs showing the magnitudes of the weights. Red lines are used in the graphs to represent the weights between input and first hidden layer; blue lines represent the weights between first and second hidden layers (if any); and green lines are used for the

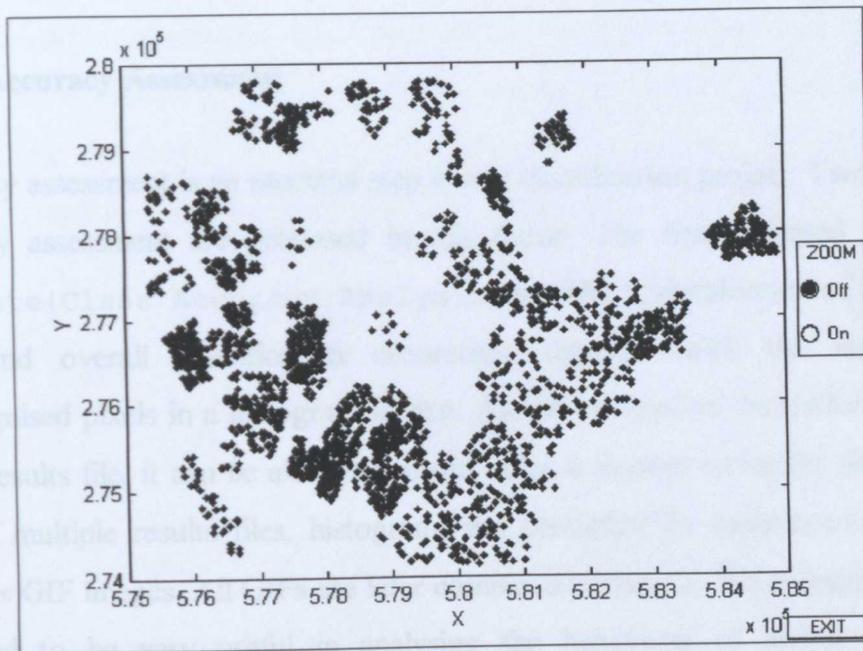


Figure A.13 Pixels selected for ground truth.

weights between the second hidden layer and output layer. When the training process is finished, these graphs are converted to GIF images, and later combined to produce a GIF animation. Such animations considerably help to understand the network training process and the behaviour of artificial neural networks. In addition, displaying the network weights is useful in observing the effect of number of iterations on the network weights.

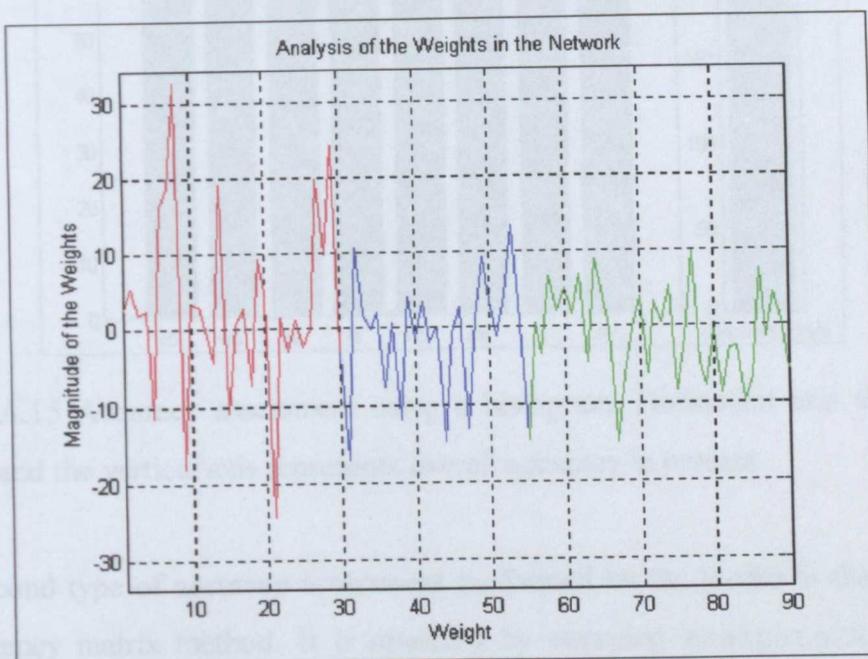


Figure A.14 Analysis of the weights in the network.

