Efficient Implementation of some Statistical and Fuzzy Classifiers for Remote Sensing Data in Context of Geographical Information System

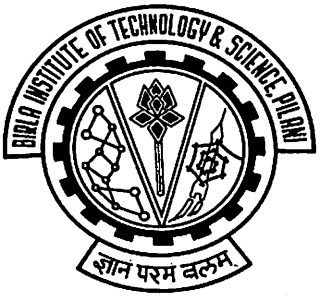
THESIS
Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

By

Mukesh Kumar Rohil

Under the Supervision of

Prof. Rajiv Gupta



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA
2004

This work is dedicated to peaceful applications of Remote Sensing, Artificial Intelligence and Geographical Information Systems

Acknowledgements

First of all, I wish to thank my supervisor Prof. Rajiv Gupta, Dean, Educational Hardware Division, Birla Institute of Technology and Science, Pilani, Rajasthan, India, for his valuable guidance, encouragement and moral support. It has been a great pleasure to be associated with him on this work.

Special thanks are due to Prof. S. Venkateswaran, Vice Chancellor; Prof. L. K. Maheswari, Director; Prof. Ravi Prakash, Dean, Research and Consultancy Division; and Prof. Ashoke K. Sarkar, Dean, Faculty Division-I and Instruction Division; Birla Institute of Technology and Science, Pilani, (Raj.), for giving me an opportunity and encouragement for PhD program. Further, I sincerely acknowledge the encouragement and help received from Prof. K. E. Raman, Deputy Director (Administration); Prof. B. R. Natarajan, Dean, Distance Learning Programs Division; Prof. G. Sundar, Dean, Student Welfare Division and Prof. J. P. Mishra, Unit Chief, Information Processing Unit, Birla Institute of Technology and Science, Pilani, (Raj.), at various stages of the work. Further, I sincerely thank members of the doctoral advisory committee, Dr. Navneet Goyal, Prof. Shan Balasubramanian, Dr. Ankush Mittal, Prof. Satyendra P. Gupta, Prof. R. C. Jain, for their critical remarks and for helping me in improving the work. I received encouragements from my friends namely Mr. K Venkatsubramanian and Dr. Rajiv Kumar. I express sincere thanks to both of them.

I have received help from associates of the Logic Programming Associates Limited, London, namely, Mr. Brian D. Steel, Technical Director; Mr. Clive Spenser, Marketing Director; Ms. Diane Reeve, Administration Director; Mr. Alan Westwood, Software Engineer; and Ms. Vanessa Westwood, Sales and Marketing Personnel. I deeply appreciate the timely help from each of them. Special thanks are due to Mr. Clive Spenser, Ms. Diane Reeve and Ms. Vanessa Westwood for gifting me a CD of WinProlog and some associated tools for this work.

Sincere thanks are also due to Dr. Santosh Kumar Pandey, Lecturer, Languages Group, Birla Institute of Technology and Science, Pilani, Pankaj Bagaria, Paras Jakhoria and Parul Sharma for helping me with proof reading. I deeply thank Mr. S D Pohekar, Research and Consultancy Division, Birla Institute of Technology and Science, Pilani, (Raj.), for providing guidelines for various requirements of the PhD program. Further, I sincerely thank all the persons who directly or indirectly helped me at various stages of the work.

Finally, but most deeply, the author thanks his parents, his wife, his daughter, his son and other family members for their love and moral support during the entire period of this research work with which this work is successfully completed.

(Mukesh Kumar Rohil)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA

C E R T I F I C A T E

This is to certify that the thesis entitled "**Efficient Implementation of some Statistical and Fuzzy Classifiers for Remote Sensing Data in Context of Geographical Information System**" and submitted by **Mukesh Kumar Rohil** ID. No. **1996PHXF004** for award of Ph.D. Degree of the Institute, embodies original work done by him under my supervision.

(Signature in full of the supervisor)
PROF. RAJIV GUPTA
Dean, Educational Hardware Division
Birla Institute of Technology and Science
Pilani – 333 031 (Rajasthan) INDIA

Date: 15/7/2004

List of Mathematical Notations

Notation	Details
ω_k	k^{th} class/group
S_D	Decision space
S_F	Feature space
S_M	Measurement space
s_{TD}	Scaling factor for calculation of Transformed Divergence using already calculated Divergence
$d_{i,j}$	Divergence between class pair i,j
$d_{i,j}^T$	Transformed Divergence between class pair i,j
$J_{i,j}$	Jaffrie-Matusita (JM) distance between class pair i,j
$E_{i,j}$	Payoff for i^{th} alternative and j^{th} state of nature
$SR_{c,b}$	Sigma Range of class c in band b
$k_{(c_1,c_2),b}$	Element of **K**-Matrix in the row of class pair (c_1, c_2) in column of band b
k	Coefficient of overlapping (used to calculate $SR_{c,b}$)
$f_c(\mathbf{X})$	Fuzzy membership function of class c
$\mathbf{F}_{ip}$	Matrix of fuzzy inputs
$\mathbf{F}'_{op}$	Primitive fuzzy output vector
$\mathbf{F}_{op}$	Final vector of fuzzy output
$\lvert\mathbf{M}\rvert$	Determinant of square matrix **M**
$d_i(X)$	Distance function or discrimination function for i^{th} class/group
$g_i(X)$	Discriminate function for i^{th} class class/group
$H_{b,c}$	Highest value of b^{th} component of the pixels of the class c
$L(i,j)$	Loss function for class pair (i, j)
$L_{b,c}$	Lowest value of b^{th} component of the pixels of the class c
$\mathbf{M}^{-1}$	Inverse of matrix **M**
$\mathbf{M}_i$	Mean vector of i^{th} class
$P(w_i)$	Priori probability of i^{th} group

Notation	Details
PA_i	Producer's Accuracy of i^{th} class
$\mathbf{PCM}_i$	Pooled Covariance Matrix of i^{th} class
$P_i(\mathbf{X})$	Probability density function of i^{th} group
$Tr(\mathbf{A})$	Trace of square matrix A
UA_i	User's Accuracy of i^{th} class
$\mathbf{w}_i$	Weight vector of i^{th} group
B	Number of bands (channels or variables)
C	Number of classes/groups
m	Number of classes/groups
n	Number of samples / Number of states
O	Universe of objects
$R_A(x)$	Fuzzy Related on Set A
$\gamma(I)$	Evaluates gamma function at I
$\max_i(E)$	Finds maximum among the entries in expression E with i ranging in appropriate range or set
$\max_i(E)$	Finds minimum among the entries in expression E with i ranging in appropriate range or set
#	Number of
κ	KHAT or Kappa coefficient
$\mu_{b,c}$	b^{th} component of the mean vector of class c
$\sigma_{b,c}$	Standard deviation in band b of the pixels of class c
μ_i	Mean vector of i^{th} group
Σ_i	Covariance (Unbiased) Matrix of i^{th} class
$\mathbf{E}$	Confusion matrix or Error Matrix or Payoff Matrix
$\mathbf{R}$	Corelation Matrix
$\mathbf{V}$	Vector
$\mathbf{X}$	Pixel Vector (unclassified)
$\mathbf{M}'$ or $\mathbf{M}^T$	Transpose of matrix M

List of Abbreviations

Abbreviation	Details or Expanded Form
AA	Proposed statistical supervised method, where MaxMan rule sets coefficient of overlapping, k
AI	Proposed statistical supervised method, where MaxMin rule sets coefficient of overlapping, k
APA	Average Producer's Accuracy (%)
AUA	Average User's Accuracy (%)
BF	Univariate beta distribution based fuzzy supervised classifier
CAT	Class Assignment Time
CT	Classification Time (Total)
ES	Expert Systems
FD	Fuzzy supervised classifier (fuzzy membership is dependent on distance from the centroid of multivariate data)
FM	Fuzzy supervised classifier (fuzzy membership is normally distributed in multivariate)
GIS	Geographical (Geographic) Information System
IA	Proposed statistical supervised method, where MinMax rule sets coefficient of overlapping, k
IE	Improved explicit fuzzy method
II	Proposed statistical supervised method, where MinMin rule sets coefficient of overlapping, k
IP	Image Processing
IRS	Indian Remote Sensing Satellite
KBES	Knowledge-Based Expert Systems
KBS	Knowledge-Based Systems
LIDAR	Light Detection and Ranging
LISS	Linear Imaging and Self-Scanning Sensor
MB	Univariate beta-distribution based MaxMin modulated fuzzy method
MC	Classifier based on the Maximum count of bands in which fuzzy memberships is maximum in given bands
ML	Maximum Likelihood / Two-character Code (used in Figs. 7.1-7.4) for Maximum Likelihood Classifier
MLC	Maximum Likelihood Classifier
MM	Simple MaxMin rule applied on normally distributed fuzzy membership in bands
MR	Fuzzy supervised method where fuzzy membership is expressed in terms of range about the mean in the bands
NIR	Near-Infrared
NN	Proposed fuzzy supervised method, MinMin rule sets coefficient of overlapping, k
NX	Proposed fuzzy supervised method, MinMax rule sets coefficient of overlapping, k
OA	Overall Accuracy (%)
PA	Producer's Accuracy (%)

Abbreviation	Details or Expanded Form
PR	Pattern Recognition
RF	Fuzzy supervised method where fuzzy membership is expressed in terms of range of values in the bands
RS	Remote Sensing
SCT	Statistics Computing Time
SF	Proposed fuzzy supervised method, coefficient of overlapping, k, set to 1
SM	Proposed statistical supervised method, coefficient of overlapping, k, set to 1
TM	Thematic Mapper
UA	User's Accuracy (%)
XN	Proposed fuzzy supervised method, MaxMin rule sets coefficient of overlapping, k
XX	Proposed fuzzy supervised method, MaxMax rule sets coefficient of overlapping, k

List of Figures

List of Tables

Abstract

Last four decades have witnessed successful utilization of image analysis techniques, which involve two prominent and computationally intensive pattern recognition methods known as supervised classification and unsupervised classification, to identify roads, forests, grass-lands, minerals, oil, rocks, soils, landforms, water, social activities, etc. on the earth and to monitor development of facilities and resources, and also to model some of the natural phenomenon like expansion of deserts, soil erosion, floods, volcanoes, etc., from remote sensing data. For this purpose an imaging system classifies remote sensing data in information classes like land cover, land use, utility lines, etc. and an analyst after assessing the accuracy of the classification, merges the output of the classification with a Geographical Information System (GIS). Problem of computationally intensiveness of classifiers is becoming more and more severe due to ever increasing spatial, spectral, radiometric and temporal resolutions of multi-band sensors. So, potential areas of research are reduction of computational efforts and improvements of accuracy for on-line incorporation of classifiers. Researchers are currently attempting to solve this problem in the following ways: 1) modifying the implementations of existing algorithms, 2) developing new classifiers, and 3) using special purpose computer architectures.

In the present research, no explicit attempts are made to propose any computer architecture for computation. However, a relatively new declarative language, Prolog, is used for implementation of classifiers. Prolog, owing features like natural levels of abstractions, backtracking, unification capabilities, declarative and naturally object oriented, provides a natural modeling mechanism and permits easy derivation of new knowledge from existing data. In the present work some fast and robust statistical and fuzzy supervised classifiers are developed and are implemented using *LPA WinProlog* and associated tools. The performance of these classifiers is compared with Maximum Likelihood Classifier (MLC). The present work emphasizes development and implementation of the following proposed supervised classifiers and their comparison with MLC:

1) A new statistical supervised classifier in five variants, which utilizes a parameter, indicating overlapping of class-pairs in various bands, derived from mean and standard deviation. Similarly, a new fuzzy classifier in five variant utilizes fuzzy mean and fuzzy standard deviation.
2) An improved explicit fuzzy supervised classifier with an improved modulation.
3) Two new fuzzy supervised classifiers, one uses beta-distribution, and other uses modulation of parameters of beta-distribution for fuzzy membership function followed by max-min rule.
4) Two new fuzzy supervised classifiers – in first classifier, the initial fuzzy membership is defined by multivariate normal distribution. Then fuzzy mean vectors and fuzzy covariance matrices are estimated to hard classify pixels in the manner of MLC but in fuzzy environment. In second

Abstract

Last four decades have witnessed successful utilization of image analysis techniques, which involve two prominent and computationally intensive pattern recognition methods known as supervised classification and unsupervised classification, to identify roads, forests, grass-lands, minerals, oil, rocks, soils, landforms, water, social activities, etc. on the earth and to monitor development of facilities and resources, and also to model some of the natural phenomenon like expansion of deserts, soil erosion, floods, volcanoes, etc., from remote sensing data. For this purpose an imaging system classifies remote sensing data in information classes like land cover, land use, utility lines, etc. and an analyst after assessing the accuracy of the classification, merges the output of the classification with a Geographical Information System (GIS). Problem of computationally intensiveness of classifiers is becoming more and more severe due to ever increasing spatial, spectral, radiometric and temporal resolutions of multi-band sensors. So, potential areas of research are reduction of computational efforts and improvements of accuracy for on-line incorporation of classifiers. Researchers are currently attempting to solve this problem in the following ways: 1) modifying the implementations of existing algorithms, 2) developing new classifiers, and 3) using special purpose computer architectures.

In the present research, no explicit attempts are made to propose any computer architecture for computation. However, a relatively new declarative language, Prolog, is used for implementation of classifiers. Prolog, owing features like natural levels of abstractions, backtracking, unification capabilities, declarative and naturally object oriented, provides a natural modeling mechanism and permits easy derivation of new knowledge from existing data. In the present work some fast and robust statistical and fuzzy supervised classifiers are developed and are implemented using *LPA WinProlog* and associated tools. The performance of these classifiers is compared with Maximum Likelihood Classifier (MLC). The present work emphasizes development and implementation of the following proposed supervised classifiers and their comparison with MLC:

1) A new statistical supervised classifier in five variants, which utilizes a parameter, indicating overlapping of class-pairs in various bands, derived from mean and standard deviation. Similarly, a new fuzzy classifier in five variant utilizes fuzzy mean and fuzzy standard deviation.
2) An improved explicit fuzzy supervised classifier with an improved modulation.
3) Two new fuzzy supervised classifiers, one uses beta-distribution, and other uses modulation of parameters of beta-distribution for fuzzy membership function followed by max-min rule.
4) Two new fuzzy supervised classifiers – in first classifier, the initial fuzzy membership is defined by multivariate normal distribution. Then fuzzy mean vectors and fuzzy covariance matrices are estimated to hard classify pixels in the manner of MLC but in fuzzy environment. In second

classifier, initial fuzzy membership is defined in terms of the distance of the pixel from the class mean and sum of its distances from all the given classes.

5) Two new fuzzy supervised classifiers, one of them uses range for fuzzy membership in each band and other classifier also uses mean apart from range for defining fuzzy membership function.
6) Two less promising fuzzy classifiers based on the univariate normal distribution, one uses purely max-min rule and other uses count of bands where maximum fuzzy membership exists.

For comparative study, LISS-III data is taken and four cases of training sites are specified on this data. First two cases involve overlapping and mixed classes, third case involves less separable classes and fourth is a case of clear separation of classes. The separability of the classes is specified in the terms of divergence, transformed divergence and Jaffrie-Matusita (JM) distance. The interpretation and analysis of the results of executing above-mentioned 20 algorithms on data taken in 15 combinations of bands (four combinations of single band, six combinations of two bands, four combinations of three bands and one combination of four bands) for the above-mentioned four cases, reveals promising results.

In cases of separable classes, the beta-distribution based classifiers achieve slightly less overall accuracy as compared to MLC but these are much faster than MLC. Three variants of the new proposed fuzzy algorithm achieve considerably higher overall accuracy than MLC, whenever the classes have mixed pixels and are less separable. In the case of less-separations among the classes, the improved explicit fuzzy classifier performs better in terms of both overall accuracy and classification time over all the five variants of the proposed algorithms and the MLC. However, if the classes are more separable, proposed improved explicit fuzzy method performs significantly faster than MLC but overall accuracy is reduced marginally. The classifiers making use of coefficient equal to one to establish overlapping are considerably faster than MLC and achieve overall accuracy comparable to MLC. Further, selected bands with some of the classifiers further improve classification accuracy. Due to modulation of standard deviation and better estimates of the expected extend of classes in various bands the improved explicit fuzzy classifier achieves a significantly high speed up over MLC and still significantly improves overall accuracy of overlapping classes. For fuzzy classifiers, the selection of suitable fuzzy membership function and then its modulation both tasks are important.

The algorithms proposed in this work can be integrated with GIS. As compared to MLC their better performance in GIS is assured even for the overlapping and mixed classes because the accuracy achieved by these methods is less dependent on the training sites. Thus, the different approaches with varying classification accuracy and time provide a wide spectrum to integrate a classifier with GIS and one can choose it depending on the desired accuracy and/or speed.

Table of Contents

CHAPTER 1 *Introduction*

Image analysis is a process of discovering, identifying and understanding patterns that are relevant to the performance of an image-based task. The ultimate objective of image analysis is to determine a high level description of a multi-dimensional scene or phenomenon with a competency level comparable to that of a human. A typical image analysis system performs following operations [Patterson 1990, Sonka et al. 2001]:

1. Image formation, sensing and digitization
2. Local processing, image segmentation and classification
3. Shape formation and interpretation
4. Semantic analysis and description

Imaging systems or image analysis systems closely resemble human visual system which is the most important means by which human perceive the outer world. Sensors in the imaging systems resemble human eye (and other concerned sensory organs) and digital computer functions like human brain. Imaging systems are finding more and more applications due to ever increasing technological developments in the fields of sensors, optics, communication systems, digital cameras, high resolution displays, etc. In the last four decades the use of imaging systems in most of the scientific disciplines and some of socio-economic systems gained high momentum due to their high information content and ever developing technologies to extract and analyze dominant information [Jain 1995], such as to:

1) Identify roads, forests, grasslands, minerals, oil, rocks, soils, landforms, water, social activities, etc. on the earth and to monitor development of facilities and resources, and also to model some of the natural phenomenon [Curran 1988].
2) Identify air-borne targets in defense [Awcock and Thomas 1995].
3) Identify abnormalities and to locate tumors in human body [Awcock and Thomas 1995].
4) Recognize hand written and printed characters in man-machine communications [Jain 1995].
5) Identify criminals through fingerprints and facial features in forensic science [Jain 1995].
6) Verify signatures for security reasons and to establish identity of a person using fingerprints, palm prints, retina images, facial features, etc. [Schalkoff 1989].
7) Control quality in production of printed circuit boards, construction and teaching, etc. [Awcock and Thomas 1995, Gupta et al. 2000].
8) Identify objects and their ranges in robotic vision [Awcock and Thomas 1995].
9) Identify stellar locations in astronomy and to identify weak zones (cavities) in space borne materials through non-destructive testing techniques [Crane 1997].

10) Identify potential control points in an image for subsequent operations like image morphing and warping to establish animation in games and movies [Crane 1997].
11) Scientifically analyze chromatograph, chromosome, wear particle size distribution in lubricating oils, asbestos fiber, scanning electron microscopy, etc. [Awcock and Thomas 1995, Crane 1997].

In this work the part of imaging systems namely the classification of remote sensing data is investigated. Before we discuss about classification, a brief description of remote sensing and computer processing of remote sensing data is presented in the next section.

1.1 Computer Processing of Remote Sensing Data

Remote Sensing (RS) is defined as collection of information about an object (or phenomenon) and its surroundings without physically contacting the same [Swain 1978, Jensen 2000, Lillesand and Kiefer 2000]. An air-borne or a space borne remote sensing platform primarily consist of imaging sensors to acquire information about the existing terrain conditions in the form of digital images in different wavelength ranges in the electromagnetic spectrum. Terrain characteristics, concerned to electromagnetic waves, such as reflectance, scattering, emission in each of the wavelength ranges is collected in the form of digital images and these images are used in identifying the terrain as one of the land cover classes such as soil, water, rock, crop, forest, etc. Further, remote sensing is also used for weather forecasting, atmospheric modeling, and defense application. Remote sensing techniques like LIDAR (Light Detection and Ranging) are used for detecting anomalies in Earth's magnetic field and slides in layers of Earth.

Satellite remote sensing has been popular since the launch of the imaging satellite NIMBUS-1 in August 1964 by NASA under Nimbus satellite program [Szekielda 1988]. This program was aimed to develop an observational system capable of meeting the research and development needs of atmospheric scientists. Since then, applications of remote sensing data for earth resources gained momentum due to the easy availability of image data from a number of remote sensing satellites launched in past four decades. Due to their versatile use, a number of satellites are also planned in future. The information can be derived by the two basic processes involved in remote sensing, namely, data acquisition and data analysis. Data acquisition involves the sub-processes - emitted or reflected energy from energy source, propagation of the energy through atmosphere, the energy interacts with earth surface features, after interaction a part of the energy is re-transmitted through atmosphere, an airborne and/or space sensor senses the re-transmitted energy, the output of sensor is recorded. The data captured by satellite are available in form of images. These images are of two types – 1) Panchromatic and 2) Multispectral. Panchromatic images are comprised of a single spectral band, whereas multi-spectral images are comprised of multiple bands. For example, the PAN sensor mounted on IRS-1D provides panchromatic image and Landsat [Mowle and Dennetur 1991] Thematic Mapper (TM)

images have seven bands: blue, green, red, near-infrared (NIR), mid-infrared-I, thermal-infrared, and mid-infrared-II. These images are in analogue or digital form. For computer processing the analog images are also converted into digital form. These digital images consist of number of pixels (picture elements). By analyzing the pixel values, an image classification system can extract various information from satellite images, such as land use and land cover, hydrograph, water quality, areas of eroded soils, etc. The data analysis involves the following tasks [Jensen 2000, Lillesand and Keifer 2000]:

1. Data is examined using imaging systems, or using various viewing and interpretation devices or by visual interpretation. Here reference data such as maps and census data can help the interpretation.
2. The interpreted information can be compiled in hard copy maps and tables. However, recent trend is to record this compilation in computer files.
3. Finally, the information is used by users for different applications in various fields.

During image interpretation, visual techniques make use of the excellent ability of the human mind to qualitatively evaluate spatial patterns in a scene. However, visual techniques has certain disadvantages:

1. These may require extensive training
2. These are laborious
3. Spectral characteristics are not always fully evaluated due to limited ability of human eye to discern tonal values on an image

In applications, where spectral patterns are highly informative, it is preferable to analyze digital image data using imaging systems rather than visual techniques. Hence, remote sensing images are processed in the following steps:

1. Image pre-processing to reduce geometric distortions, noise, spectral errors.
2. A more discriminating sub-set of features is selected depending on the application. In this stage, only relative features are selected by eliminating irrelevant features and by grouping redundant features. To reduce complexity and dimensionality of the problem, original variables are converted to transformed variables such that the latter are functions of the former. Next step is the classification.
3. Classification of unknown pattern in one of the pre-specified classes is done by discovering some similarity of the unknown pattern with one of the pre-specified patterns (supervised classification) or automatically by natural grouping (unsupervised classification). This step is computationally intensive and it closely resembles pattern recognition process [Patterson 1990].
4. The output of classification stage is various themes, which could be merged with other geographically referenced data, and can be used for different applications or analysis purposes after post processing like corrections in geo-referencing, attaching additional attribute data, etc.

Classification is one of the prominent steps and it is computationally intensive step in image analysis. Hence, success of image analysis depends largely on classification accuracy and the classification time.

1.2 Classification

In classification, unknown pixel is classified into one of the predefined classes using feature selection and some classification technique. Criteria for feature selection and classification is developed with the help of mathematical (or intuitive) theories [Schalkoff 1989]. Based on the way of implementation, classification methods can be further classified as:

1) Supervised,
2) Unsupervised, and
3) Hybrid.

In the supervised approach, classification logic is developed with the help of some truth information (training set). Membership of training samples in each of the classes is known before hand. Training data set is used to develop the feature selection and classification logics. Test set is used to find the classification efficiency of the classifier. Unsupervised classification logic, on the other hand, finds natural groups in the given data. Due to capabilities to find natural grouping, unsupervised classification can be said as an aide to supervised methods [Swain and Davis 1978]. In hybrid classification, the classification logic can involve both supervised and unsupervised approaches. The supervised classification methods have found their practical utilization in interpretation of RS data [Mather 1987]. In the next two sections the development and improvements of the classifiers over time has been discussed briefly. A detailed literature survey of these techniques is given in chapter 2, where objectives of present research are also laid down formally.

1.3 State-of-the-Art of Classification of Remote Sensing Data by Supervised Methods

Basic principle is to assign a class label to an object on the basis of its properties. Maximum Likelihood Classifier (MLC), K-Nearest Neighbour (KNN) [Friedman et al. 1975] and a faster version of KNN [Sethi 1981], Minimum Distance, and Linear Discriminate Functions [Swain and Davis 1978, Devijver and Kittler 1982] utilize only spectral information and their effectiveness depends on [Argialis and Harlow 1990]:

a) The availability of features those are invariant to the expected changes within the pattern classes.
b) The amount of discriminating information contained in the measurements (features),
c) The effective utilization of the information in the classification algorithm, and
d) If the stated assumptions for the method are applicable to the training and test data sets.

For recognition of complex and structured image patterns such as highways, airports, drainage patterns, and landforms, only statistical approaches are not adequate. In order to abstract the information from image effectively, these complex patterns can be recognized with structural pattern recognition methods [Banks 1990, Treash and Amaratunga 2000]. The techniques of structural pattern recognition is also applicable in other fields like Natural Language Processing, which also uses some sort of classification or recognition of patterns permissible by grammar of a particular language [Rohil 2000a].

As spatial resolution increases, spatial context can contribute to efficient interpretation of an image. But, prominent statistical methods are not utilizing the spatial context. So, methods are also proposed to utilize spatial context in addition to spectral information [Haralick et al. 1973, Khazenie and Crawford 1990, Marceau et al.1990, and Lee and Landgrebe 1991]. Further, future of the scope of these prominent statistical and syntactic (structural) methods is doubtful due to their weaknesses to handle high resolution (spatial, spectral, radiometric and temporal) and multi-source data [Middlekoop and Jensen 1991, Serpico and Roli 1995, Solberg et al. 1996, Binaghi et al. 1997] such as:

a) Multi-sensory data
b) Large image sizes due to increased pixel resolution
c) Many data sources such as topography, maps, and local knowledge
d) Nominal non-ordinal data in GIS

Prominent statistical and syntactic (structural) methods are easy to implement, can also take into account the spread and orientation of classes in the multi-dimensional space. However, the use of the prominent statistical and syntactic (structural) methods can be justified only when large sample size is given and the pixels in these samples are spectrally uniform so that the samples can be multivariate normally distributed with a single mode (peak) in the multidimensional histogram [Mather 1987]. These weaknesses of the prominent statistical and syntactic (structural) methods lead to the development of Expert Systems (ES) and Knowledge-Based Systems (KBS) for Image Classification [Goodenough et al. 1987, Jensen 1990, Mchldau and Schiwengerdt 1990, Schenk and Zilberstein 1990, Middlekoop and Jensen 1991, Kartikeyan et al. 1995, Dobson et al. 1996], which are found to be having the following additional qualities:

a) A strong relationship between rules in statistical pattern recognition and rules in Expert Systems
b) Ability to handle non-numeric data
c) Ability to handle information in an intelligent way
d) Ability to cope with uncertainty and probabilities

However, to be practically useful and productive, expert systems and knowledge-based technologies require a large knowledge-base, a knowledge representation scheme, an efficient search and control strategy which can intelligently handle diverse, non-exact, uncertain and inaccurate knowledge or information (even from

contradictory sources), yet these systems are required to be user-friendly and easy in use. These further requirements of knowledge-based systems lead to fuzzy logic based classification methods [Wang 1990, Abe and Lan 1995, Binaghi et al. 1997, Melgani et al. 2000, Bardossy and Samaniego 2002]. As extended in chapter 4, fuzzy logic based classification has many advantages over conventional statistical classifiers like MLC.

1.4 State-of-the-Art of Classification of Remote Sensing Data by Unsupervised Methods

The unsupervised methods simply determine the characteristics of non-overlapping groups of pixels in terms of their attribute values. Two statistical methods, namely, K-means and ISODATA, are widely used for this task. Mather [1987] describes these two methods in detail. Bezdek et al. [1986] developed Fuzzy C-means method, which is an extension of K-means in fuzzy environment and Cannon et al. [1986] provided its efficient implementation. Other unsupervised methods are proposed [Gath and Geva 1989, Delbo et al. 2000, Jia and Richards 2002, Maulik and Bandyopadhyay 2003, Jia and Richards 2003] over time. Use of Markov random fields for unsupervised classification [Nishii 2003] is also made to increase the posterior probabilities of correct classification. An adaptive system for continuous classification from satellite imagery [Saitwal et al. 2003] can make use of two probabilistic neural network classifiers and a context-based predictor to perform continuous cloud classification. However, the major problems with unsupervised classification are:

a) High computational time even for small size of image.
b) Lack of techniques to validate the clusters obtained after unsupervised classification [Windham 1982].
c) Most of the clustering schemes use some user-specified parameters and effect of these parameters is not known to the extent so that clustering can be used for real life problems.

1.5 State-of-the-Art of Classification of Remote Sensing Data by Hybrid Methods

Hybrid classification approach involves elements of both supervised and unsupervised analysis. Hybrid classifiers are used particularly in analyses where there is a high degree of variability in the spectral response patterns for individual cover types present in the data. These conditions are quite common in such applications as vegetation mapping in mountainous areas. Spectral variability comes about both from variation within cover types (i.e. species) and from different site conditions (i.e. soils, slope, aspect) [Lillesand and Kiefer 2000]. High degree of variability, further causes unpredictably many sub-classes in the individual cover types. Because the elements of unsupervised classifiers are not well established to take into account even

small number of cover types, the hybrid methods currently involve more elements of supervised classification than unsupervised classification [Lillesand and Kiefer 2000]. The practical use of hybrid methods is limited to verify uniformity of training sites used in supervised classification.

Further scope of hybrid classifiers depend on techniques available for hierarchical clustering, multi-dimensional scaling, correspondence analysis, and texture analysis [Johnson and Wichern 2002, Shackelford and Davis 2003]. These techniques are not well established even for very small size of data [Banks 1990, Awcock and Thomas 1995, Johnson and Wichern 2002, Shackelford and Davis 2003].

After performing these classifications, we get themes of information classes, which can be used as layers in Geographical Information Systems (GIS) for different applications.

1.6 Geographical Information Systems

A Geographical (or Geographic) Information System (GIS) is a computer system for capturing, storing, querying, analyzing, and displaying geographic (or geo-referenced) data [Chang 2002]. One of the best ways to introduce GIS is to consider the generic types of questions they have been designed to answer and the functions they perform. These include questions about location, patterns, trends and conditions, and functions of manipulation, windowing, spatial analysis (aggregation and classification), measurements (location, perimeter, area, and simple or complex features), non-spatial queries, spatial queries, modeling (simulation of 'what if' scenario), proximity, boundary operations, and logical operations [Holdstock 1998, Boyle et al. 1998, Heywood et al. 2002].

Since the beginning (late 1960s), GIS has been important in natural resource management, such as land use planning, timber management, wildlife habitat analysis, riparian zone monitoring, and natural hazard assessment. In more recent years, GIS has been used in emergency planning, market analysis, facilities management, transportation planning, and military applications [Bowman 1998, Choudhry and Moard 1998, Zheng and Baetz 1999]. Integration of GIS with other technologies such as Global Positioning System (GPS) and the Internet has introduced new applications such as precise farming, interactive mapping on the Internet, and location-based services management [Chang 2000, Rohil 2000c, Gupta 2002, Gupta and Rohil 2002, Heywood et al. 2002].

Data in GIS is organized in such a way that they can be envisioned as digital layers or coverage of information. Each of the coverage is registered to the same common map base; each has a distinct type of features, points, lines or polygons. The GIS stores the spatial data and attribute data. Coverage represents a single theme, such as soils (polygon), streams (line), roads (line) land-use (polygon) and wells (point)

[Holdstock 1998]. The spatial data concepts presented to a user, by GIS, are required to be closely related to the source data, and from these concepts a computer based spatial model may be constructed. In this respect, there tends to be an emphasis on the geometric primitives that characterize spatial data (obtained either as a result of classification or by survey). There are two broad categories of spatial data models encountered in a typical GIS. These are the vector data model, which represents phenomena in terms of the spatial primitives, or components consisting of points, lines, areas, surfaces and volumes, and the raster data model (sometimes referred to as the grid model) that represents phenomena as occupying the cells of a predefined, grid-shaped tessellation. The vector model emphasizes the existence of discrete phenomena, delineated by their boundaries (points, lines and surfaces). The expression vector-model arises from the fact that the primitive spatial entities are themselves usually defined in terms of coordinates. Raster model places emphasis on the contents of grid-cell locations in space and hence regarded as a form of sampling. Raster based spatial models regard space as tessellation of cells, each of which is associated with a record of classification or identity of the phenomenon that occupies the cell. One layer is required to represent one class.

If all layers have same grid-size, the raster model offers easy and fast overlay and arithmetic operations on one or many layers as compared to vector model. However, raster model needs more storage space, more time for display, and is less accurate in geo-referencing as compared to vector model [Chang 2002, Heywood et al. 2002]. But, as most of the data acquisition and display devices operate in raster mode, the raster data has to be converted to vector model to get advantages of vector representation. GIS users can process satellite images (and other remote sensing data) and extract data, by means of classification, for a variety of maps in vector format such as land use and land cover, hydrograph, water quality, and areas of eroded soils. The following characteristics of satellite remote sensing data make them of immense value as a data source for GIS [Curran 1988]:

- Wide availability;
- Low cost (compared to other remotely sensed images);
- Wide area views;
- Time-freezing ability;
- High spectral and spatial resolution;
- Three-dimensional perspective and
- Readily digitized.

Classification plays an important role in many of the functionalities of GIS.

1.7 Importance of Classification in Geographical Information Systems

1.7.1 Geographic Visualization

Geographic visualization has the same objective as exploratory data analysis [MacEachren 1995]. Mainly, four applications of classification are used for geographic visualization i.e. data classification, spatial aggregation, multiple maps in a view, and map comparison.

1.7.2 Data Classification for Creating New Data

Data classification is probably the most common method for map manipulation. The method is useful for data visualization as well as for creating new attribute data and new maps. Classification aggregates data and map features using a classification method and a number of classes. By changing the classification method and the number of classes, the same data can produce different looking maps, each with its own spatial pattern. Cartographers often make several versions of the maps from the same data and choose one -typically one with a good spatial organization of classes – for final map production. For data exploration, classification is most useful in separating spatial data by some descriptive statistics. As a tool for data exploration, data classification can offer additional information by linking a classified map with other attributes that are not used in classification. New attribute data for a vector-based map can be created by selecting a data subset, that is, records that fall within a class according to the classification scheme and assigning a value to the selected data subset.

1.7.3 Data Reclassification

Creating a new raster model by classification is often referred as reclassification, recording, or transforming through look-up tables [Tomlin 1990]. Two methods of data reclassification may be used. The first method is a one-to-one change in which a cell value in the input grid (spatial reference in a layer, and layer in turn maps an geo-referenced feature) is assigned a new value in the output grid. The second method assigns a new value to a range of cell values in the input grid. Floating-point grids can only be reclassified by assigning a new value (may be spatially interpolated or extrapolated) to a range of cell values in the input grid. Reclassification of a floating-point grid results in an integer grid. Raster data reclassification is an important function in data exploration and data analysis. It has the following applications:

- **Data simplification:** Reclassification can group continuous slope values into a set of classes, for example, 1 for slopes of 0%-10%, 2 for slopes 10%-20%, and so forth.
- **Data Isolation:** Reclassification can create a new grid that contains a unique category or value such as irrigated, or a range of values such as slopes of 10%-20%.

- **Data Ranking:** Reclassification can create a new grid that shows the result of the ranking of the cell-values in the input grid. For example, a reclassified grid can show the ranking of 1 to 5, with 1 being least suitable and 5 being most suitable.

1.7.4 Spatial Aggregation

Spatial aggregation is similar to data classification except that it groups data spatially. The levels of geographic units form a hierarchical order, and we can aggregate data from one lower level to a higher level. Displaying aggregated data at different levels offers a view of the effect of spatial scaling. If distance is the primary factor in a study, spatial aggregation can be based on distance measures from points, lines, or areas. Spatial aggregation for raster data means preparing a coarser-resolution grid and computing values of the larger cells. This may use some contextual classifiers for both accurate interpolation and assigning appropriate class to coarser grid.

To integrate diverse sources of data with GIS, data representation schemes for GIS, spatial data analysis, and to solve specific problems in GIS uses of artificial intelligence techniques are investigated, illustrated and proposed [McKeown 1987, Jia 2000, Rohil 2000c]. The recent trend is to use artificial intelligence techniques for all these tasks [MacKeown 1987]. For example, Rohil [2000b] illustrates use of fuzzy logic techniques for remote sensing data analysis and geographic visualization.

1.8 Research Agenda

From the discussions in the earlier sections of this chapter, we can establish that classification plays a fundamental role in interpretation and analysis of remote sensing data. The conventional classifiers like MLC are too intensive computationally to integrate with GIS applications. When using conventional classifiers, the classification accuracy achieved is highly dependent on training data [Mather 1987, Wang 1990] because the conventional classifiers work well only when all the classes are multivariate unimodal normally or gaussian distributed. A need is always felt to develop new classifiers or to modify existing classifiers to enhance accuracy and reduce classification time and that are less dependent on training sites and particular probability distribution [Melgani 2000].

Further, the output of classifiers has to be merged with GIS. However, the need of on-line merger of the themes is also felt apart from off-line due to ever increasing resolution (spatial, spectral, radiometric and temporal) of remote sensing data and diverse applications of GIS. It is required to integrate not only the remote sensing data acquisition process but also the classifiers in GIS.

In the next chapter, these problems and research in these areas are investigated through literature review to lay down formally the objectives of the present research work.

1.9 Organization of the Thesis

A brief introduction to imaging systems, classification, remote sensing, GIS and definition of various terms are given in this introductory chapter. The major problems in imaging systems, especially in classification stage are summarized, the latest research trends, and importance of classification in GIS have been discussed in this chapter. The state-of-the-art of the classification approaches is also discussed briefly.

In chapter 2, detailed literature survey has been presented on the classifiers that apply various approaches – supervised and unsupervised, non-probabilistic and probabilistic statistical classifiers, artificial neural networks, Fourier and wavelet transform, Knowledge-based Systems, Knowledge-based Expert Systems, and Fuzzy Logic – to investigate various problems. First general references of supervised and unsupervised classifiers are discussed and then the more specific references involving improved versions are discussed. The trends of handling varying resolution and multi-source data for classification are also surveyed.

Chapter 3 explains the required theory, mathematical expressions for univariate and multivariate parameters like mean, covariance, standard deviation, various distances, divergence, transformed divergence and Jaffrie-Matusita (JM) distance, and basic and improved versions of the conventional supervised classifiers. Emphasis has also been given on the techniques used for assessment of a classifier. The interpretation and physical meaning of the various parameters is also explained. This chapter also explains concepts to be used for fuzzy classification and its advantages, apart from the a brief review of fuzzy set theory and fuzzy systems including an explanation of how to handle uncertain and vague knowledge to reason in fuzzy environment and hence improved performance of fuzzy classifiers.

Chapter 4 discusses the proposed new statistical and fuzzy supervised classifiers. A statistical supervised classifier is proposed with its five variants assuming equal *a* priori probability and normally distributed – training samples. This chapter explains a technique of handling overlapping in normal probability distributions in univariate data and five variants of this technique are proposed. Thirteen new fuzzy supervised classifiers and the technique of handling overlapping in probability distributions of classes in fuzzy environment are also discussed. Two more, fuzzy methods are developed, but are not included for detailed analysis and interpretation due to their poor performance.

Chapter 5 includes an improved explicit fuzzy method. Here an existing explicit fuzzy method [Melgani et al. 2000] has been improved in the modulation stage and justifications has been given why the proposed

modification gives better results. With the proposed improvement the method is not only made faster but also it demonstrates better accuracy (in overlapping cases) as compared to conventional methods and existing explicit fuzzy method [Melgani et al. 2000].

Chapter 6 explains the data reporting for the investigation of various algorithms and explains the implementation of the computer programs for various tasks. The chapter discusses four different cases of separable and overlapping classes. The method of performing investigations on the 20 algorithms is also explained in this chapter.

Chapter 7 details the interpretation and analysis of the results obtained by executing 20 algorithms on the four cases. In this chapter, the color plots of the results are presented in combinations of 1, 2, 3 and 4 bands. Graphs are presented for overall accuracy, KHAT, statistics computing time, class assignment time and total classification time.

Chapter 8 lists the conclusions of the thesis work.

Chapter 9 explains scope for further investigation and research of the present work. Following the chapter 9, the **List of References** is appended.