### A.5.4 Accuracy Assessment

Accuracy assessment is an essential step in any classification project. Two types of accuracy assessment are proposed in this thesis. The first, initiated from the `Evaluate|Class Accuracy Analysis`, provides a visualisation of individual class and overall classification accuracies, together with the number of unrecognised pixels in a histogram format. As the process can be performed on a single results file, it can be also carried out using a number of results files. In the case of multiple results files, histograms are produced for each results file and saved as GIF images. All GIFs are later combined to form a GIF animation, which is found to be very useful in analysing the behaviour of neural networks, particularly with respect to investigating their learning strategy. A sample histogram is portrayed in Figure A.15.

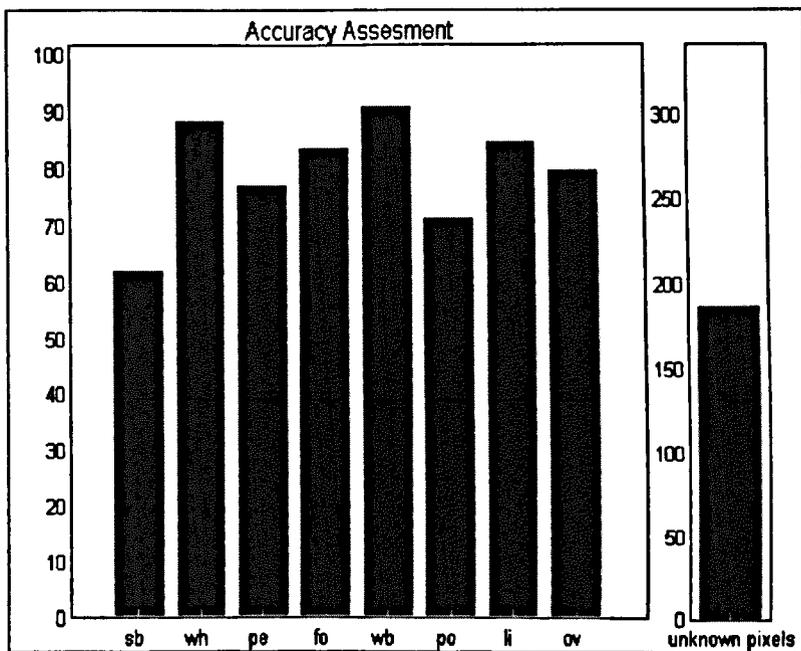


Figure A.15 Accuracy assessment using a histogram. Horizontal axis shows the classes and the vertical axis represents overall accuracy in percent.

The second type of accuracy assessment performed by the toolkit is the standard contingency matrix method. It is operated by choosing `Evaluate|Accuracy Assessment` from the Main Window. Then, the window of the corresponding

program appears, asking for two filenames and the threshold value to be used to evaluate the membership of each pixel in the results file. Two files required by the program are the SNNS results file and the ASCII text file, produced in the sampling stage, including three types of information as described below:

```
w 642 1
f 145 2
t 468 3
s 387 4
o 219 5
p 158 6
b 185 7
```

w, f, t, s, o, p, and b represent the classes of wheat, forest, potato, sugar beet, onion and peas, respectively. Values of 642, 145, 468, 387, 219, 158 and 185 show total number of pixels belong to each class. 1, 2, 3, 4, 5, 6, and 7 correspond to the class number.

The program firstly assesses the memberships of first 642 pixels, which are known to be wheat, and counts the number of pixels that are correctly classified as wheat by the ANN classifier. This assessment is performed for each class individually and the accuracy measures are computed according to the well-known formulae.

* CONTINGENCY MATRIX *								Taskin KAUZOGLU (1999)		
	w	f	t	s	o	p	b			Total
w	610	0	0	0	0	0	0			610
f	1	140	0	0	0	0	1			142
t	0	0	411	55	0	0	0			466
s	0	0	21	352	2	0	0			375
o	0	50	2	0	166	0	1			219
p	1	0	0	1	3	144	0			149
b	2	0	0	0	0	0	182			184
Total	614	190	434	408	171	144	184			2145
Overall : $\kappa$ 90.971 Kappa : 89.057 Z : 122.521 Unr : 59										
Acc. %	95.02	96.55	87.82	90.96	75.80	91.14	98.38			
Con. K	100	98.46	85.30	92.47	73.76	96.41	98.81			

Figure A.16 Accuracy assessment with contingency matrix.