Seven appendices follow the list of references. The **Appendix A** explains how to use Fuzzy logic in *LPA WinProlog* programming language. Here only important guidelines and clauses, which are directly related to fuzzy systems, are given. **Appendix B** contains the calculated statistics of all pixels, training pixels and test pixels of the various sites of the four cases. These statistics involve their count, mean vector, covariance matrices, and correlation matrices. The **Appendix C** contains tables containing results about overall accuracy and KHAT achieved by 20 algorithms and time taken by the same when each is executed on the test data represented in combinations of 1, 2, 3, and 4 bands. **Appendix D** contains confusion matrices for performance of classifiers and also contains data about K-matrices, which are used to resolve overlapping cases. As we need overall accuracy from the confusion matrix, which is already tabulated in appendix C, only the confusion matrices for case 1 in combination four bands are given. **Appendix E** contains complete listing of all the programs developed for this work in *LPA WinProlog*. **Appendix F** contains derivation of parameters of univariate beta-distribution. The results of this derivation are finally used in estimating the parameters of beta-distribution in two fuzzy classifiers that are based on univariate beta-distribution. At last, the **Appendix G** lists the research papers (or abstracts) presented, published or communicated for publication from this research work. Among these, one paper has been awarded ISRS OPTO-MECH award for being best paper and another paper is awarded with commendation certificate.

CHAPTER 2 *Literature Survey and Objectives of the Research*

2.1 Introduction

Despite over 40 years of remotely sensing earth from space at optical and other wavelengths, consistent global-scale land cover information still eludes us. Part of this problem lies in the absence of universal definitions of what information is needed and more importantly, what technologies and methodologies can be effectively and efficiently mustered to collect, assimilate, and interpret the information at such scales. The remotely sensed data requires processing to obtain the synoptic view of the earth's surface in the form of classified thematic maps. This is done using a method of categorizing (classifying) digital data into pre-specified groups. Digital methods consider the reflected radiation from different bands as digital numbers located in a feature space and apply pattern recognition [Watanable 1969] techniques to build classification models for discriminating land covers. However, the selection of classification model depends on number of factors like desired accuracy, intended purpose of the results of classification, number of features in use, etc.

2.2 Literature Survey

In general, the classification methods [Tou and Gonzales 1974, Narendra and Goldberg 1977, Venkateswarlu and Raju 1991, Lillesand and Kiefer 2000] can be grouped as statistical (decision theoretic) and syntactic (structural or linguistic). Supervised and unsupervised image classification [Tou 1974, Hord 1982, Dobson et al. 1996] using statistical methods have been found to be very successful with remote sensing data at local to regional scales, generally yielding 85% to 90% correct classification within a scene. When, insufficient observations or documentary evidence of the nature of the land-cover types is available for the geographical area covered by a remotely sensed image, it would not be possible to estimate the mean centers of the classes. Moreover, the number of such classes will be unknown. Unsupervised classification directly strikes at this problem to assist an analyst to estimate the number of natural clusters.

Unsupervised classifiers involve algorithms that examine the unknown pixels in image and aggregate them into a number of classes based on the natural grouping or clusters present in the image values. Unsupervised classifiers do not utilize training data as the basis for classification. The basic premise is that values within a given cover type should be close together in the measurement space whereas data in different classes should be comparatively well separated. The classes that result from unsupervised classification are spectral classes. Because the spectral classes are solely based on the natural groupings in the image values, the identity of

these will not be initially known. The analyst must compare the classified data with some form of reference data (such as larger scale imagery or maps) to determine the identity and informational value of the spectral classes. Thus, in the unsupervised approach we determine spectrally separable classes and then define their informational utility.

There are numerous clustering algorithms (unsupervised classifiers) that can be used to determine the natural spectral groupings present in the data set. Tou [1974], Jensen [1986], and Mather [1987] detail out the two common forms of clustering, K-means and ISODATA approaches. Due to iterated migration of the means and distance calculations K-means is computationally intensive even for small size of images. ISODATA algorithm is even more computationally intensive than K-means algorithm due to expansion and shrinking of the clusters apart from the calculations for migrating mean and the distances. The major problems in unsupervised classification are:

1) High computational time even for small size of image.
2) Lack of techniques to validate the clusters obtained after unsupervised classification [Windham 1982].
3) Most of the clustering schemes use some user-specified parameters and effect of these parameters is not known to the extent so that clustering can be considered practical to generate themes to be merged with GIS.
4) The performance of the clustering algorithms varies with the number of clusters required [Gath 1989].
5) The effects of band reduction and using additional features are not known to extent that the clustering schemes can be considered practical for GIS applications.

GIS uses vector and raster data. Classification has been always important for converting data from raster to vector and vice-versa. High accuracy is always desired for vector data to be practical. As there are no acceptable methods to validate the clustering in context of GIS, the use of unsupervised methods for GIS is neither feasible nor practical with the present state-of-the-art of the unsupervised classifiers. As in some of the GIS applications like navigation, dynamic data is required and which has to be updated online based on the current position. Unsupervised methods, which are computationally intensive, may not provide results in synchronized manner. Moreover, due to limitations of cluster validation certain errors may be propagated during this process. In GIS, all objects are to be geo-referenced. Due to lack of techniques for detecting the magnitude of the errors in the classification of unsupervised classification data, geo-referencing cannot be done acceptably. Use of contextual information can improve upon this accuracy but then complex image processing tasks for establishing and using contextual information are implemented off-line in almost all GIS. The spectral classes resulted from unsupervised methods have to be assigned labels of information classes. This task can be done either by a trained analyst or by an expert system. When making use of

unsupervised classifications for GIS, an expert system or knowledge-based system will be an additional requirement. These additional systems are though practical but are not easy to integrate with GIS.

Due to number of limitations of unsupervised classification, supervised classifiers are recommended for GIS input. Hence, the present work deals only with supervised classifiers. Non-probabilistic statistical supervised classifiers, which do not assume any probability distribution, are distance classifiers (or centroid classifiers) and parallelepiped classifiers [Swain and Davis 1978, Mather 1987]. Further, the distance classifiers use Euclidean distance and Mahalanobis distance measures [Duda and Heart 1973, Mather 1987]. However, these distance methods are computationally intensive and are not able to account for the high correlation and overlapping in classes. Fast distance classifiers [Swain and Davis 1978, Hodgson 1988, Bryant 1989] use computational logics like contextual information, ensemble average [Zaki et al. 1989] or nearest neighbour distance (NDD) [Hodgson 1988], which makes them fast. NDD is distance of a class from its nearest class. During classification a pixel is considered to be nearer to one class by looking if it is within one-half of the NDD or not. However, these approaches can speed up the classification at the most by two folds as compared to Euclidean distance classifier. Except those based on the contextual information approach none of these approaches could add to the classification accuracy. The major problem with NDD is that some pixels are unclassified.

Mahalanobis distance based classifiers [Mather 1987, Zaki et al. 1989] can account for covariance and can provide higher classification accuracy, provided the samples are multivariate unimodal normal distributed. But due to matrix multiplication this method is slow and with increasing dimensionality (number of bands or channels) more computations are required due to need of finding inverse of square matrices (number of such matrices is equal to the number of pre-specified classes). During the classification stage, each pixel's Mahanalobis distance from each of the given classes has to be calculated, which lead to high classification time and increases with the increase in the number of bands used for data representation and number of unknown samples to be classified.

The parallelepiped classifiers use regions, defined in terms of lower and upper bounds of values in the bands for each of the pre-specified training samples [Mather 1987, Lillesand and Kiefer 2000]. These classifiers are also very fast and efficient computationally. However, difficulties are encountered when category ranges overlap. Unknown pixel observations that occur in the overlap areas will be classified as "not sure" or be arbitrarily placed in one (or more) of the overlapping classes. Overlap is caused largely because of the category distributions exhibiting correlation or high covariance are poorly described by the decision regions. In the presence of high covariance, the decision regions fit the category training data very poorly resulting in confusion for a parallelepiped classifier. Unfortunately, spectral response patterns are frequently highly correlated and high covariance is often observed among them.

The most popular probabilistic statistical classifier is Maximum Likelihood Classifier (MLC) [Mather 1987, Venkateswarlu and Raju 1991]. Lee and Landgrebe [1991] and Venkateswarlu and Raju [1991] proposed faster versions of MLC. Venkateswarlu and Raju [1991] used a linear algebra theorem, which defines the ranges of a quadratic form [Eppler 1976, Venkateswarlu and Raju 1991] with the MLC and achieves a speed-up of about 1.5 over conventional MLC, and further speed-ups can be obtained by using lower triangular form, threshold-based logics, and partial sum approach [Bryant 1989, Venkateswarlu and Raju 1991]. Lee and Landgrebe [1991] proposed a multistage MLC in which some groups are terminated at each stage by using a few number of variables reduced by some type of threshold information. Further improvements in speed can be achieved by using feature selection, transformed variables, look-up table operations and by using contextual information in MLC [Swain and Devis 1978]. However, these techniques are not well suited to global-scale application because the multivariate spectral signatures are: 1) sensitive to sensor calibration uncertainty, 2) dependent upon illumination geometry, 3) sensitive to local atmospheric conditions, and 4) may vary from year to year. There have been numerous attempts to classify land cover and other geophysical quantities [Siegal and Gillespire 1980, Gath and Geva 1989, Dobson et al. 1996, Gorte and Stein 1998]. A comparative study of many of the classification attempts [Goldberg and Scilien 1978, Harsanyi and Chang 1994, Paole and Schowengerdt 1995, Dobson et al. 1996] indicates that they are either class oriented or attached by complex and time–consuming procedures making them almost impractical to apply over an entire image. Table 2.1 further summarises some advantages and disadvantages of some selected classification approaches.

Algorithms based on Fourier transforms can classify pixels of an image independently of the classification of their neighboring pixels [Stromberg and Farr 1986]. However, these are computationally intensive. The methods based on wavelet transforms [Simhadri et al. 1998, Fukuda and Hirosawa 1999] can be faster than those based on Fourier transforms and we need not constraint to use only sinusoidal functions, which are computationally intensive due to series expansion. However, a function, used for wavelet transform, providing accurate results to discriminate one set of classes may not achieve higher accuracies when classifying another set of classes with similar overlapping [Gonzalez and Woods 1993].

A faster classification algorithm was introduced for Hyper Spectral Image Classification by reduction of dimensionality using projection approach [Harsanyi and Chang 1994]. Efforts are also made to detect features by using just one band [Kaufman and Remer 1994]. Use of just one band reduces computational time and complexity but only one type of land-cover is detected at a time. In most of the times, this becomes domain specific. Some attempts have been made to obviate this restriction [Abe and Lan 1995].

Though digital methods are fast and accurate when their model assumptions are true, their success has been limited due to the natural variability of the landscape and the over simplicity of their model assumptions. In

practice different land covers have different probability models and the spectral signature of each class varies dynamically according to context. Thus, there is a need to consider the dependencies of features based upon spatial context and other factors external to the scene [Binaghi et al. 1997]. The possibility of improving classification by using spatial context increases with increasing resolution but the current methods, like autocorrelation [Gonzalez and Woods 1993] for finding context information are computationally very intensive. Moreover, to perform classification, algorithms have to use syntactic approach, which may further require computationally intensive graph traversal algorithms. If one wants to use external features like thematic layers for establishing context, the available accuracy of the thematic maps are not consistent with the resolution of the remote sensing data. More errors (of unknown magnitudes) are introduced if these thematic maps are re-sampled to make them consistent with the remote sensing data. The re-sampling operation is also computationally expensive due to interpolations. Hence, using context is though practical but it has its problem to integrate with GIS with current state-of-the-art techniques.

Feature reduction [Mather 1987], most of the times, improves classification accuracy and at the same time it speeds up the classification process. Slight improvement in classification accuracy has been achieved by automated reduction of Hyper-spectral imagery using wavelet spectral analysis [Kaewpijit et al. 2003]. However, feature reduction methods can be used to classify less number of classes and the selection of features depends on particular application. This dependence is mathematically complex to establish [Remer and Kaufman 1994].

Supervised and unsupervised Artificial Neural Network (ANN) architecture has been widely applied in image enhancement, pattern recognition, image segmentation, etc. [Yoshida and Omatu 1994, Baraldi and Parmiggiani 1995, Serpico and Roli 1995, Paole and Schowengerdt 1995, Dobson et al. 1996]. ANN architecture: 1) is much faster than conventional ones [Tou 1974, Jensen 1986, Lillesand and Kiefer 2000] like K-means, and ISODATA, 2) does not require a priori knowledge of the classical statistical distribution, i.e., distribution free [Dobson et al. 1996, Goldberg and Scilien 1978], and 3) does not require a priori specification of the weight that each data source must have on the classification process, i.e., it is preferable for multi-source remote sensing data classification. However, these algorithms are computationally intensive and to make them robust, a large database is required.

To solve the problems posed by ANN, classifications are being tried by Artificial Intelligence methods, like Knowledge Based Expert Systems [Dobson et al. 1996]. To obtain the advantages of both methodologies, i.e., the human experts' method of considering spatial and other ancillary information in the data analysis, led to what is called knowledge based expert system (KBES). There have been few approaches of KBES to analyze remote sensing data for land cover classification. The rule-based, frame based and the blackboard architectures are some of the popular models. However, such classifiers need exposure to structural classes

from a wide variety of ecosystems at various times of year in order to validate the spatial and temporal stability.

A fuzzy rule-based classification algorithm, where fuzzy rule system is derived from training set using simulated annealing as an optimization algorithm is proposed [Bardossy and Samaniego 2002]. This method shows slight improvement, 0.4 %, over MLC in terms of classification accuracy by using only nine rules for four different land cover classes. Many fuzzy-logic based methods have been experimentally tested [Binaghi et al. 1997] and are found to perform better than MLC. Wang [1990] proposed a fuzzy classification method, where multivariate normal distribution is assumed. In Wang's [1990] approach, by using pre-calculated fuzzy mean and fuzzy covariance, the fuzzy membership of the pixels in each of the given classes can be calculated and thus the pixels can be soft classified into the number of classes with varying degree of fuzzy membership. During defuzzyfication an "unknown" pixel is assigned to the class where it has maximum fuzzy membership. Although Wang's approach has advantage of achieving higher classification accuracy, it requires more classification time as compared to MLC. Chen [1999] proposes a further slower method but achieving higher classification accuracy.

Integration of images with GIS plays an important role and some attempts are made in this direction. Sophisticated digital techniques have been developed to tackle specific image processing tasks in the analysis stage and the need for efficient reference to collateral data like maps have led to special handling systems called GIS [Kartikeyan et al. 1995. Jia 2000]. A framework to represent a broad class of problems in the analysis of remote sensing imagery is proposed and an inference engine to tackle such problems is derived [Kartikeyan et al. 1995]. A simple model for spectral knowledge representation is used along with a method for quantification of knowledge through an evidence calculation approach. An automatic knowledge extraction technique [Kartikeyan et al. 1995] to gather knowledge from training samples is also proposed. The proposed approach [Kartikeyan et al. 1995] has the advantages of avoiding commission errors and can incorporate non-spectral and collateral knowledge, while its accuracy using only spectral knowledge is comparable with MLC.

Jia [2000] proposed a tool for integrating KBES with GIS and infer solutions after this integration. If the spatial knowledge is represented using Expert Systems techniques and then reasoning with this knowledge can generate decision-tree, which can be used for classification. However, Jai [2000] used this method for classifying roads based on their utility.

TABLE 2.1

COMPARISON OF SOME SELECTED CLASSIFICATION APPROACHES

SNo	Classifier Approach	Advantages	Disadvantages
1.	Maximum Likelihood Classifier [Mather 1987]	• Can resolve normally distributed classes with diverse covariance • Can adapt to training sites	• Have singularity problems • Requires priori estimation and uniform training samples • With increasing resolution, it is difficult to obtain representative training samples
2.	Context Based [Binaghi et al. 1997]	• Can improve accuracy • Good for detecting both linear and regional features	• Computationally intensive • Requires external features to find context information
3.	Fourier Transform [Stromberg and Farr 1986]	• Classify pixels of image independent of the classification of their neighboring pixels • More suitable for detecting linear features like roads, railway-lines, etc.	• Requires more computing time [Simhadri et al. 1998] • Requires heuristic extensions to real environment
4.	Wavelet Transforms [Fukuda and Hirosawa 1999]	• Good for temporal analysis • Faster than Fourier Transforms	• Computationally intensive • Results may not agree at multi-resolutions
5.	Dimensionality Reduction [Remer and Kaufman 1994]	• Becomes problem domain specific • Computational time and complexity is reduced	• Can detect only less number of classes
6.	Projection Approach [Harsanyi and Chang 1994]	• Applicable to spectrally poor as well as mixed pixels	• Performance varies depending on the particular scenario
7.	Artificial Neural Network [Paole and Schowengerdt 1995]	• Preferable for multi-source remote sensing data • Do not require any priori knowledge of the classical statistics	• Computationally intensive when on serial machine • To make them robust, a need of large database [Melgani 2000]
8.	Knowledge-based or Expert Systems [Kartikeyan et al. 1995]	• Capable of handling multi-source data • Can handle non-numeric data	• Needs large knowledgebase • Need controlled search strategy

SNo	Classifier Approach	Advantages	Disadvantages
9.	Fuzzy Logic Based Approaches in General [McKeown 1987, Wang 1991, Melgani et al. 2000]	• Capable of representing diverse, non-exact, uncertain, and inaccurate knowledge even from contradictory sources • A fast classification time • Improved representation of geographical information • Improved estimates of classification parameters • Identification of cover class components of mixed pixels • Higher overall accuracy as compared to MLC	• User must decide about suitability of fuzzy membership function • Multivariate versions are slow • In case of separable classes only marginal improvement in accuracy is achieved as compared to MLC
10.	Multivariate Fuzzy Classifiers [Wang 1990, Chen 1999]	• Offers higher accuracy than MLC • Can take into account wide spread of fuzzy covariance even in case of overlapping and mixed classes	• Slower than MLC due to initial fuzzy membership calculations by using matrix multiplication • Chen's approach is even more slower due to iterative computations
11.	Explicit fuzzy methods [Melgani et al. 2000]	• Significantly faster than MLC • Offers higher overall accuracy for test samples	• The tested for overlapping and mixed classes only • A parameter obtained by empirical observations is used to control its performance

A classification is needed, which can be applied at regional to global scales and operates at a resolution consistent with the need to retain measures of heterogeneity for certain applications like GIS. This is one of the vital areas where remotely sensed data and their processing assist to determine the various land-covers [Holdstock 1998]. Traditionally, pixel based approaches generate one-pixel-to-one-class mapping, which makes these approaches more difficult as the area of interest contain a mixture of land-cover classes [Duda and Heart 1973, Lillisand and Kiefer 2000]. Moreover, in the traditional methods like MLC or its faster versions, the pixel may not be mapped to any class, if the probability that it belongs to a class is less than some threshold, and hence the pixel is unclassified. In other words, if thresholds are not used, there may be some situations when the maximum likelihood is almost negligible or very close to its likelihood in other classes, even though it is classified into one of the classes. Fuzzy classification has been used to deal with mixed pixel problem that allows every pixel to have a membership value between 0 and 1 in every class. For the classification of multi-spectral remote sensing data, some fuzzy supervised methods have been proposed.

For example, Wang [1990] proposed a fuzzy supervised approach for fuzzy classification by using pre-calculated fuzzy mean vectors and fuzzy covariance matrices. Chi-Fran Chen [1999] proposed fuzzy training data for supervised classification by calculating fuzzy means iteratively to a stable value and then performing hard classification. Farid Melgani et al. [2000] proposed a fuzzy supervised classification method based on MIN fuzzy reasoning rule applied after modulating standard deviation of each class in each band. This method does not only demonstrate 7.2 fold sped up, but also demonstrates an improvement of 2.65% in overall accuracy, over MLC, when applied on test samples. However, when it is applied on training samples, it is slightly inferior (0.83%) to MLC in terms of overall accuracy. A method to classify high-dimensional data, utilizing a two-stage classifier, is also developed and implemented [Tu et al. 1998]. Maulik and Bandyopadhyay [2003] proposed unsupervised fuzzy partitioning using a real-coded variable-length genetic algorithm with automatic evolution of clusters for pixel classification. A hierarchical fuzzy classification approach making use of both spectral and spatial information has been proposed for High-resolution Multispectral data over urban areas [Shackelford and Davis 2003]. Shackelford and Davis's [2003] method achieves 8 % to 11 % larger accuracies than MLC, but the authors present no analysis of classification time. Moreover, this approach is applicable for urban areas. The major problem faced with hierarchical classifiers is that these are computationally intensive and because of different standard being followed for the hierarchical classification of the information to be derived from remote sensing data there is no proper method of validation of the results achieved from these method and the problem become more severe if these methods are used in other type of hierarchical classification scheme. A fuzzy logic classification scheme is proposed [Moore et al. 2001] which is applicable only for ocean color satellite images.

Using current spatial and relational database technologies, all classifiers may not always process spatial data efficiently and provide results in acceptable time, because some tasks are time consuming especially the calculation of class signatures and matrix multiplications. Therefore, the results of the various sub-tasks are stored in some database for subsequent use, which need not be a relational database to cope with other requirements like contextual and secondary information for classification. It may be object relational or deductive databases [Mckeown 1987]. The following are some issues related to the classifiers and database technology being used for integrating remote sensing data with GIS:

1. What should be the structure and organization of data files?
2. How to handle images which have been taken at various resolutions for the same area? How to merge such two images and store in database [McKeown 1987]?
3. How it can be assured that an algorithm developed for low-resolution imagery will also work efficiently and reliably on high-resolution imagery and vice versa [Hassan 1998, Dangermond and Schutzberg 1998]?
4. How to handle dynamic search of a potentially large database to decide what features to display based (primarily) on their spatial location and intrinsic properties [McKeown 1987]?

Present GIS software packages do not support complex image processing tasks, and image processing software packages are off-line which causes frequent delays before inputting data into GIS. Most of the GIS have been developed only to process low-resolution satellite imagery and database storage is suggested efficiently only for low-resolution imagery [Hassan 1998, Dangermond and Schutzberg 1998]. One of the most attractive features of high-resolution satellite imagery is that it will cost much less to acquire data than using alternative terrestrial and extraterrestrial approaches. It is expected that high-resolution data will result in easier and faster data acquisition, lower overall cost, less uncertainty in the information produced by GIS, and the availability of new data that could not be obtained before. For example, using current methods we are able to detect some pixel (and hence geo-referenced area) belonging to a particular class say drainage, or river, but what is expected is to even obtain complete network of drainage and rivers and to find the topology of these features for various analysis, estimates and simulation of 'what-if" scenario, by using artificial intelligence techniques. However, to be able to use these high-resolution remote sensing satellite systems, new algorithms and tools for automated extraction of data from images captured from these systems are needed. As these imagery becomes commonplace, there will be a need of GIS packages capable of processing and extracting data from them by through integration of remote sensing data with GIS using the techniques of fuzzy logic, artificial intelligence, pattern recognition, matrix algebra, etc.

2.3 Objectives of the Research

As revealed from literature survey, the following are the major problems with use of remote sensing data in GIS:

- Absence of universal definition of information to be derived from remote sensing data
- Conventional classifiers are efficient only for images of small sizes and low spectral resolutions
- Due to change of land cover over time, off-line classification is not of much use to analyst Lack of methodologies and techniques to correlate different types of databases
- Techniques to integrate GIS database with classifiers are not developed properly to explore the feasibility of dynamic database

To reduce some of these gaps, the present research is aimed at:

- Development or identification of suitability of some classifiers based on classification time, accuracy, features used, and type of data to be classified, for classification of moderate size and moderate-resolution remotely sensed imagery.
- Investigation of different approaches like fuzzy logic and logic programming such that classifiers can efficiently convert data from one form to another at various stages and infer useful deductions through classification.

CHAPTER 3

Theory – Concepts and Methods

3.1 Introduction to Classification

One of the most basic and essential characteristics of living things is the ability to classify, recognize and identify objects. Certainly all higher-animals depend on this ability for their very survival. Without it, they would be unable to function even in a static unchanging environment. This section discusses the process of computer based recognition and classification, a process whereby computer programs are used to classify and recognize various forms of input stimuli such as visual, acoustic or electromagnetic patterns. In the subsequent sections, we will see that classification of remote sensing data is also similar to this process. We will also see that the conventional Maximum Likelihood Classifier is also developed along these concepts by making use of probability distributions.

3.1.1 Classification and Recognition Process

In artificial (or computer) classification or recognition, essentially four steps (Fig. 3.1), must be performed [Patterson 1990]. In the first step, when sensory devices receive stimuli produced by objects, the values of more prominent attributes and their relations are used to characterize an object in the form of a pattern vector X, as a string generated by some grammar, as a classification tree, a description-graph or some other means of representation. The range of characteristic attribute values is known as the measurement space S_M. In the second step, a subset of attributes whose values provide cohesive object grouping or clustering, consistent with some goals associated with the object classification, are selected to produce high intra-class and low inter-class grouping. The subset represents a reduction in the attributes space dimensionality and hence simplifies the classification process. The range of the subset of attribute values is known as the feature space S_F. In step three, by using the selected attribute values, object or class characterization models are learned and stored for subsequent classification by forming generalized prototype descriptions, classification rules or decision functions. The range of the decision function values or classification rules is known as the decision space S_D. In the fourth step, recognition of familiar objects is achieved through application of the rules learned in step three by comparison and matching of object features with the stored models. Refinements and adjustments can be performed continually thereafter to improve the quality and speed of recognition (Fig. 3.1). There are two basic approaches to the recognition problem [Patterson 1990]:

1. The decision theoretical approach
2. The syntactic approach

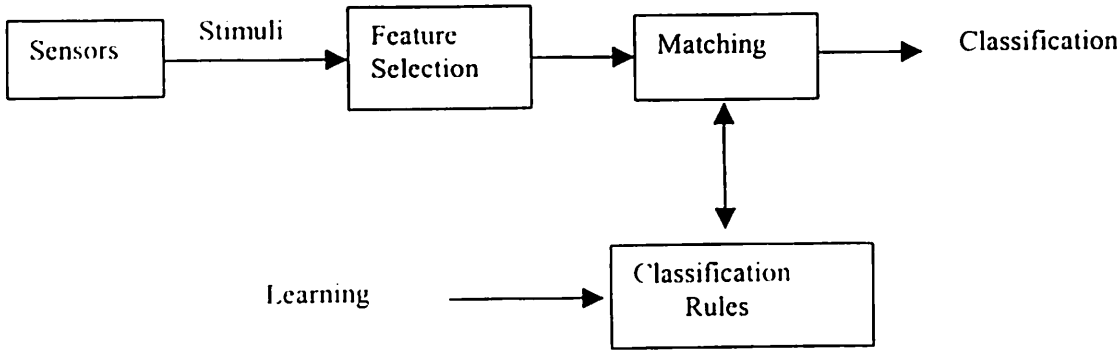


Fig. 3.1 Process of Pattern Classification and Recognition

3.1.2 Decision Theoretical Approach for Classification

In decision theoretic approach, a decision function maps pattern vector **X** into decision regions of decision space S_D as illustrated in the Figs. 3.2 and 3.3. More formally:

1. Given a universe of objects $O = \{o_1, o_2, o_3, o_4, ..., o_n\}$, let each o_i have B observable attributes and rotations expressable as a vector $\mathbf{V} = \{v_1, v_2, v_3, ..., v_B\}$.

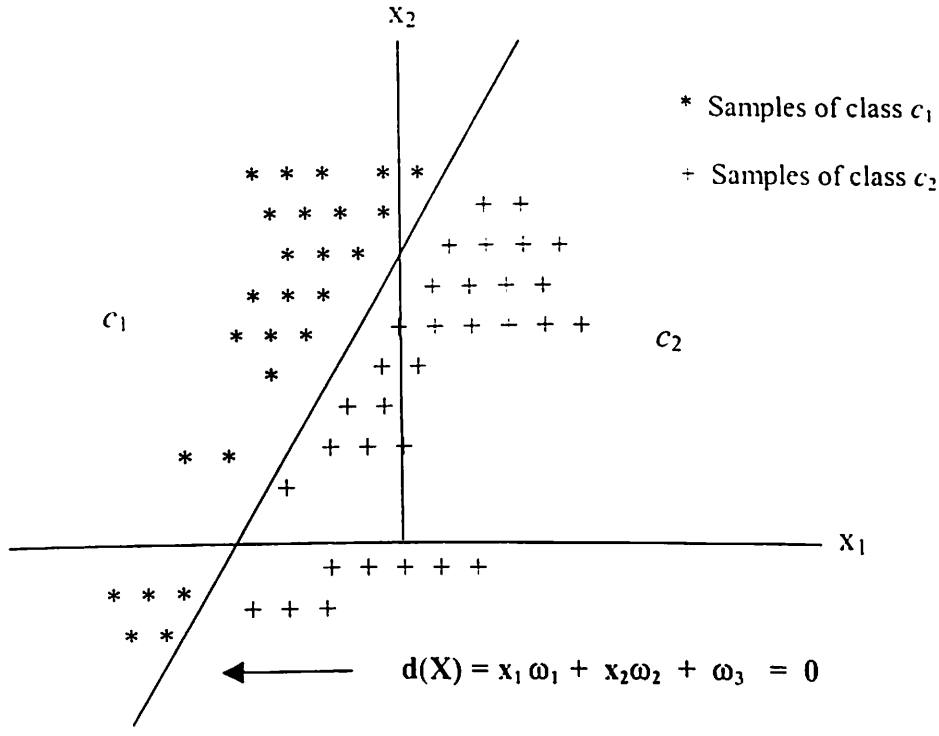


Fig. 3.2 Linear Decision Regions

2. Determine:

   a). A subset $f \leq B$ of the v_i, say $\mathbf{X} = \{x_1, x_2, x_3, ..., x_f\}$

b). For, $C \geq 2$ groupings or classifications of the o_i which exhibit high interclass similarities such as a decision function d(**X**) can be found which partitions S_D into C disjoint regions. The regions are then used to classify each o_i, as belonging to at most one of the C classes.

Determining the feature attributes and decision regions requires stipulating or learning mapping from the measurement space S_M to the feature space S_F and then a mapping from S_F to the classification or decision space S_D, $S_M \rightarrow S_F \rightarrow S_D$ When a line d(x) (or more generally a hyper plane in B-space) can be found that separates classes into two or more groups as in Fig. 3.2, we say that the classes are linearly separable. Classes that overlap each other or surround one another, as in Figs. 3.3 (a) and 3.3 (b), cannot generally be classified with the use of simple decision functions. For such cases, more general non-linear (or piecewise linear) functions may be required [Morrison 1976, Patterson 1990]. Alternatively, some other selection technique (like heuristics) may be needed. In cases where the attribute values are offered by noise or other random fluctuations, it may be more appropriate to define probabilistic decision functions. In such cases the attribute vector **X** are treated as measures of likelihood of class inclusion. For example, using Baye's rule [Morrison 1976, Patterson 1990], one can compute the conditional probability $P(c_i|\mathbf{X})$, that the class of an object o_j is c_i given the observed value of **X** for o_j. This approach requires a knowledge of the prior probability $P(c_i)$, the probability of the occurence of samples from c_i, as well as $P(\mathbf{X}|c_i)$. A decision rule for this case is to choose class c_i if:

$$P(c_i|\mathbf{X}) > P(c_j|\mathbf{X}) \text{ for all } i \neq j \qquad (3.1)$$

A more comprehensive probabilistic approach is one, which is based on the use of a loss or risk Bayesian function where the class is chosen on the basis of minimum loss or risk. Let the loss function $L(i, j)$ denotes the loss occurred by incorrectly classifying an object actually belonging to class c_i as belonging to c_j when $L(i, j)$ is constant for all i,j and $i \neq j$, a decision rule can be formulated using the likelihood ratio defined as $P(\mathbf{X}|c_k)/P(\mathbf{X}|c_j)$ The rule is to choose class c_k whenever the following relation holds $\forall j \neq k$.

$$P(\mathbf{X}|c_k)/P(\mathbf{X}|c_j) > P(\mathbf{X}|c_j)/P(\mathbf{X}|c_k) \qquad (3.2)$$

Probabilistic decision rules may be constructed as either parametric or non-parametric depending on the knowledge of the distribution forms. The decision theoretic approach is apparently simple for two-dimensional case. However, in practice it is much more mathematically involved. We extend this concept in section 3.3 and 3.4 for multidimensional hyperspace. To classify B-dimensional data in one of the m partitions of hyperspace classifiers make use of more advanced mathematical concepts like probabilistic models. For example, Baye's classification [Swain and Davis 1978] on which the conventional classifier, namely, Maximum Likelihood Classifier [Swain and Davis 1978, Mather 1987] is based, makes use of conditional probability.

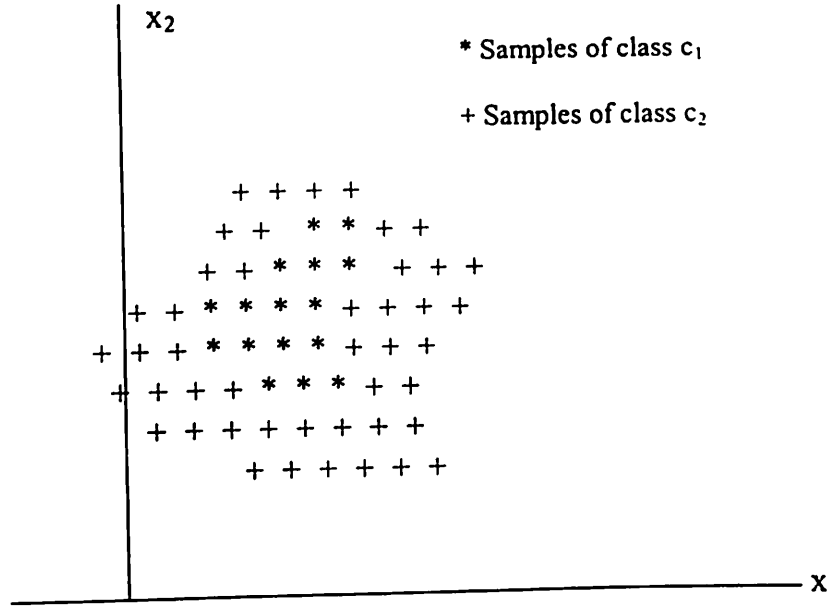


Fig. 3.3 (a) Example of Nonlinearly Separable Classes

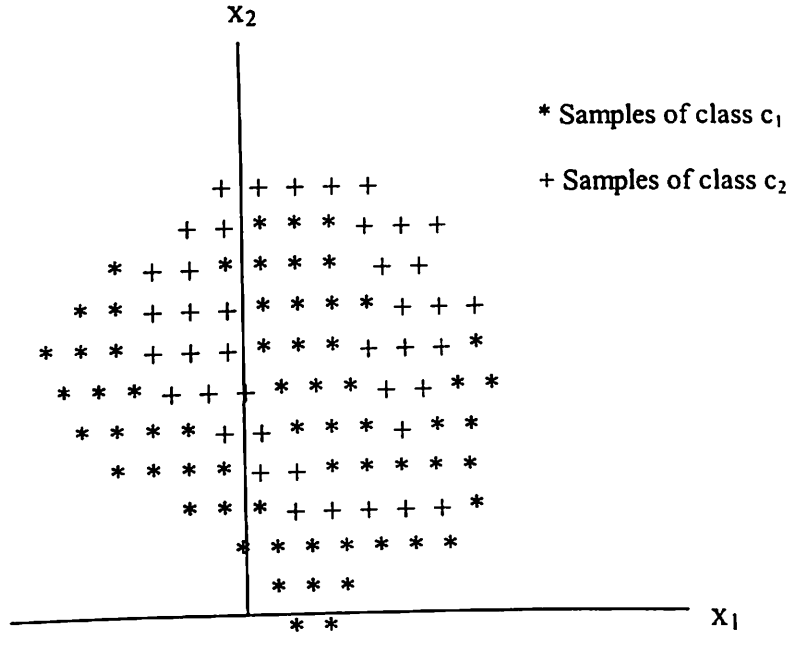


Fig. 3.3 (b) Example of Nonlinearly Separable Classes

3.1.3 Syntactic Classification

The syntactic recognition approach is based on the syntactic "structure" among the object classes. With this approach a grammar or generative grammar, defined in terms of an alphabet of character or terminal words, the vocabulary, is based on shape primitives.

3.2 Digital Image Processing of Multispectral Data

Digital image processing has developed rapidly because of the need to process digital MSS data from Landsat and other satellite based sensors. In principle, digital image processing involves data processing, image enhancement, and image classification, the latter two at times considered as part of image interpretation. Computers are used in various phases of data manipulation and analysis and in more sophisticated information extraction methods. They are also instrumental in integrating remote sensing results into other data sources in models for status assessments and interpretations, decision-making, and operational control. Computer classification eliminates the repetitious judgment of a human operator and increases the detail of the classification. Extensive interaction with a human interpreter is still required because the "ground truth" requirements are very similar to those of photo interpretation. Known examples of the classes selected for classification must be identified for the computer as training sets. The computer abstracts the mean and standard deviation of each of the spectral measurements for all elements in each training set. For maximum error at the 95% confidence level, the minimum sample size per field must vary between 1 to 58 pixels for ground radiometric and airborne MSS measurements [Curran 1988].

Subsequent discussions require multivariate expressions of mean vector, biased estimates of covariance matrix, and unbiased estimates of covariance matrix, which for a population of n samples are given by equations (3.3), (3.4), and (3.5) respectively [Morrison 1976, Hair et al. 2003].

$$\mu = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j \tag{3.3}$$

$$\Sigma_{biased} = \frac{1}{n}\sum_{j=1}^{n}\{(\mathbf{x}_j - \mu)(\mathbf{x}_j - \mu)'\} \tag{3.4}$$

$$\Sigma = \frac{1}{n-1}\sum_{j=1}^{n}\{(\mathbf{x}_j - \mu)(\mathbf{x}_j - \mu)'\} \tag{3.5}$$

In analysis of remote sensing data, unbiased covariance matrix is used, and for simplicity it is called as covariance matrix [Mather 1987]. Another type of matrix, called corelation matrix, **R**, whose elements are related to those of the covariance matrix by equation (3.6), can also be used to establish corelations among bands.

$$r_{i,j} = \frac{v_{i,j}}{\sqrt{v_{i,i}v_{j,j}}} \tag{3.6}$$

where, $v_{i,j} = i,j^{th}$ element of covariance matrix

3.3 Conventional Non-probabilistic Supervised Classifiers

There are three basic steps involved in a typical classification procedure. First, the analyst identifies representative training areas and develops numerical description of the spectral attributes of each land cover type of interest in the scene. Second, in the classification stage, each pixel in the image data set is categorized into the land cover class it most closely resembles. If the pixel is insufficiently similar to any training data set, it is usually labeled unknown. The category label assigned to each pixel in this process is then recorded in the corresponding cell of an interpreted data set. Thus, the multi-dimensional image matrix is used to develop a corresponding matrix of interpreted land-cover category types. After the entire data set has been categorized, in third stage, the results are presented in the output stage. Being digital in character, the result may be used in a number of different ways. Three typical forms of output products are thematic maps, tables of full scene or sub-scene area statistics for the various land-cover classes, and digital data files, which are also amenable to inclusion in a Geographic Information System (GIS). In this later case the classification output become a GIS input. In this section emphasis is given on the conventions statistical methods for supervised classification.

3.3.1 Minimum-Distance-to-Means Classifier

First the mean or average spectral value in each band for each category is determined. These values comprise the mean vector for each category. A pixel of unknown identity may be classified by computing the distance between the value of the unknown pixel and each of the category means. After computing the distances the unknown pixel is assigned to the closest class as given below.

$\mathbf{X} \in \omega_i, i \in \{1,2,3.....m\}$ if

$$g_i(\mathbf{X}) < g_j(\mathbf{X}) \quad \text{for all } j \neq i \tag{3.7}$$

$$g_i(\mathbf{X}) = (\mathbf{X} - \mathbf{M}_i)^T (\mathbf{X} - \mathbf{M}_i) \tag{3.8}$$

where, m = no. of classes

$\omega_i = i^{th}$ class

$\mathbf{M}_i$ = Mean vector of the i^{th} class calculated by using equation (3.3).

If the pixel is farther than an analyst-defined distance from any category mean, it would be classified as "unknown." This strategy is mathematically simple and computationally efficient but it has certain limitations. Most importantly, it is insensitive to different degrees of variance in the spectral response data.