A contingency matrix derived from the ASCII file listed above is given in Figure A.16. It should be noted that the values of the Kappa coefficient are represented by

the values multiplied by 100. Detailed information about contingency matrices and accuracy measures derived from such matrices can be found in section 2.7 of Chapter II.

### A.5.5 Reliability Analysis

Reliability analysis is one of the most important analyses provided by the toolkit in that it gives some insight into the reliability of the test data by investigating each pixel's membership in terms of the output of different networks. Thus, the performance of the classification can be analysed and improved by excluding atypical pixels. In order to perform a reliability analysis (Evaluate | Reliability Analysis for Pixels) several network configurations must be used to learn the characteristics of the same training data. Next, the same dataset must be tested using these networks. Image files in IDRISI ASCII format must be created from the results files. They are later used to examine how many times a pixel is assigned to the same class by the separate networks. The result of this examination is portrayed in a figure similar to Figure A.17.

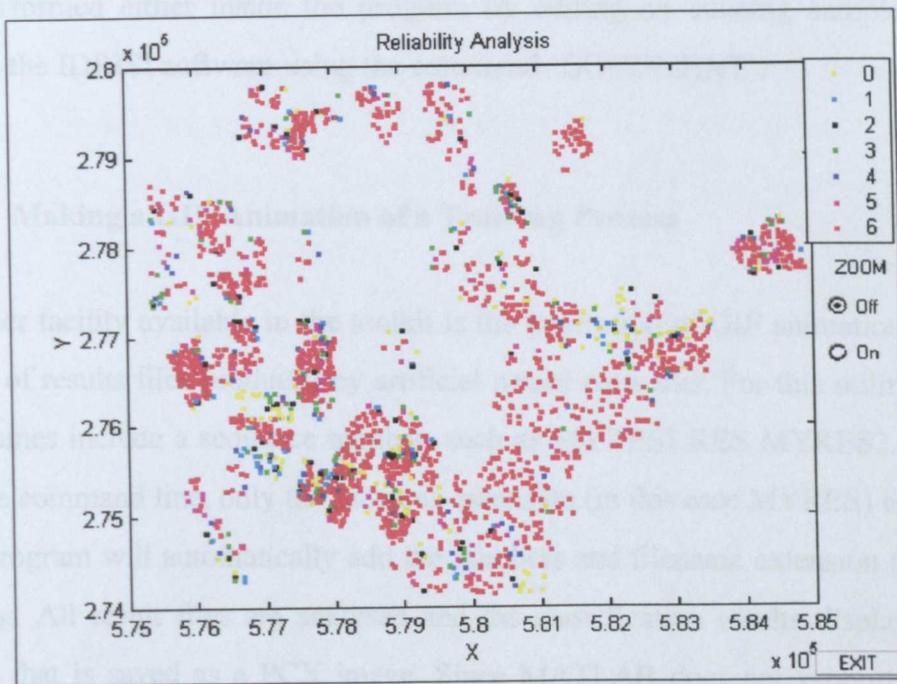


Figure A.17 Reliability analysis for each pixel used in testing. X and Y coordinates represent Easting and Northing.

As can be seen from the figure, the results of ANN classification using six different network structures are assessed and the number of times that the pixel was allocated to the correct class is calculated and displayed in different colours.

#### **A.5.6 Creating IDRISI Image Files**

This section describes the operation to generate IDRISI images from results files produced by SNNS software. This module can be launched from Evaluate|Make Images for IDRISI menu item. With this module, a results file is processed and class labels are written to an ASCII text file that is in a format suitable for IDRISI software. The decision to allocate a pixel to one of the classes is made by analysing the output node activation values. If the highest output value for a given class is higher than a threshold value, then the pixel is labelled as belonging to that class. After creating the ASCII image file, it is essential to have a document file including image-related information, such as the coordinates of the corner points, and number of rows and columns. This step can be performed either inside the program by editing an existing sample file, or inside the IDRISI software using the command 'DOCUMENT'.