Because of such problems this classifier is not widely used in applications where spectral classes are close to one another in the measurement space and have high variance [Mather 1987, Lillesand and Kiefer 2000]. As

$$g_i(X) = (X^T - M_i)(X - M_i) \tag{3.9}$$

therefore, to speed up equation (3.7) we can use equation (3.10):

$$g_i(X) = M_i^T M_i - 2X^T M_i \tag{3.10}$$

Further sped up can be done by ensemble average classifier [Zaki et al. 1989] as follows:

(i). Calculate $\mathbf{M}_i$

(ii). X will be classified as ω_i if

$$X^T (M_i - M_j) < T_{i,j} \quad \text{for all } j \neq i \tag{3.11}$$

$$T_{i,j} = \frac{(M_i^T M_i)[M_i^T M_i^{-2} M_j] - (M_j^T M_j)[M^T (M_j^{-2} M_i)]}{2 (M_i^T M_i - M_j^T M_j - 2 M_i^T M_j)} \tag{3.12}$$

Note that $T_{i,j} = -T_{j,i}$ (3.13)

3.3.2 Parallelepiped Classifiers

The parallelepiped classifier requires the least information from the user with respect to other classifier described in this chapter. For each of the classes specified, the user provides an estimate of the minimum and maximum pixel values on each of the bands or features. Alternatively a range, expressed in terms of a given number of standard deviation units on either side of the mean of each feature, can be used. The decision rule employed in the parallelepiped classifier is simple. Each unknown pixel is taken in turn and its value on each of the features is checked to see whether it lies inside any of the parallelepipeds. Two extreme cases might occur; in the first, the point in hyperspace representing a particular pixel does not lie inside any of the regions defined by the parallelepipeds. Such pixels are of an unknown type. In the second extreme case the point lies inside just one of the parallelepipeds and the corresponding pixel is therefore labeled as a member of the class represented by that parallelepiped. However, there is the possibility that a point may lay inside two or more overlapping parallelepipeds, and the decision then becomes more complicated. The easiest is to allocate the pixel to the first (or some other arbitrarily-selected) parallelepiped inside whose boundaries it falls. The order of evaluation of the parallelepipeds then becomes of crucial importance, and there is often no sensible rule that can be employed to determine the best order [Lillesand and Kiefer 2000]. The technique is easy to program and is relatively fast in operation. Since, the techniques make use only of the minimum and

maximum values of each feature for each training set it should be realized that (a) these values may be unrepresentative of the actual spectral classes that they allegedly represent, and (b) no information is gathered from those pixels in the training set other than the largest and the smallest in value on each band.

3.3.3 Centroid classifier

The centroid (mean centre or weighted mean) of each training class is computed - it is simply the vector comprising the mean of each of the B features used in the analysis. The Euclidean distance (or any similar type of distance not involving covariance in any form) from each unknown pixel is calculated by using equation (3.8) for each centre in turn and the pixel is given the label of the centre to which its Euclidean distance is smallest. The alteration to the decision rule involving a threshold distance is effectively acknowledging that the shape of region in B-dimensional space that is occupied by pixels belonging to a particular class is important. The third classification technique, Maximum Likelihood Classifier, described in section 3.4.3 begins with this assumption and uses a rather more refined method of describing the shapes of the regions in B-space that are occupied by the members of each class [Jensen 1986, Mather 1987].

3.3.4 Mahanalobis Distance Classifier

For each unknown vector **X** calculate $g_i(\mathbf{X})$ for all $i = \{1, 2, 3, \ldots, m\}$ using equation (3.10)

$$g_i(\mathbf{X}) = (\mathbf{X} - \mathbf{M}_i)^T (\mathbf{PCM}_i)^{-1} (\mathbf{X} - \mathbf{M}_i) \qquad (3.14)$$

and then **X** will in ω_i if $g_i(\mathbf{X}) < g_j(\mathbf{X})$ for all $j \neq i$

$(\mathbf{PCM}_i)^{-1}$ is the inverse of pooled covariance matrix of the i^{th} group

$$\mathbf{PCM}_i = \sum_{i=1}^{m} p(\omega_i) \Sigma_i \qquad (3.15)$$

where, $p(\omega_i)$ = priori probability density function of ω_i

m = number of classes, Σ_i = unbiased covariance matrix of i^{th} class

Mahalanobis classifier retains degree of direction sensitivity via covariance matrix [Swain and Davis 1978, Mather 1987]. However, this is computationally intensive.

3.4 Maximum Likelihood Classifier

3.4.1 Introduction

Maximum Likelihood Classifier (MLC), the conventional and widely used supervised classification method used with remote sensing image data, has roots in Baye's classification [Andrews 1972, Duda and Heart 1973]. MLC algorithm assumes:

1. The frequency distribution of the class membership can be approximated by the multivariate normal distribution. If number of training samples in each class is high, it has been found that the assumption of normality holds reasonably well.
2. Actual frequency distribution of each class is unimodal [Jensen 1986, Mather 1987].

3.4.2 Baye's Classification

Let the spectral classes for an image be represented by $\omega_1, \omega_2, \ldots, \omega_C$ where C is the total number of classes. To determine the class or category to which a pixel at a location $\mathbf{X}$ belongs it is strictly the conditional probability $p(\omega_i \mid \mathbf{X})$, $i = 1, 2, \ldots, C$, that are used. The position vector $\mathbf{X}$ is a column vector of brightness values of the B-dimensional pixel: B as number of bands or channels. It describes the pixel as a point in multi-spectral space with co-ordinates defined by the brightness. The probability, $p(\omega_i \mid \mathbf{X})$, gives the likelihood that the correct class is ω_i for a pixel at position $\mathbf{X}$. Classification is performed according to

$$\mathbf{X} \in \omega_i \text{ if } p(\omega_i \mid \mathbf{X}) > p(\omega_j \mid \mathbf{X}) \text{ for all } j \neq i \qquad (3.16)$$

i.e., the pixel at $\mathbf{X}$ belongs to class ω_i if $p(\omega_i \mid \mathbf{X})$ is the largest. This intuitive decision rule is a special case of a more general rule in which the decision can be biased according to different degrees of significance being attached to different incorrect classifications. The general approach is called Baye's Classification [Andrews 1972, Duda and Heart 1973].

3.4.3 Maximum Likelihood Classifier

Let us denote $\omega_1, \omega_2, \ldots, \omega_m$, m distinct populations (classes) with known d-dimensional probability density functions $p_1(\mathbf{X}), p_2(\mathbf{X}), \ldots, p_m(\mathbf{X})$, respectively. The priori probabilities that an observation is selected from populations $\omega_1, \omega_2, \ldots, \omega_m$ are denoted by $q_1, q_2, \ldots, q_m$ respectively. According to the Bayesian Maximum Likelihood (ML) classification rule, [Swain and Davis 1978, Mather 1986, Venkateswarlu and Raju 1991] assuming equal costs for misclassification, a random pixel vector $\mathbf{X}$ (of dimension B) is classified in ω_k class if

$$q_k p_k(\mathbf{X}) = \max\{q_i p_i(\mathbf{X})\} \ \forall\ i = 1, 2, \ldots, m. \tag{3.17}$$

Assuming equal priori probabilities of all the classes, decision rule (3.17) becomes

$\mathbf{X} \in \omega_k$ if

$$p_k(\mathbf{X}) = \max\{p_i(\mathbf{X})\},\ \forall\ i = 1, 2, \ldots, m. \tag{3.18}$$

In equations (3.17) and (3.18), the probability density $p_k(\mathbf{X})$ is expressed as:

$$p_k(\mathbf{X}) = \frac{1}{(2\pi)^{\frac{B}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{X}-\mu_k)^T \Sigma_k^{-1} (\mathbf{X}-\mu_k)}{2} \right) \tag{3.19}$$

Here, μ_k and Σ_k are the mean vector and covariance matrices of the k^{th} class, and are calculated from the training data. Σ_k is a symmetric positive definite matrix. Σ_k^{-1}, $|\Sigma_k|$ are the inverse and determinant of the covariance matrix Σ_k. By taking logarithms of both sides of equation (3.19) and using rule (3.18), a new classification rule can be represented after eliminating some constant terms as:

$\mathbf{X} \in \omega_k$ if

$$d_k(\mathbf{X}) = \max\{d_i(\mathbf{X})\},\ \forall\ i = 1, 2, \ldots, m. \tag{3.20}$$

where,

$$d_k(\mathbf{X}) = -\ln|\Sigma_k| - Q_k(\mathbf{X}) \tag{3.21}$$

$$Q_k(\mathbf{X}) = (\mathbf{X}-\mu_k)^T \Sigma_k^{-1} (\mathbf{X}-\mu_k) \tag{3.22}$$

In equation (3.21), $d_k(\mathbf{X})$ is generally known as the discriminate function and $Q_k(\mathbf{X})$ as the quadratic term. Calculation of the quadratic term makes the MLC rule computationally inefficient. Calculation of the quadratic term for a group requires $B(B-1)$ multiplications and about same order of additions or subtractions. So classification of a pixel vector $\mathbf{X}$ into one of the m groups necessitates $mB(B+1)$ multiplications. By neglecting the negative sign in the left-hand side of the equation (3.21), the classification rule can be rewritten as:

$\mathbf{X} \in \omega_k$ if

$$d_k(\mathbf{X}) = \min\{d_i(\mathbf{X})\},\ \forall\ i = 1, 2, \ldots, m. \tag{3.23}$$

where,

$$d_k(\mathbf{X}) = \ln|\Sigma_k| + Q_k(\mathbf{X}) \tag{3.24}$$

3.5 Assessment of Classification

For the analysis of the performance of the classification algorithms, both newly developed and conventional MLC, we utilize confusion matrix and the time taken for the classification of test pixels (samples). In this section we present generation and interpretation of confusion matrix, generation and interpretation of the

distance matrices indicating statistical separation among different classes, and the ways of measuring the accuracy of classification achieved by classification algorithms.

3.5.1 Confusion Matrix

The most commonly used method of representing the degree of accuracy of a classification, for m classes, is to build an m x m confusion (or error or contingency) matrix, **E** [Jensen 1986, Mather 1987, Lillesand and Kiefer 2000]. The elements of row i of matrix **E** give the number of pixels which the operator has identified as being members of class i that have been allocated to classes 1 to m by the classification procedure. Element i of row i (the i^{th} element along the main diagonal) contains the number of pixels identified by the operator as belonging to class i that have been correctly labeled by the classifier. The other elements of row i give the number and distribution of pixels that have been incorrectly labeled. Confusion matrix (also called as an error matrix or a contingency table) can be used to evaluate or assess spectral classification. From this matrix we can estimate user's accuracy (UA), producer's accuracy (PA), average user's accuracy (AUA), average producer's accuracy (APA) and overall accuracy (OA). An example of confusion matrix along with assessment of classification is given in Table 3.1. The last two columns and last two rows of the Table 3.1 are extended to the error matrix. These extended rows and columns present the assessment of the classification depicted by the error matrix.

TABLE 3.1

A SAMPLE CONFUSION MATRIX

Class	Crop	Urban	Trees	Sand	Total	PA
Crop	251	24	15	20	310	80.97
Urban	194	172	210	88	664	25.90
Trees	9	30	244	16	299	81.61
Sand	44	37	59	142	282	50.35
Total	498	263	528	266	809	APA=59.71
UA	50.40	65.40	46.21	53.38	AUA=53.85	OA = 52.03

User's accuracy measures commission errors and indicates the probability that a pixel classified into a given category actually represents that category on ground. Producer's accuracy measures omission errors and indicates how well pixels of the given cover-type are classified. Assume **E** represents the error matrix or confusion matrix for the given m classes, then the UA for class c, PA for class c, AUA, APA and OA is given by equations (3.25), (3.26), (3.27), (3.28) and (3.29) respectively [Mather 1987, Lillesand and Kiefer 2000].

$$UA_c = \frac{\mathbf{E}_{c,c}}{\sum_{r=1}^{m} \mathbf{E}_{r,c}} \tag{3.25}$$

$$PA_c = \frac{\mathbf{E}_{c,c}}{\sum_{r=1}^{m} \mathbf{E}_{c,r}} \tag{3.26}$$

$$AUA = \frac{\sum_{c=1}^{m} UA_c}{m} \tag{3.27}$$

$$APA = \frac{\sum_{c=1}^{m} PA_c}{m} \tag{3.28}$$

$$OA = \frac{\sum_{i=1}^{m} E_{i,i}}{\sum_{c=1}^{m}\sum_{r=1}^{m} E_{r,c}} \tag{3.29}$$

From this information, classification errors of omission or commission can be studied. In an ideal case, all non-diagonal elements of the confusion matrix would be zero, indicating no misclassification. Commission errors are represented by non-diagonal elements of the matrix where pixels are classified into a category to which they do not actually belong. Omission errors represent the reverse type of situation. If more than an acceptable percentage of the pixels in a class misclassified, that category may warrant further inspection and retraining.

3.5.2 KHAT (Kappa Coefficient)

KHAT, also called as kappa factor or kappa coefficient [Lillesand and Kiefer 2000], is another measure for classification. Kappa coefficient, κ , is computed from the confusion matrix by equation (3.30).

$$\kappa = \frac{N\sum_{i=1}^{r} x_{i,i} - \sum_{i=1}^{r} (x_{i+} \cdot x_{+i})}{N^2 - \sum_{i=1}^{r} (x_{i+} \cdot x_{+i})} \tag{3.30}$$

r = number of rows in error matrix

$x_{i,i}$ = observation along the major diagonal of the error matrix

x_{i+} = total number of observation in row i

x_{+i} = total number of observation in column i

N = Total number of observations

The calculated value of κ ranges in [-1, 1] and if value is more than or equal to 0.6 we say that classifier is very good. A negative value is worthless. So, the values of κ in the range [0,1] are acceptable. The κ can be multiplied by 100 and interpreted like percentage.

3.6 Separability Measures for Multivariate Normal Spectral Classes

To quantify the separation between a pair of probability distributions as an indication of the degree of overlap, a combination of both the distance between means and measure of standard deviation are required. These quantities are referred to as measures of separability, which implies the ease with which patterns can be correctly associated with their classes by means of statistical pattern classification [Swain and Davis 1978, Mather 1987]. We will discuss here three commonly used measures of separability, namely divergence, transformed divergence, and the Jeffrie-Matusita (JM) distance (or Bhattacharrya distance) [Mather 1987]. Calculations of these distances require multivariate expressions of mean vector and covariance matrix, which for a population of n samples are given by equations (3.3), and (3.5) respectively.

3.6.1 Divergence of a pair of normal distributions

Divergence is a measure of the separability of a pair of probability distributions that has its basis in their degree of overlap. It is defined in the terms of likelihood ratio. Since, spectral classes in remote sensing image data are modeled by multidimensional normal distribution, it can be shown that the divergence, $d_{i,j}$ between i,jth pair of populations can be expressed as given in equation (3.31).

$$d_{i,j} = \frac{1}{2} Tr\{(\Sigma_i - \Sigma_j)(\Sigma_j^{-1} - \Sigma_i^{-1})\} + \frac{1}{2} Tr\{(\Sigma_j^{-1} - \Sigma_i^{-1})(\mu_i - \mu_j)(\mu_i - \mu_j)^T\} \qquad (3.31)$$

where, $Tr\{\}$ is the trace of the subject matrix, μ_i (mean vector of i^{th} class) and Σ_i (covariance matrix of i^{th} class) are given by equations (3.3) and (3.5) respectively.

For a given set of features (any proper subset of these features), the divergence can be calculated for each pair of classes and can be structured as a matrix, called divergence matrix. Alternatively, divergence can be used to reduce the number of features [Mather 1987, Swain and Davis 1978]. However, there are two problems with divergence [Swain and Davis 1978].

1. Its increase is quadratic with separation between spectral classes. It implies that for easily separable classes, a small increase in divergence will lead vastly better classification accuracy whereas in practice this is not the case i.e. there is only very slight increase in the probability and hence the classification accuracy.

2. Easily separable classes will weigh average divergence upwards in a misleading fashion to the extent that sub-optimal reduced feature subsets might be indicated as best.

3.6.2 Transformed divergence of a pair of normal distributions

As the distribution given by $d_{i,j}$ is not well known so a measure called the transformed divergence is used instead. This has the effect of reducing the range of the statistic, the effect increasing with the magnitude of the divergence. Thus, when averages are taken, the influence of one or more pairs of widely separated classes will be reduced. The transformed divergence, $d_{i,j}^{T}$, between the i,j^{th} pair of classes is obtained from (3.32).

$$d_{i,j}^{T} = s_{TD}(1 - e^{-\frac{d_{i,j}}{8}}) \tag{3.32}$$

where, s_{TD} is used to scale the value of transformed divergence. We can use 100 so that we can then treat transformed divergence like percentage and a value of $d_{i,j}^{T}$ of 80 or more indicates good separability of the corresponding classes i and j [Mather 1987].

3.6.3 The Jeffrie-Matusita (JM) distance or Bhattacharrya distance between a pair of normal distributions

The Jeffrie-Matusita (JM) distance (also sometimes called the Bhattacharrya distance), $J_{i,j}$ between a pair of normal distributions i and j, which is seen to be a measure of the average distance between the two class density functions, is given by equation (3.33).

$$J_{i,j} = 2(1 - e^{-\alpha}) \tag{3.33}$$

where,

$$\alpha = \frac{1}{8}(\mu_i - \mu_j)\left[\frac{\Sigma_i + \Sigma_j}{2}\right]^{-1}(\mu_i - \mu_j) + \frac{1}{2}\ln\left[\frac{(\Sigma_i + \Sigma_j)^{\frac{1}{2}}}{|\Sigma_i|^{\frac{1}{2}} + |\Sigma_j|^{\frac{1}{2}}}\right] \tag{3.34}$$

If JM distance is plotted as a function of distance between class-means of a pair of classes it shows a saturating behaviour not unlike that expected for the probability of correct classification. It is asymptotic to 2.0, so that JM distance of 2.0 between spectral classes would imply classification of pixel data into those classes, (assuming they were the only two), with 100% accuracy. This saturating behaviour is highly desirable since it does not suffer the difficulty experienced with divergence.

3.7 Decisions under Uncertainty

It may be inappropriate or impossible to assign probabilities to the several futures identified for a given decision situation. Often no meaningful data are available from which probabilities may be developed. In other instances, the decision maker may be unwilling to assign a subjective probability, as is often the case when the future could prove to be unpleasant. When probabilities are not available for assignment to future events, the situation is classified as decision making under uncertainty. As compared with decision making under certainty and under risk, decisions under uncertainty are made in a more abstract environment [Blanchard and Fabrycky 1998]. In this section, three approaches to take decision under uncertainty are mentioned.

3.7.1 Laplace criterion

In the absence of priori-probabilities one might reason that its possible state of nature is as likely to occur as any other. The rationale of this assumption is that there is no stated basis for one state of nature to be more likely than any other. This is called the Laplace principle. Under this principle, the probability of the occurrence of each future state of nature is assumed to be $1/n$, where n is the number of possible future states [Blanchard and Fabrycky 1998].

3.7.2 MaxMin and MaxMax Criteria

Two simple decision rules are available for dealing with decisions under uncertainty. The first is the MaxMin rule, based on an extremely pessimistic view of the outcome of nature. The use of this rule would be justified if it is judged that nature will do its worst. The second is the MaxMax rule, based on an extremely optimistic view of the future. Use of this rule justified if it is judged that nature will do its best. Because of the pessimism, MaxMin rule, its application will lead to the alternative that assures the best of the worst possible outcomes. If $E_{i,j}$ is used to represent the payoff for the i^{th} alternative and the j^{th} state of nature, the required computation is $\max_i(\min_j(E_{i,j}))$. The optimism of the MaxMax rule is in sharp contrast to the pessimism of the MaxMin rule. Its application will choose the alternative that assures the best of the best possible outcomes. As before, if $E_{i,j}$ is used to represent the payoff for the i^{th} alternative and the j^{th} state of nature, the required computation is $\max_i(\max_j(E_{i,j}))$ [Blanchard and Fabrycky 1998].

3.7.3 Hurwicz Criterion

Because the decision rules, MaxMin and MaxMax are extreme, many decision makers shun them. Most people have a degree of optimism or pessimism somewhere between the extremes. A third approach to

decision making under uncertainty involves an index of relative optimism and pessimism. It is called the Hurwicz rule [Blanchard and Fabrycky 1998]. A compromise between optimism and pessimism is embraced in the Hurwicz rule by allowing the decision maker to select an index of optimism, α, such that α in [0,1]. When $\alpha = 0$, the decision maker is pessimistic about nature, while an $\alpha = 1$, indicates optimism about nature. Once α is selected, the Hurwicz rule requires the computation of $\max_i\{\alpha[\max_j(E_{ij})] + (1 - \alpha)[\min_j(E_{ij})]\}$, where E_{ij} is the payoff for the i^{th} alternative and the j^{th} state of nature [Blanchard and Fabrycky 1998].

Use of fuzzy logic to take decisions under uncertainty has been emphasized in the last two decades. Due to the versatile applications, and radically different approaches of fuzzy systems we discuss fuzzy logic and fuzzy classification in detail.

3.8 Fuzzy Logic Based Classification of Remote Sensing Data

Lotfi A. Zadeh of the University of California, Berkeley, first introduced fuzzy sets in 1965. His objectives were to generalize the notions of a set and propositions to accommodate some type of fuzziness or vagueness. Fuzzy systems can represent complex knowledge and even knowledge from contradictory sources. Fuzzy systems are based on fuzzy logic, which represents a powerful approach to decision making [Berkan and Trubatch 2000, Sonka et al. 2001] which can be used to discriminate various situation for example classification of mixed and overlapping pixels. In cases of overlapping and mixed classes, the current remote sensing image analysis methods are unable to extract the majority of information dormant within digital remotely sensed data and the accuracy levels of image classification are quite unsatisfactory. To improve the analysis it was suggested that studies are required on resolutions of sensor systems, physical principles of remote sensing. Further, it was suggested to improve the data processing algorithms (i.e. including probability distributions in the classifiers). However, so far the effects of knowledge-based and fuzzy logic based classifiers and information representation has not been given its deserved attention in remote sensing data analysis [Wang 1990, Kartikeyan et al.1995]. Fuzzy logic theory directly strikes at this problem. Geographical information (including remote sensing-derived information) is imprecise in nature and impreciseness results from natural variations or arises through original measurements, as well as through data processing. For example, in talking about land cover, a piece of land with sparse grass can be classified into either grassland or soil. Fuzzy set theory provides useful concepts and tools to deal with imprecise information because fuzzy classification estimates the contribution of each class in the pixel and permits partial membership of a pixel in each class. Hence, the fuzzy logic based classification methods have following advantages:

1. A fast classification time, which becomes a factor of great importance with the advent of new generations of sensors providing broad collection of data, for example, HIRIS (High Resolution Imaging Spectrometer) with 192 spectral bands,

2. The strength of the methods is certainly its simplicity and its optimal extraction of the data through the explicit fuzzyfication,
3. Supports modular architecture involving a high flexibility for easily inserting new bands or removing bands without disturbing the remaining parts of the classifier,
4. The implementation of fuzzy methods on artificial neural networks would not pose any difficulty.
5. Representation of geographical information.
6. Partitioning of spectral space,
7. More accurate estimates of classification parameters,
8. Identification of cover class components of mixed pixels, and
9. Higher overall classification accuracy.

Usually, a fuzzy classifier is implemented in three steps – Fuzzyfication, Application of some Fuzzy Reasoning Rule, and Defuzzyfication. During fuzzyfication, fuzzy membership of a B-dimensional pixel is calculated by using a suitable expression. In case of multivariate mode the fuzzy membership of a pixel is calculated in each class and for univariate fuzzy classifiers fuzzy membership of a pixel is calculated for each class c of m given classes in each band b of B bands. In case of univariate classifiers, the output of the fuzzyfication is an m x B matrix of fuzzy inputs, $\mathbf{F}_{ip}$, whose element in row c and column b is the fuzzy membership degree of the pixel for class c in band b. Next stage applies a suitable fuzzy reasoning rule on $\mathbf{F}_{ip}$ to obtain a primitive fuzzy output vector $\mathbf{F}_{op'}$ of order m x 1. Next, to perform fuzzy partition we can normalize elements of $\mathbf{F}_{op'}$ by dividing each element of $\mathbf{F}_{op'}$ by the sum of all the elements of $\mathbf{F}_{op'}$ to obtain fuzzy output vector $\mathbf{F}_{op}$. The final stage, defuzzification, applies another fuzzy rule (or some fuzzy operation) on $\mathbf{F}_{op}$ to hard classify a pixel in one of the classes. In case of multivariate classifiers, the output of fuzzification is a vector of fuzzy inputs, which is then normalized and finally, defuzzification usually applies Max fuzzy rule on normalized fuzzy input vector for performing hard classification.

Before we discuss fuzzy classifiers in detail in the next two chapters, the concepts of theory of fuzzy sets, operations on fuzzy sets and reasoning with fuzzy logic are presented briefly in subsequent sections in that order.

3.9 Theory of Fuzzy Sets

In standard set theory, an object is either a member of a set or it is not. There is no in between. The natural numbers 3, 11, 19, and 23 are members of the set of prime numbers, but 10, blue, cow, and house are not members. Traditional logics are based on the notions that P(a) is true as long as a is a member of the set belonging to class P and false otherwise. There is no partial containment. This amounts to the use of a characteristic function f for a set A, where $f_A(x) = 1$ if x is in A; otherwise it is 0. Thus, f is defined on the

universe U and for all $x \in U$, f: $U \rightarrow \{0,1\}$. We may generalize the notion of a set by allowing characteristic functions to assume values other than 0 and 1. For example, we define the notion of a fuzzy set with the characteristic function u, which maps from U to a number in the real interval [0,1]; that is u: $U \rightarrow [0,1]$. Thus we define a fuzzy set $\tilde{A}$ as follows:

Let U be a set, denumerable or not, and let x be an element of U. A fuzzy subset $\tilde{A}$ of U is a set of ordered pairs $\{(x, u_{\tilde{A}}(x))\}$, for all x in U, where $u_{\tilde{A}}(x)$ is a membership characteristic function with values in [0,1], and which indicates the degree or level of membership of x in $\tilde{A}$. A value of $u_{\tilde{A}}(x) = 0$ has the meaning as $f_A(x) = 0$, that x is not a member of $\tilde{A}$, whereas a value of $u_{\tilde{A}}(x) = 1$ signifies that x is completely contained in $\tilde{A}$. Values of $0 < u_{\tilde{A}}(x) < 1$ signify that x is a partial member of $\tilde{A}$.

Characteristic functions for fuzzy sets should not be confused with probabilities. A probability is a measure of the degree of uncertainty, likelihood or belief based on the frequency or proportion of occurrence of an event. A fuzzy characteristic function, on the other hand, relates to vagueness and is a measure of the feasibility (or ease of attainment) of an event. Fuzzy sets have been related to possibility distributions, which have some similarities to probability distributions, but their meanings are entirely different.

3.10 Operations on Fuzzy Sets

Operations on fuzzy sets are somewhat similar to the operations of standard set theory. They are also intuitively accepted.

A = B if and only if $u_A(x) = u_B(x)$ for all $x \in U$	equality	(3.35)
$A \subseteq B$ if and only if $u_A(x) < u_B(x)$ for all $x \in U$	containment	(3.36)
$u_{A \cap B}(x) = \min_x \{u_A(x), u_B(x)\}$	intersection	(3.37)
$u_{A \cup B}(x) = \max_x \{u_A(x), u_B(x)\}$	union	(3.38)
$u_{A'}(x) = 1 - u_A(x)$	complement set	(3.39)

The single quotation mark denotes the complement fuzzy set, A`. With the above definitions, it is possible to derive a number of properties, which hold for fuzzy sets much the same as they do for standard sets. For example, we have

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	distributivity	(3.40)
$(A \cup B) \cup C = A \cup (B \cup C)$, $(A \cap B) \cap C = A \cap (B \cap C)$	associativity	(3.41)
$A \cup B = B \cup A$, $A \cap B = B \cap A$	commutativity	(3.42)

$A \cup A = A,\ A \cap A = A$ idempotency (3.43)

There is also a form of DeMorgan's laws:

$u_{(A \cap B)'}(x) = u_{A' \cup B'}(x)$ (3.44)

$u_{(A \cup B)'}(x) = u_{A' \cap B'}(x)$ (3.45)

Note however, that $A \cap A' \neq \varnothing$, $A \cup A' \neq U$ since in general for $u_A(x) = a$, with $0 < a < 1$, we have

$u_{A' \cup A}(x) = \max[a, 1 - a] \neq 1$ (3.46)

$u_{A \cap A'}(x) = \min[a, 1 - a] \neq 0$ (3.47)

On the other hand the following relations do hold:

$A \cap \varnothing = \varnothing$ (3.48)

$A \cup \varnothing = A$ (3.49)

$A \cap U = A$ (3.50)

$A \cup U = U$ (3.51)

A number of operations that are unique to fuzzy sets have been defined. A few of more common operations defined on $\tilde{A}$ include Dilation, Concentration and Normalization expressed by the equations (3.52), (3.53) and (3.54) respectively.

$DIL(\tilde{A}) = [u_A(x)]^{1/2}$ for all x in U (3.52)

$CON(\tilde{A}) = [u_A(x)]^2$ for all x in U (3.53)

$NORM(\tilde{A}) = u_A(x) / \max_x\{u_A(x)\}$ for all x in U (3.54)

Note that dilation tends to increase the degree of membership of all partial members x by spreading out the characteristic function curve. The concentration is the opposite of dilation. Normalization provides a means of normalizing all characteristic functions to the same base. In addition to the above a number of other operations have been defined including fuzzification. Fuzzification permits one to fuzzify any normal set. These operations are discussed in next section.

3.11 Reasoning with Fuzzy Logic

The characteristic function for fuzzy sets provides direct linkage to fuzzy logic. The degree of membership of x in $\tilde{A}$ corresponds to the truth-value of the statement x is a member of $\tilde{A}$ where $\tilde{A}$ defines some propositional or predicate class. When $u_A(x) = 1$, the proposition $\tilde{A}$ is completely true, and $u_A(x) = 0$ it is completely false. Values between 0 and 1 assume corresponding values of truth or falsehood. Truth tables are useful in determining the truth-value of a statement. In general, this is not possible for fuzzy logic since there may be an infinite number of truth-values. One could tabulate a limited number of truth-values; say those corresponding to the terms false, not very false, not true, true, very true, and so on. More importantly, it

would be useful to have an inference rule equivalent to a fuzzy modus ponens. For example, let A, A1, B, and B1 be statements characterized by fuzzy sets. Then one form of the generalized modus ponens reads

Premise: x is A1

Implication: If x is A then y is B

Conclusion: y is B1

Although different forms of fuzzy inference have been proposed, we present only Zadeh's original compositional rule of inference. First, recall the definition of a relation. For two sets A and B, the Cartesian product A × B is set of all ordered pairs (a, b), for $a \subseteq$ A and $b \subseteq$ B. A binary relation on two sets A and B is a subset of A × B. Likewise, we define a binary fuzzy relation R as a subset of the fuzzy Cartesian product A × B, a mapping of A → B characterized by the two parameter membership function $u_R(a, b)$. For example, let A = B = $\Re$ the set of real numbers, and let R: = "much larger than". A membership function for this relation might then be defined as

$$u_R(a, b) = 0 \qquad \text{for } a \le b$$

$$= (1 + (a - b)^{-2})^{-1} \qquad \text{for } a > b$$

Now let X and Y be two universes and let A and B be fuzzy sets in X and X × Y respectively. Define fuzzy relations $R_A(x)$, $R_B(x, y)$ and $R_C(y)$ in X, X × Y, and Y respectively. Then the compositional rule of inference is the solution of the relational equation

$$R_C(y) = R_A(x) \circ R_B(x, y)$$

$$= \max_x \{\min\{u_A(x), u_B(x, y)\}\}$$

where, the symbol o signifies the composition of A and B.

In this chapter, we reviewed conventional classifiers and some of their faster versions. The method of assessment of classification is presented and we also investigated qualitatively that fuzzy logic handles uncertainty in more accurate and tolerable ways. We discussed the theoretical concepts of fuzzy logic and it is shown that how fuzzy classification is superior to conventional classification. In the next chapter, proposed statistical and fuzzy supervised classifiers are discussed.

CHAPTER 4 *Statistical and Fuzzy Supervised Classifiers*

4.1 Introduction

In the past three decades, computer-assisted pixel based statistical methods for supervised and unsupervised classification has been successfully applied to the analysis of remote sensing data. In these methods, a set of characteristic measures called features, is extracted from each of the given pattern set (training data). An unknown pattern (pixel) is classified into one of the pattern sets by partitioning the feature space. One of the widely used methods, for such a classification, is Maximum Likelihood Classifier (MLC) [Duda and Heart 1973, Fu 1976, Jensen 1986, Lillesand and Kiefer 2000]. A faster version of MLC is proposed [Venkateswarlu and Raju 1991], which makes use of quadratic terms and range of the quadratic terms [Graybill 1969] to gain speed up. In this work, a new statistical supervised classification algorithm is proposed and implemented in five variants as explained in next two sections.

Traditionally, pixel based approaches generate one-pixel-to-one-class mapping, which makes these approaches more difficult as the area of interest contain a mixture of land-cover classes [Wang 1990, Lillisand and Kiefer 2000]. Moreover, in the traditional methods like MLC, the pixel may not be mapped to any class, if the probability that it belongs to a class is less than some threshold, and hence the pixel is unclassified. Fuzzy classification has been used to deal with mixed pixel problem that allows every pixel to have a membership value between 0 and 1 in every class. For the classification of multi-spectral remote sensing data, some fuzzy supervised methods have been proposed. For example, Wang [1990], proposed a fuzzy supervised approach; Chi-Fran Chen [1999] proposed fuzzy training data for supervised classification by calculating fuzzy means iteratively to a stable value and then performing hard classification. In present work 13 new fuzzy supervised classifiers are developed.

4.2 Proposed Statistical Supervised Classifier

The first stage of this two-stage classifier (implemented in five variants), performs fast classification of some of the pixels by processing in univariate mode. Second stage classifies remaining pixels by using MLC. In this method we use notion of 'coefficient of overlapping', k, value of which can be set by some rule as explained in section 4.3.

A. Computation of signatures

1). Calculate mean vector for each class. Mean vector μ_c of class c containing n pixel, is given by (4.1).

$$\mu_c = \frac{\sum_{i=1}^{n} \mathbf{x}_i}{n} \tag{4.1}$$

2). Calculate covariance matrix for each class. Covariance matrix, Σ_c, of class c with n pixels, is given by (4.2).

$$\Sigma_c = \frac{\sum_{i=1}^{n} (\mathbf{x}_i - \mu_c)(\mathbf{x}_i - \mu_c)^T}{n(n-1)} \tag{4.2}$$

3). Extract standard deviation $\sigma_{c,b}$ of class c in band b from the covariance matrix using (4.3). The diagonal elements of the covariance matrix indicate standard deviation.

$$\sigma_{c,b} = ((\Sigma_c)_{b,b})^{1/2} \tag{4.3}$$

where, $(\Sigma_c)_{r,s}$, is the element of the covariance matrix of class c at row r and column s.

4). Calculate the sigma range $SR_{c,b}$ of class c in band b, equally distant on both sides of the mean, for the given value of the coefficient of overlapping, k, using (4.4).

$$SR_{c,b} = [\mu_{c,b} - k.\sigma_{c,b}, \mu_{c,b} + k.\sigma_{c,b}] \tag{4.4}$$

Set the value of k either by any of the rules explained in section 4.3 or explicitly specify a value for it (e.g. fifth variant in section 4.3, where use $k = 1$ is used).

B. Classification

1). The pixel $\mathbf{x}$, belongs to class c if $\forall$ band b, lower($SR_{c,b}$) $\leq \mathbf{x}_b \leq$ upper($SR_{c,b}$), where the lower and upper functions evaluate the lower (i.e. $\mu_{c,b} - k.\sigma_{c,b}$) and upper (i.e. $\mu_{c,b} + k.\sigma_{c,b}$) bounds of $SR_{c,b}$ respectively, else the pixel is assigned to class 0, indicating that the pixel is unclassified by this rule.

2). For all pixels which are unclassified by the above rule 1), calculate the likelihood of each pixel in the given m classes using (4.5). A B-dimensional pixel is then classified to the class in which it has maximum likelihood

$$P_i(\mathbf{x}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right]}{(2\pi)^{B/2}|\Sigma_i|^{1/2}} \tag{4.5}$$

In actual implementation, if aim is only the hard classification, this step is implemented as implemented in MLC, using *min* expression, as given in equation (3.23). This makes the method faster because here the class-signatures are calculated once and then these are used for classifying all unknown pixels.

4.3 Five Variants of the Proposed Statistical Supervised Classifier

For a pair of classes c_i and c_j, where $i \neq j$ and both c_i and c_j are from a set of m classes, in band b, we define a parameter $k_{(c_i,c_j),b}$ as:

$$k_{(c_i,c_j),b} = \frac{|\mu_{c_i,b} - \mu_{c_j,b}|}{\sigma_{c_i,b} + \sigma_{c_j,b}} \tag{4.6}$$

where,

$\sigma_{c_i,b}$ = Standard deviation of class c_i in band b

$\sigma_{c_j,b}$ = Standard deviation of class c_j in band b

$\mu_{c_i,b}$ = Mean of class c_i in band b

$\mu_{c_j,b}$ = Mean of class c_j in band b

For each class-pair (c_i,c_j), find $k_{(c_i,c_j),b}$ values in each band b of the total bands B, to obtain a matrix and we call this matrix as **K**-Matrix. The **K**-Matrix has the form as given in Table 4.1.

TABLE 4.1

FORMAT OF **K**-MATRIX FOR C CLASSES AND B BANDS

Class Pairs ↓	Bands			
	1	2	...	B
(c_1,c_2)	$k_{(1,2),1}$	$k_{(1,2),2}$	...	$k_{(1,2),B}$
(c_1,c_3)	$k_{(1,3),1}$	$k_{(1,3),2}$	...	$k_{(1,3),B}$
...	...	...	...	...
(c_{c-1},c_c)	$k_{(c-1,c),1}$	$k_{(c-1,c),2}$	...	$k_{(c-1,c),B}$

The particular choice for the expression for $k_{(c_i,c_j),b}$ can be derived as follows:

For two univariate normal distributions, L and R, with mean μ_L and μ_R, and standard deviation σ_L and σ_R respectively, and defined in the range $[L_1, L_2]$ and $[R_1, R_2]$ respectively, we get

$$L_2 - \mu_L \cong 3\sigma_L \tag{4.7}$$

$$\mu_R - R_1 \cong 3\sigma_R \tag{4.8}$$

$$\mu_R - \mu_L = k_L\sigma_L + k_R\sigma_R \tag{4.9}$$

Assume, $k_L = k_R = k$ then

$$(\mu_R - \mu_L) = k\,(\sigma_L + \sigma_R) \tag{4.10}$$

or

$$k = (\mu_R - \mu_L) / (\sigma_L + \sigma_R) \tag{4.11}$$

If k is less than 3 then some overlapping between the pair of distributions is certain [Martin 1976]. So, in general larger the k, less overlapping is observed. To increase k we must either increase μ_R or decrease μ_L, σ_L or σ_R. Hence, the ratio expressed by equation (4.11) is known as 'coefficient of safety'. However, due to consistency reasons we will call it as 'coefficient of overlapping' for subsequent discussions in this work.

We propose five variants by applying the following rules:

A. MaxMax (AA) rule

Calculate k by using (4.12) and use this value of k in the proposed method.

$$k = \max_{(c_i, c_j)}\left(\max_b\left(k_{(c_i, c_j), b}\right)\right) \tag{4.12}$$

B. MaxMin (AI) rule

Calculate k by using (4.13) and use this value of k in the proposed method.

$$k = \max_{(c_i, c_j)}\left(\min_b\left(k_{(c_i, c_j), b}\right)\right) \tag{4.13}$$

C. MinMax (IA) Rule

Calculate k by using (4.14) and use this value of k in the proposed method.

$$k = \min_{(c_i, c_j)}\left(\max_b\left(k_{(c_i, c_j), b}\right)\right) \tag{4.14}$$

D. MinMin (II) Rule

Calculate k by using (4.15) and use this value of k in the proposed method.

$$k = \min_{(c_i, c_j)}\left(\min_b\left(k_{(c_i, c_j), b}\right)\right) \tag{4.15}$$

E. Use $k = 1$ in the proposed method for the fifth variant (SM).

The two characters (AA, AI, IA, II) appearing against the name of each of the rules given in *A, B, C* and *D* above denotes the abbreviation of the first, second, third and fourth variant of the proposed method respectively. For each class, in each band we take ranges, which are spread about mean by a quantity equal to k times standard deviation of the class in the corresponding band. The four variants abbreviated as AA, AI, IA, and II use MaxMax, MaxMin, MinMax, and MinMin rules respectively to calculate k as explained above. The fifth variant, abbreviated as SM, uses $k = 1$. Next, thirteen new fuzzy supervised classifiers are presented in subsequent sections.