#### **A.5.7 Making a GIF Animation of a Training Process**

Another facility available in the toolkit is the generation of GIF animations from a series of results files produced by artificial neural networks. For this utility, results file names include a sequence number, such as MYRES1.RES MYRES2.RES etc. On the command line, only the filename extension (in this case MYRES) is entered. The program will automatically add the numbers and filename extension (.RES) to the tag. All result files are analysed and the classification results displayed on a figure that is saved as a PCX image. Since MATLAB does not support GIF file format, images are first saved in PCX format. They are later converted to GIF format using MULTIGIF.EXE program, which is a freeware downloaded from the internet (<http://www.kfs.org/~abw/code/mgifdl.html>). At the end of the process, all

GIF images are combined in a single GIF animation file. Animagic GIF animation software is then used to read the GIF files. This software can be also used to create a GIF animation from the GIF images that are available in the working directory. Also, Animagic GIF Animator can be directly run by selecting `Run Animator` from the menu. Several GIF animation files, provided on the CD-ROM attached to this thesis, can be viewed using a GIF animator.

## **A.6 Summary**

This appendix describes the use of the toolkit written for visualising multi-dimensional data and analysing the results of artificial neural networks. The toolkit is the main analytical contribution made in this study. A wide variety of visualisation and analysis techniques can be applied to image data using the toolkit. Its use requires minimal amount of background information about the techniques that are employed. It is intended to help new users of neural networks to apply the techniques of their choice to their problems. As it is not intended to be a commercial product, it is not totally user friendly. It is recommended that some practice should be carried out to understand the concepts presented in the toolkit and discussed in this appendix in detail. It should be also noted that it is always possible to use the functions available in MATLAB to apply new techniques to determine the characteristics of the data used and to assess the results of artificial neural networks.

### LIST OF PUBLICATIONS

T. Kavzoglu and C.A.O. Vieira, 1998, An Analysis of Artificial Neural Network Pruning Algorithms in Relation to Land Cover Classification Accuracy. In *Proceedings of the Remote Sensing Society Student Conference*, Oxford, UK. pp. 53-58.

T. Kavzoglu and C.A.O. Vieira, 1998, Using Multitemporal, Multispectral and Multisource Remotely Sensed Data to Classify Crops: An Innovative Approximation. In *Proceedings of the Remote Sensing Society Student Conference*, Oxford, UK. pp. 47-52.

T. Kavzoglu and P.M. Mather, 1998, Assessing Artificial Neural Network Pruning Algorithms. In *Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society*, Greenwich, UK. pp. 603-609.

T. Kavzoglu and P.M. Mather, 1999, Pruning Artificial Neural Networks: An Example Using Land Cover Classification of Multi-Sensor Images. *International Journal of Remote Sensing*, vol. 20, no. 14, pp. 2787-2803.

T. Kavzoglu, 1999, Determining Optimum Structure for Artificial Neural Networks, In *Proceedings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society*, Cardiff, UK. pp. 675-682.

T. Kavzoglu, J. Jaafar and P.M. Mather, 2000, Extraction of Field Boundary Information from Classified Satellite Images. In *Proceedings of GIS Research UK (GISRUK'2000)*, York, UK. pp. 156-160.

T. Kavzoglu and P.M. Mather, 2000, Using Feature Selection Techniques to Produce Smaller Neural Networks with Better Generalisation Capabilities. In *Proceedings of the IGARSS'2000* (CD-ROM), Hawaii, USA.

T. Kavzoglu and P.M. Mather, 2000, The Use of Feature Selection Techniques in the Context of Artificial Neural Networks. In *Proceedings of the 26th Annual Conference of the Remote Sensing Society* (CD-ROM), Leicester, UK.

T. Kavzoglu, J. Jaafar and P.M. Mather, Extraction of Field Boundary Information: Using Satellite Images Classified by Artificial Neural Networks. In *Innovations in GIS 8*. Taylor & Francis. [In press].

T. Kavzoglu and P.M. Mather, The Role of Feature Selection in Artificial Neural Network Applications [Accepted to be published in *International Journal of Remote Sensing*].