4.4 Proposed Multivariate-Univariate Fuzzy Supervised Classifier and its Five Variants

In the first stage of this two stage fuzzy approach, which is implemented in five variants, the algorithm performs fast classification of some of the pixels by processing in univariate mode. The second stage involves classification of remaining pixels by fuzzy partition of the multi-spectral space.

A. Computation of signatures

1). Calculation of initial fuzzy membership, for the given cover classes. The initial fuzzy membership function $f_c(\mathbf{x})$ of class c is given by (4.16).

$$f_c(\mathbf{x}) = \frac{P_c(\mathbf{x})}{\sum_{i=1}^{m} P_i(\mathbf{x})} \tag{4.16}$$

where,

$P_i(\mathbf{x})$ is given by equation (4.5).

2). Calculate fuzzy mean vector for each class. Fuzzy mean vector μ_c^* of class c containing n pixels, is given by (4.17).

$$\mu_c^* = \frac{\sum_{i=1}^{n} f_c(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} f_c(\mathbf{x}_i)} \tag{4.17}$$

3). Calculate fuzzy covariance matrix for each class. Fuzzy covariance matrix, Σ_c^*, of class c with n pixels, is given by (4.18).

$$\Sigma_c^* = \frac{\sum_{i=1}^{n} f_c(\mathbf{x}_i)(\mathbf{x}_i - \mu_c^*)(\mathbf{x}_i - \mu_c^*)^T}{\sum_{i=1}^{n} f_c(\mathbf{x}_i)} \tag{4.18}$$

4). Extract fuzzy standard deviation $\sigma_{c,b}$ of class c in band b from the fuzzy covariance matrix using (4.19). The diagonal elements of the fuzzy covariance matrix indicate fuzzy standard deviation.

$$\sigma_{c,b}^* = ((\Sigma_c^*)_{b,b})^{1/2} \tag{4.19}$$

where, $(\Sigma_c^*)_{r,s}$ is the element of the fuzzy covariance matrix of class c at row r and column s.

5). Calculate the sigma range $SR_{c,b}$ of class c in band b, equally distant on both sides of the fuzzy mean, for the given value of the coefficient of overlapping, k, using (4.20).

$$SR_{c,b} = [\mu^*_{c,b} - k.\sigma^*_{c,b}, \mu^*_{c,b} + k.\sigma^*_{c,b}] \quad (4.20)$$

Set the value of k either by any of the rules explained in section 4.3 or explicitly specify a value for it, say k = 1, as in fifth variant (abbreviated as SF).

B. Classification

1). The pixel x, belongs to class c if $\forall$ band b, lower ($SR_{c,b}$) $\leq x_b \leq$ upper ($SR_{c,b}$), where the lower and upper function evaluate the lower (i.e. $\mu^*_{c,b} - k.\sigma^*_{c,b}$) and upper (i.e. $\mu^*_{c,b} + k.\sigma^*_{c,b}$) bounds of $SR_{c,b}$ respectively, else the pixel is assigned to class 0, indicating that the pixel is unclassified by this rule.

2). For all pixels which are unclassified by the above rule 1), calculate fuzzy membership of each pixel in the given m classes, using (4.21). A B-dimensional pixel is then, hard classified to the class in which it has maximum fuzzy membership.

$$f_c(\mathbf{x}) = \frac{P^*_c(\mathbf{x})}{\sum_{i=1}^{m} P^*_i(\mathbf{x})} \quad (4.21)$$

where,

$$P^*_i(\mathbf{x}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}-\mu^*_i)^T \Sigma^{*-1}_i (\mathbf{x}-\mu^*_i)\right]}{(2\pi)^{B/2}\left|\Sigma^*_i\right|^{1/2}} \quad (4.22)$$

4.4.1 Five Variants of the Proposed Multivariate-Univariate Fuzzy Supervised Classifier

For each class in each band, take the ranges about the mean, which are spread about mean by a quantity equal to some k-times standard deviation, where k is either set by one of the rules explained latter in this section or specified explicitly as in fifth variant, given in sub-section E of section 4.3. For a pair of classes c_i and c_j, where $c_i \neq c_j$ and both c_i and c_j are from a set of m classes in band b, we define a parameter $k_{(c_i,c_j),b}$ as given in (4.6) and generate **K**-Matrix as explained in the section 4.3. Four variants are proposed by applying the rules given by the equations (4.12), (4.13), (4.14) and (4.15) respectively with two character codes replaced by XX, XN, NX and NN for AA, AI, IA, and II respectively and for the fifth variant we take k = 1. The four variants abbreviated as XX, XN, NX, and NN uses MaxMax, MaxMin, MinMax and MinMin rules as explained respectively in sub-sections A, B, C, and D of section 4.3. The fifth variant, abbreviated as SF, uses k = 1.

The algorithm illustrated above apparently seems to be similar to the method proposed by Wang [1990]. However, The four significant differences between the newly proposed algorithm and the Wang's approach are as follows:

1. In case of Wang [1990], one has to explicitly (or manually) assign the fuzzy membership. But in the proposed algorithm, the fuzzy membership is estimated as given in step 1 of sub-section *A* of section 4.4.
2. In case of Wang [1990] there is neither neither any guideline nor any procedure is given to calculate the initial fuzzy memberships required for the calculation of the fuzzy mean and the fuzzy covariance. In the proposed new algorithm, there are clear steps (steps 1 and 2 of sub-section *A* of section 4.4) to estimate the fuzzy membership.
3. In Wang's approach, the calculation of fuzzy membership values for fuzzy classification requires values of the fuzzy mean and fuzzy covariance in advance. These parameters have to be supplied by the analyst, which is not the case with the proposed new algorithm.
4. During the classification stage, Wang [1990] classifies all pixels by performing calculations similar to MLC and generates a matrix of fuzzy memberships and then uses hard classification. In the present work, first the classification is done using discrimination functions in single band. However, remaining (if any) unclassified pixels are classified using method similar to MLC.

These differences are found by carefully comparing the steps to be followed in [Wang 1990] and in the new proposed algorithm.

4.5 Multivariate Fuzzy Supervised Classifier and its Two Variants

In these algorithms, multivariate data are assumed to follow multivariate normal distribution. In the first variant, abbreviated as FM, the fuzzy membership is calculated from multivariate normal distribution and then it is normalized by dividing the fuzzy membership of the pixel in the class by sum of the fuzzy memberships of the pixel in all classes. In the second variant, abbreviated as FD, a distance weighted membership is taken. The three steps (first two steps calculate signatures and third step performs the classification) to be followed are:

1. Calculate initial membership of the training pixels using equation (4.16) in the first variant (FM) of the algorithm and using equation (4.23) in the second variant (FD).

$$f_c(\mathbf{x}) = 1 - \frac{D_c(\mathbf{x})}{\sum_{i=1}^{m} D_i(\mathbf{x})} \tag{4.23}$$

where,

$$D_i(\mathbf{x}) = \sqrt{\sum_{j=1}^{B} (\mathbf{x}_j - \mu_{i,j})^2} \tag{4.24}$$

m = Number of classes

$\mu_{i,j}$ = j^{th} component of mean-vector of class i

B = Number of bands

2. Calculate fuzzy mean and fuzzy covariance matrix for each class, using the equations (4.17) and (4.18) respectively.
3. To classify a pixel, first calculate fuzzy membership of each pixel in the given m classes, using equation (4.21). A pixel is then, hard classified to the class in which it has maximum fuzzy membership.

4.6 Univariate Beta-distribution based Fuzzy Supervised Classifier and its Two Variants

To incorporate various distributions, the beta-distribution with parameters p and q is fitted. The values of the p and q are calculated from the mean and variance of the data in a band. In the first variat, abbreviated as BF, the fuzzy membership of a pixel is calculated in all the bands separately in the given classes. The probability indicated by the beta-distributed is normalized to get the membership. Next, max-min rule is used to perform the classification. Second variant, abbreviated as MB, performs the modulation of the membership (as obtained in the first variant) by using two parameters α and β. The parameters α and β ranges in [0,1]. Again max-min rule will perform the classification.

4.6.1 Steps to be followed in case of first variant (BF) are as follows:

1. Calculate fuzzy membership of pixel $\mathbf{x}$ in band b and class c using equation (4.25). Please refer appendix F for the derivation of the equation (4.25).

$$f_{b,c}(\mathbf{x}) = \frac{(\mathbf{x}_b - L_{b,c})^{(p-1)} \times (H_{b,c} - \mathbf{x}_b)^{(q-1)}}{(H_{b,c} - L_{b,c})^{(p+q-1)} \times \left(\frac{\gamma(p).\gamma(q)}{\gamma(p+q)} \right)} \qquad (4.25)$$

where,

$p = (v_1 / v_2)(v_1 - v_1^2 - v_2)$

$q = p((1/v_1) - 1)$

$\gamma(r)$= Evaluates gamma function for some real number r

$v_1 = (\mu_{b,c}-L_{b,c})/(H_{b,c}-L_{b,c})$

$v_2 = \sigma_{b,c}/((H_{b,c}-L_{b,c})^2)$

$\mu_{b,c}$ = b^{th} component of the mean vector of class c

$\sigma_{b,c}$ = standard deviation in band b of the pixels of class c

$H_{b,c}$ = Highest value of b^{th} component of the pixels of the class c

$L_{b,c}$ = Lowest value of b^{th} component of the pixels of the class c

2. Apply MaxMin rule to perform classification of a pixel.

4.6.2 Steps to be followed in case of second variant (MB) are as follows:

1. Calculate fuzzy membership of pixel x in band b and class c using equation (4.25)
2. Now calculate maximum fuzzy membership and minimum fuzzy membership, and calculate modulated fuzzy membership, $f^*_{b,c}(\mathbf{x})$, using the equation (4.26)

$$f^*_{b,c}(\mathbf{x}) = \alpha(\min_b(f_{b,c}(\mathbf{x}))) + \beta(\max_b(f_{b,c}(\mathbf{x}))) \quad (4.26)$$

where,

α in [0,1]

$\min_b$ = min operator over b

$\max_b$ = max operator over b

$f_{b,c}(\mathbf{x})$ to be calculated as in equation (4.25).

3. Apply MaxMin rule to perform classification of a pixel.

The choice of the two parameters α and β, has its basis in Hurwicz criterion as discussed in section 3.7.3, for taking decision under uncertainty. However, in this variant, unlike Hurwicz criterion, two parameters are used and their sum is not necessarily 1. While executing the second variant, the value of is α incremented from 10 to 0 in steps of-0.01 and then for each value of α, the value of β is decremented from 10 to 0 in steps of 0.01. The fuzzy membership is, then normalized by dividing it by the sum of all the fuzzy membership values. So, there is no problem in taking the values of α and β more than 1. The results obtained from these are used to deduce that value of the modulating parameter α can be taken as 1 and that for β as 0.1, for high overall accuracy.

4.7 Univariate Normal-distribution based Fuzzy Supervised Classifiers

This method assumes normal distribution of class in each band and fuzzy membership is calculated as per equation (4.27). In first variant (abbreviated as MM) simple max-min is used to classify, and second variant (abbreviated as MC) calculates the number (count) of bands where fuzzy membership in that class shows max, and then the pixel is classified in the class for which this count is highest. In case of tie, class with smaller index is taken. This variant can be useful only for the cases where number of the classes is less than

or equal to the number of the bands used to represent the remote sensing data. The fuzzy membership function of class c in band b is

$$f_{b,c}(\mathbf{x}) = \frac{1}{\sigma_{b,c}\sqrt{2\pi}} \times \exp\left(- \frac{(\mathbf{x}_b - \mu_{b,c})^2}{2\sigma_{b,c}^2} \right) \tag{4.27}$$

Where $\mu_{b,c}$ and $\sigma_{b,c}$ are the mean and standard deviation of the class c in band b.

4.8 Univariate Range based Fuzzy Supervised Classifiers

First variant (abbreviated as RF) of this method essentially uses fuzzy membership derived using the lower and upper limits of values of pixels of a class in a band. Second method (abbreviated as MR), uses mean also to find fuzzy membership. The steps to be followed are as follows:

1. In case first variant (RF) calculate fuzzy membership, $f_{b,c}(\mathbf{x})$, using equation (4.28) and in case of second variant (MR) use equation (4.29) to calculate fuzzy membership.

$$f_{b,c}(\mathbf{x}) = \frac{\min((\mathbf{x}_b - L_{b,c}),(H_{b,c} - \mathbf{x}_b))}{\frac{(H_{b,c} - L_{b,c})}{2}} \tag{4.28}$$

where,

$H_{b,c}$ = Highest value of b^{th} component of the pixels of the class c

$L_{b,c}$ = Lowest value of b^{th} component of the pixels of the class c

$$f_{b,c}(\mathbf{x}) = \frac{\min((\mathbf{x}_b - L_{b,c}),(\mu_{b,c} - \mathbf{x}_b))}{\frac{(\mu_{b,c} - L_{b,c})}{2}} \text{ if } L_{b,c} \le \mathbf{x}_b \le \mu_{b,c} \tag{4.29a}$$

$$f_{b,c}(\mathbf{x}) = \frac{\min((\mathbf{x}_b - \mu_{b,c}),(H_{b,c} - \mathbf{x}_b))}{\frac{(H_{b,c} - \mu_{b,c})}{2}} \text{ if } \mu_{b,c} \le \mathbf{x}_b \le H_{b,c} \tag{4.29b}$$

where, $\mu_{b,c}$ = b^{th} component of the mean vector of class c

2. Apply MaxMin rule for classification.

4.9 Fuzzy Supervised Classifiers based on Spurious Pixel Elimination

This algorithm slightly deviates from second variant of the algorithm given in section 4.7. Here if the count comes less than 50% of the bands, then the pixel is treated as spurious and removed from the training set and the signatures are calculated again. This method does not provide satisfactory results because the number of

spurious pixels in most of the cases is only 0.02% – 0.05% of the number of pixels in the class. So, this approach is not included in the further analysis.

4.10 Fuzzy Supervised Classification by Iterative Algorithms

In this method, the first variant of the algorithm described in section 4.5, is iterated to get a stable estimates of the fuzzy means and the covariance matrix. The two major problems encountered in this approach are the high classification consumed time, and convergence may not be achieved. This algorithm is also not considered for further analysis.

In this chapter, a statistical classifier is proposed and explained in its five variants. Four of the five variants utilize different rules to select overlapping coefficient, k, and fifth variant uses $k = 1$. Thirteen proposed fuzzy supervised classifiers are also explained in this chapter, which we compare with MLC and also with proposed statistical classifier. Next chapter discusses an improvement to an explicit fuzzy supervised classification method [Melgani et al. 2000].

CHAPTER 5 *An Improved Explicit Fuzzy Supervised Classification Method*

In this chapter, the explicit fuzzy supervised classification method [Melgani et al. 2000] is improved in the modulation step and implemented. As a result of this improvement the overall accuracy has been improved significantly and the class assignment time is reduced to $1/10^{th}$ as compared to the conventional Maximum Likelihood Classifier (MLC) and $7/10^{th}$ with respect to the existing explicit fuzzy supervised classification method [Melgani et al. 2000].

The explicit fuzzy classification method [Melgani et al. 2000] and also the improved explicit classification method (abbreviated as IE) are carried out in three steps. First, an explicit fuzzyfication step is carried out to obtain an estimation of the class contribution in each band assuming normal distribution of the class samples. The second step applies a fuzzy reasoning rule on these fuzzy inputs to obtain, after a rescaling (modulation) operation, the fuzzy classification of the pixels. Finally, a hard classification can be deduced in the defuzzyfication step. In order to make the decision rule more simple and reliable, the explicit fuzzy method first analyses the data in order to extract a maximal amount of information through an explicit fuzzyfication process. Then, a MIN reasoning rule is carried out to pick out the minimal fuzzy membership degree for each class among the different bands. The inferred raw fuzzy outputs are rescaled (modulated) to provide a fuzzy classification of the pixel. The jump from a fuzzy classification to hard classification is easily done by a MAX operation representing the defuzzyfication process [Melgani et al. 2000].

Step 1: Explicit Fuzzyfication Process

The fuzzy domain consists of several fuzzy-sets representing the bands, and each fuzzy set (band) contains fuzzy subsets representing cover classes. Each fuzzy subset (cover class c) in a given fuzzy set (band b) is fully defined by its fuzzy membership function $f_{b,c}(\mathbf{x}_b)$, where $\mathbf{x}_b$ is the brightness value of the multi-spectral pixel $\mathbf{X}$ in the band b. The pixel vector $\mathbf{X}$ in the B-dimensional space is

$$\mathbf{X} = [x_1, x_2, \ldots, x_b, \ldots, x_B]^T \qquad (5.1)$$

The fuzzy membership function is assumed to be the Gaussian distribution because it represents a powerful general distribution model and involves a minimal extraction computational cost for the statistical characteristics of the signatures. Furthermore, the linear models are not able to express correctly the natural non-linear distribution of the classes. The class distributions are totally independent of each of the other and, consequently, no mixing assumption is imposed. In this method, the sum of the fuzzy memberships is not required to be one because 1). The MIN fuzzy reasoning rule is insensitive to this normalizing requirement, 2). This insensitiveness suggests that we can alleviate the burden of computations due to the normalizing

operation, and 3). The class distributions keep their natural aspect. If the prior knowledge of the class extents in the scene is available, the Gaussian distribution can be modulated. As in most of the cases the prior knowledge of the class extents is not available so the facultative prior knowledge of the class extents in the study area is reflected in the size of the class signatures necessary to carry out a statistics extraction in order to compute two important parameters to define completely the fuzzy membership function.

1. The mean (μ) of the class signature which represents the ideal pixel of the class, or in other words. only the point without any ambiguity about its fuzzy membership.
2. The standard deviation (σ) of the class signature, which will determine in a way the width of the fuzzy subset.

The fuzzy membership function of class c in band b is

$$f_{b,c}(x_b) = \exp\left(-\frac{(x_b - \mu_{b,c})^2}{2\sigma^{\bullet 2}_{b,c}}\right) \tag{5.2}$$

Where $\mu_{b,c}$ is the mean of the class c in band b, $\sigma^{\bullet}_{b,c}$ is the modulated standard deviation of class c in band b.

and

$$\sigma^{\bullet}_{b,c} = \alpha_{b,c}\sigma_{b,c} \tag{5.3}$$

where $\sigma_{b,c}$ is the standard deviation of class c in band b, and $\alpha_{b,c}$ is the modulation factor. If no prior knowledge of the class extents in the study area is available, the modulation factor must be neutralized to work with a pure Gaussian distribution. It means that $\forall\ b,c\ \alpha_{b,c} = 1$, otherwise, the modulation factor is calculated by the formula

$$\alpha_{b,c} = \log_c(P_{b,c} + \beta_0) \tag{5.4}$$

where, $P_{b,c}$ is the expected extent of class c in band b, and β_0 is the constant.

Melgani et al. [2000] does not use $P_{b,c}$ directly in the modulation factor in order to reduce the problem that the prior knowledge of the class extents in the scene is generally vague. Melgani et al. [2000] reflect it in the size of the class signatures and introduce the notion of the expected extent of the classes in each band by deducing the number of pixels expected to correspond to the mean of each class signature. For a given band b, $T_{b,c}$ represents the expected number of the pixels, in the signature histogram, corresponding to the mean $\mu_{b,c}$ of the class c in band b. The expected extent is calculated from the class histogram [Melgani et al. 2000] using the equation (5.5).

$$P_{b,c} = \frac{T_{b,c}}{\sum_{i=1}^{m} T_{b,i}} \tag{5.5}$$

where, m is the number of classes.

In explicit fuzzy method the calculation of class histogram requires more time due to sorting and then generation of frequency table. In the present research it is, assuming that the pixels of class c follow normal distribution and observing that to find expected extend at mean, the exponential term in the distribution becomes unity. This is modified as:

$$T_{b,c} = \frac{n}{\sigma_{b,c}\sqrt{2\pi}} \tag{5.6}$$

where ,

n = No of pixels in the class, and

$\sigma_{b,c}$ = Standard deviation of class c in band b

The advantage of this improvement is that we need not find class histogram. Hence, calculation of the signature is very fast. Moreover, when the mean is not an integer, some kind of interpolation is performed in the explicit fuzzy method [Melgani et al. 2000]. In the proposed improvement one does not need to interpolate the histograms. This improvement also results in better estimates of the fuzzy memberships for classification of pixels in the given classes. The modulation principle is useful to favour the large extent classes over the lower extent ones and, particularly, in the spectral regions where overlaps make the decision difficult. It has been deduced by empirical observation that a constant $\beta_0 = 1.25$ is the optimal one to satisfy the need to "not strengthen too much the strong classes and not weaken too much the weak classes."

The fuzzyfication process computes for a given pixel the fuzzy membership degrees for each class c and band b from the fuzzy membership functions fully defined using the Gaussian distribution and the results of the statistics extraction. We obtain a matrix of fuzzy inputs $\mathbf{F}_{ip}$ of order B x m where m is the number of classes and B the number of bands. For a multispectral pixel $\mathbf{X}$, the matrix of fuzzy inputs $\mathbf{F}_{ip}$ can be written

$$\mathbf{F}_{ip} = \begin{bmatrix} f_{1,1}(x_1) & f_{1,2}(x_1) & \cdots & f_{1,m}(x_1) \\ f_{2,1}(x_2) & f_{2,2}(x_2) & \cdots & f_{2,m}(x_2) \\ \cdots & \cdots & \cdots & \cdots \\ f_{B,1}(x_B) & f_{B,2}(x_B) & \cdots & f_{B,m}(x_B) \end{bmatrix} \tag{5.7}$$

This matrix is then analysed by the MIN reasoning rule, which is the second step of the method to produce the fuzzy classification.

Step 2: MIN Fuzzy Reasoning Rule

The absence of covariance information makes the fuzzy partition not necessarily optimal because classes with a high degree of covariance will not be limited to the high fuzzy membership areas. Therefore, high

fuzzy membership value more closely represents a maximum possible value rather than a high prior probability of class membership. Consequently, the lowest value of the fuzzy membership for all the bands is more likely to be "true" fuzzy membership. The fuzzy MIN reasoning rule, applied on the matrix of fuzzy inputs produced by step 1 above, will consider for each class, the membership degrees provided by the different fuzzy sets (bands) and pick out the minimal membership degree to represent the class extent in the pixel. Applying the MIN operation, we obtain a primitive fuzzy output vector

$$\mathbf{F}_{op}^{'} = [F_1^{'}(\mathbf{X}), F_2^{'}(\mathbf{X}), \ldots, F_m(\mathbf{X})]^T \tag{5.8}$$

where,

$$\mathbf{F}_i^{'}(\mathbf{X}) = Min(f_{b,i}(x_b)) \quad \text{with } b = 1, 2, \ldots, B. \tag{5.9}$$

The fast, simple and reliable step of the MIN fuzzy reasoning rule is immediately followed by a rescaling operation in order to normalize the class extents deducted from different fuzzy sets and sharing the same pixel. The final vector of fuzzy outputs is

$$\mathbf{F}_{op} = [F_1(\mathbf{X}), F_2(\mathbf{X}), \ldots, F_m(\mathbf{X})]^T \tag{5.10}$$

where,

$$F_i(\mathbf{X}) = \frac{F_i^{'}(\mathbf{X})}{\sum_{j=1}^{m} F_j^{'}(\mathbf{X})} \tag{5.11}$$

This vector represents the fuzzy classification showing the inferred class extents of the pixel and from which a hard classification can be deducted by a defuzzyfication step. We note that the extraction from the data of first order statistical characteristics for deducting the fuzzy membership functions and the application of the MIN fuzzy reasoning rule involve a partition of the spectral space into fuzzy parallelepiped regions which relate conceptually the method to a fuzzy parallelepiped classifier. Furthermore, the explicit fuzzification associated to the MIN fuzzy reasoning rule reveals the modular aspects of the method, which allows adding and removing any band from the classification scheme of the study area because of the relative independence between the bands. The adding or removal of a vector class will, however, require a minor computational cost due to the adjustment of the modulation factors if a prior knowledge of the class extents is available.

Step 3: Defuzzyfication

Next to the fuzzy classification, performing a MAX operation to defuzzify the fuzzy output into a hard output provides a hard classification. We select, among the classes mixed in the pixel, the class c with the highest extent such that

$$\forall\, i \in 1, 2, \ldots, m \text{ and } i \neq c \quad F_c(\mathbf{X}) \geq F_i(\mathbf{X}) \tag{5.12}$$

In this chapter, an existing explicit fuzzy supervised classifier [Melgani et al. 2000] has been improved and it can be seen analytically that the proposed improvement will work due to better estimates of the number of samples at the mean. As it avoids histogram generation and interpolation, it is expected to be faster both at signature calculation and at the class assignments. Next chapter explains the data reporting for the investigation of various algorithms and explains the implementation of the computer programs for various tasks. The chapter discusses four different cases of separable and overlapping classes. The method of performing investigations on the 20 algorithms is also explained in the next chapter. The number of classifiers being large, some sophisticated programs are written to present the results and plot them. These programs are presented in Appendix E.

CHAPTER 6 *Data Reporting and Implementation*

6.1 Data Reporting

To compare the performance of all the classification algorithms explained in section 3.4.3 and in chapters 4 and 5, the study has been performed on four band LISS-III data covering the Pilani town, in the state of Rajasthan, India. The LISS-III sensor, mounted on IRS-1C, has four bands: Band 1 (0.52-0.59 μm, green band), Band 2 (0.62-0.68 μm, red band), Band 3 (0.77-0.86 μm, Near infra-red band) and Band 4 (1.55-1.70 μm, Short wave Infra-red band). The spatial resolution of Bands 1, 2, and 3 is 23.5 m and that of Band 4 is 70.5 m. Due to different spatial resolution, the data in Band 4 are re-sampled at the resolution of 23.5 m from 70.5 m, using IDIRISI GIS [Heywood et al. 2002] software. All the classifiers implemented in this work are supervised classifiers and hence, require training sites to be provided by the user. In this research work, four cases (Table 6.1) are considered and training sites are specified for each of the cases. The regions of the training sites in LISS-III color composite images are identified and then recorded in an external data file for subsequent classification to perform a comparative study.

TABLE 6.1

THE FOUR CASES ON WHICH ALL ALGORITHMS ARE APPLIED

Case	Details of sites and statistics		Number of		Number of Pixels		
	Sites	Statistics	Sites	Classes	Training	Test	Total
1.	Fig. 6.1	Table 6.3	9	4	3114	1555	4669
2.	Fig. 6.2	Table 6.4	7	4	2130	1063	3193
3.	Fig. 6.3	Table 6.5	6	3	1728	863	2591
4.	Fig. 6.4	Table 6.6	3	2	970	485	1455

The proposed classification methods along with MLC are mentioned in Table 6.2 along with their abbreviation and a brief description. As all proposed classifiers are assigned two character abbreviations, MLC is abbreviated as ML for consistency reasons. These abbreviations are used in the Tables 6.3 - 6.6; Tables C1 – C4 given in appendix C, appendix D; Figs. 6.5, and 7.1 – 7.5 and in chapter 7. Author applied all the algorithms listed in Table 6.2 on the four cases mentioned above (Table 6.1). The false color composite, shown in Figs. 6.1 to 6.4, is obtained by assigning band 3 to red, band 2 to green, and band 1 to blue color channel. The results, obtained after executing the algorithms mentioned in Table 6.2, are plotted in a format as given in Fig. 6.5. Fig. 6.5 illustrates various components of the plots given in Fig. 7.1 to 7.4. These

components are illustrated in red colour in Sans Comic font and with arrows indicating the components in Fig. 6.5. For the four cases, the sites are marked in the Figs. 6.1, 6.2, 6.3, and 6.4 respectively for case 1, 2, 3, and case 4, as small areas enclosed by rectangle or square. The number at top left corner of a typical rectangular or square area in Figs. 6.1–6.4 is the serial number of the site and these sites along with their class-types and the number of pixels in each of these sites is tabulated in the Tables 6.3 (a), 6.4(a), 6.5 (a), and 6.6 (a) respectively for case 1, 2, 3, and 4. The classes of the sites and the total number of pixels in them and their split in training and test pixels is given in the Tables 6.3 (b), 6.4 (b), 6.5 (b) and6.6(b) respectively for cases 1, 2, 3, and 4.

TABLE 6.2
LIST OF ABBREVIATIONS OF PROPOSED CLASSIFIERS AND MAXIMUM LIKELIHOOD CLASSIFIER

Classifier	Description	Ref. Sec.
ML	Maximum Likelihood Classifier	3.4.3
FM	Fuzzy Supervised classifier (fuzzy membership is normally distributed in multivariate)	4.5
FD	Fuzzy Supervised classifier (fuzzy membership is dependent on distance from the centroid of multivariate data)	4.5
SM	Proposed statistical supervised method, coefficient of overlapping, k, set to 1	4.3E
SF	Proposed fuzzy supervised method, coefficient of overlapping, k, set to 1	4.4E
IE	Improved explicit fuzzy method	Ch. 5
BF	Univariate Beta distribution based fuzzy supervised classifier	4.6.1
MB	Univariate Beta-distribution based MaxMin modulated fuzzy method	4.6.2
MC	Classifier based on the Maximum count of bands in which fuzzy memberships is maximum in given bands	4.7
MM	Simple MaxMin rule applied on normally distributed fuzzy membership in bands	4.7
AA	Proposed statistical supervised method, where MaxMax rule sets coefficient of overlapping, k	4.3A
AI	Proposed statistical supervised method, where MaxMin rule sets coefficient of overlapping, k	4.3B
IA	Proposed statistical supervised method, where MinMax rule sets coefficient of overlapping, k	4.3C
II	Proposed statistical supervised method, where MinMin rule sets coefficient of overlapping, k	4.3D
XX	Proposed fuzzy supervised method, where MaxMax rule sets coefficient of overlapping, k	4.4.1
XN	Proposed fuzzy supervised method, where MaxMin rule sets coefficient of overlapping, k	4.4.1
NX	Proposed fuzzy supervised method, where MinMax rule sets coefficient of overlapping, k	4.4.1
NN	Proposed fuzzy supervised method, where MinMin rule sets coefficient of overlapping, k	4.4.1
RF	Fuzzy supervised method where fuzzy membership is expressed in terms of range of values in the bands	4.8
MR	Fuzzy supervised method where fuzzy membership is expressed in terms of range about the mean in the bands	4.8

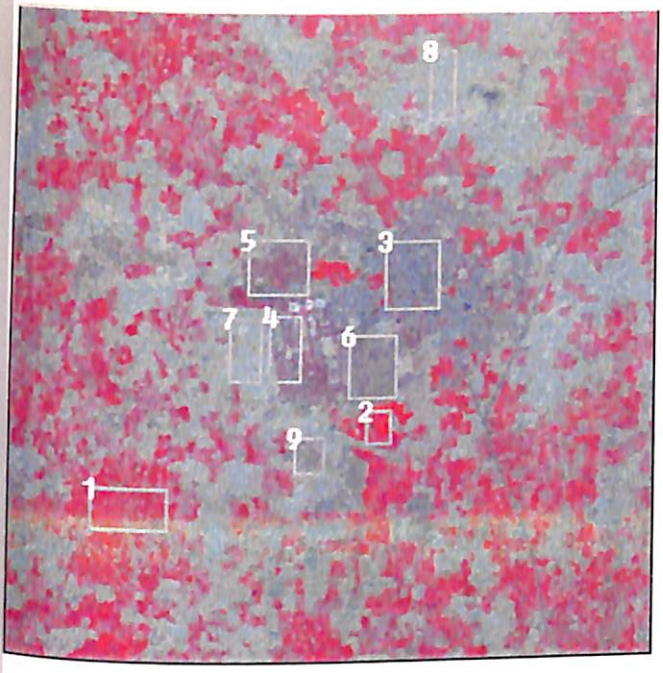


Fig. 6.1 Sites of Case 1 Indicated on Color Composite of Raw LISS-III Data of Town Pilani, Rajasthan, India

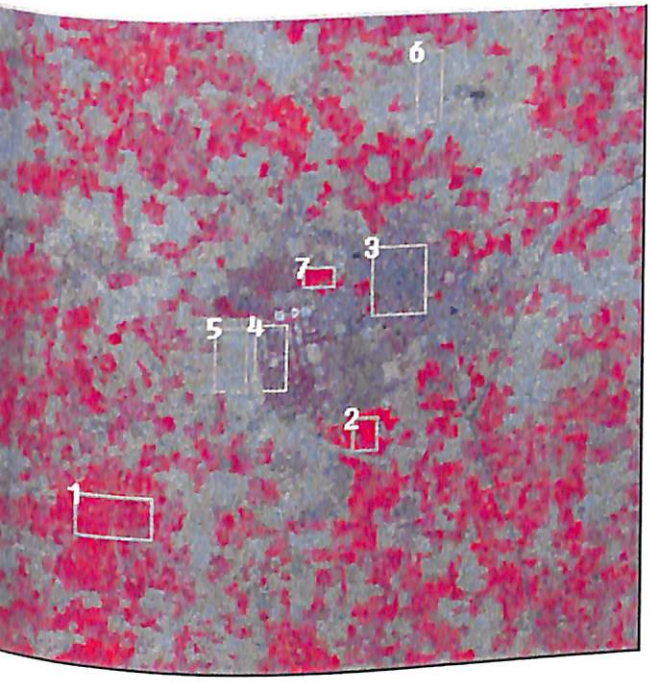


Fig. 6.2 Sites of Case 2 Indicated on Color Composite of Raw LISS-III Data of Town Pilani, Rajasthan, India

Fig. 6.3 Sites of Case 3 Indicated on Color Composite of Raw LISS-III Data of Town Pilani, Rajasthan, India

Fig. 6.4 Sites of Case 4 Indicated on Color Composite of Raw LISS-III Data of Town Pilani, Rajasthan, India

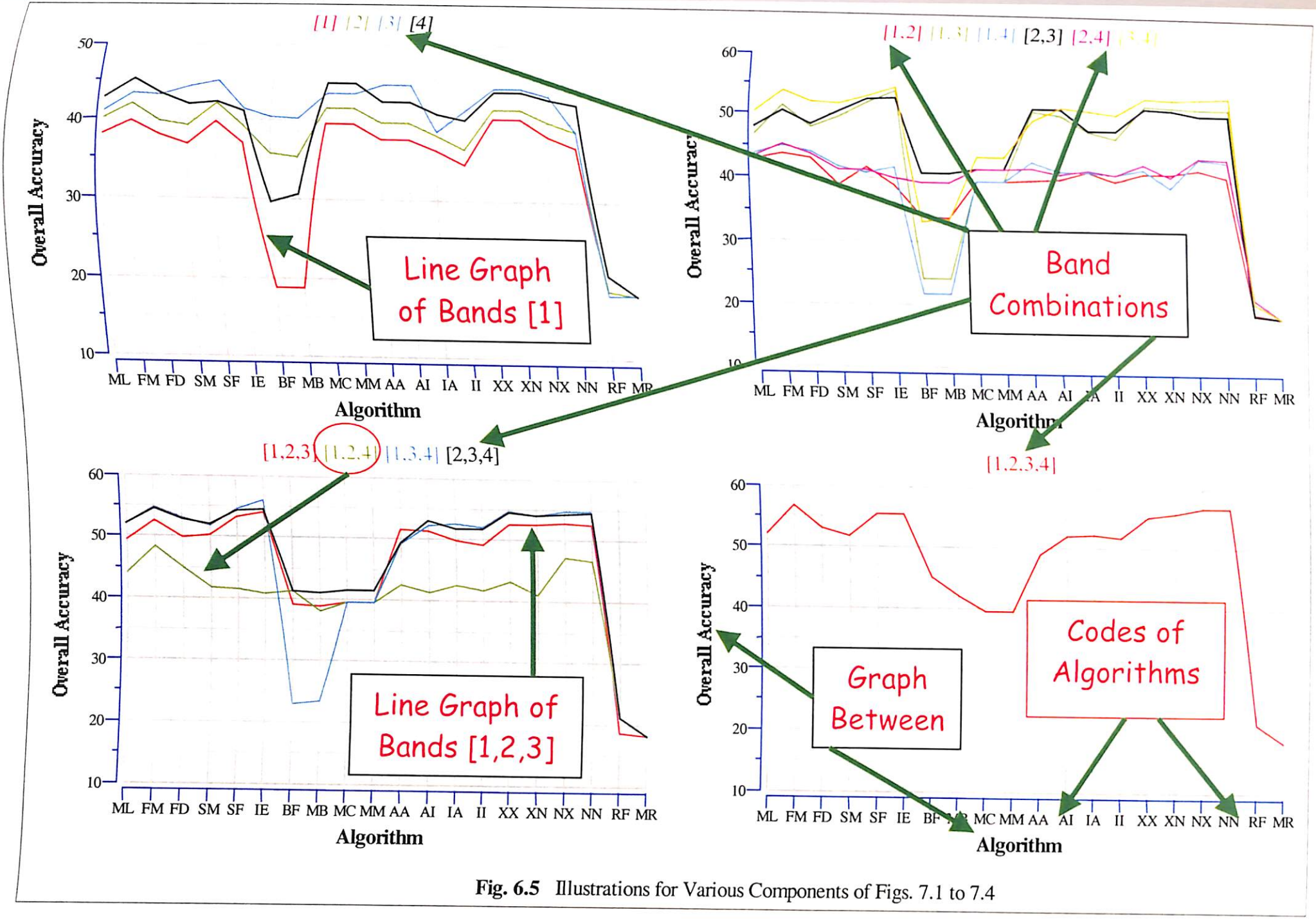


Fig. 6.5 Illustrations for Various Components of Figs. 7.1 to 7.4

Divergence matrix generated for the cases 1, 2, 3, and 4 are tabulated in the Tables 6.3 (c), 6.4 (c), 6.5 (c), and 6.6 (c) respectively. Tables 6.3 (d), 6.4 (d), 6.5 (d), and 6.6 (d) give the values of transformed divergence (on the scale of 100) between all pairs of the classes for the four cases respectively. Similarly, Tables 6.3 (e), 6.4 (e), 6.5 (e), and 6.6 (e) give the values of JM distances between all pairs of the classes for the four cases respectively. Divergence, transformed divergence and JM-distances are calculated by representing the training data in all bands. Tables of the results obtained by executing 20 classification algorithms on the test pixels of the four cases represented in the combinations of 1, 2, 3 and 4 bands are presented in appendix C and confusion matrices for case 1, when data are classified using all bands are given in appendix D.

TABLE 6.3

SAMPLE SETS AND DISTANCE MATRICES OF TRAINING PIXELS OF CASE 1 **(a)** SITES **(b)** CLASSES **(c)** DIVERGENCE MATRIX **(d)** TRANSFORMED DIVERGENCE MATRIX **(e)** JM DISTANCE MATRIX

Site	Class	# Pixels
1	Crop	740
2	Crop	192
3	Urban	800
4	Urban	465
5	Urban	728
6	Trees	660
7	Sand	450
8	Sand	396
9	Trees	238
Total # Pixels		4669

(a) SITES

Sample Set	Number of Pixels in Class			
	Crop	Urban	Trees	Sand
Total	932	1993	898	846
Training	622	1329	599	564
Test	310	664	299	282

(b) CLASSES

Class	Crop	Urban	Trees	Sand
Crop	0	3	12	7
Urban		0	3	3
Trees			0	5
Sand				0

Average = 5.5

(c) DIVERGENCE MATRIX

Class	Crop	Urban	Trees	Sand
Crop	0	35	77	56
Urban		0	32	31
Trees			0	47
Sand				0

Average = 46.33

(d) TRANSFORMED DIVERGENCE MATRIX

Class	Crop	Urban	Trees	Sand
Crop	0	0.39	1.15	0.74
Urban		0	0.28	0.28
Trees			0	0.45
Sand				0

Average = 0.548

(e) JM DISTANCE MATRIX

TABLE 6.4

SAMPLE SETS AND DISTANCE MATRICES OF TRAINING PIXELS OF CASE 2 (a) SITES (b) CLASSES (c) DIVERGENCE MATRIX (d) TRANSFORMED DIVERGENCE MATRIX (e) JM DISTANCE MATRIX

Site	Class	# Pixels
1	Gram	740
2	Wheat	192
3	Urban	800
4	Urban	465
5	Sand	450
6	Sand	396
7	Wheat	150
Total # Pixels		3193

(a) SITES

Sample Set	Number of Pixels in Class			
	Gram	Wheat	Urban	Sand
Total	740	342	1265	846
Training	494	228	844	564
Test	246	114	421	282

(b) CLASSES

Class	Gram	Wheat	Urban	Sand
Gram	0	3	7	12
Wheat		0	2	5
Urban			0	2
Sand				0

Average = 5.17

(c) DIVERGENCE MATRIX

Class	Gram	Wheat	Urban	Sand
Gram	0	32	56	78
Wheat		0	23	45
Urban			0	26
Sand				0

Average = 43.33

(d) TRANSFORMED DIVERGENCE MATRIX

Class	Gram	Wheat	Urban	Sand
Gram	0	0.36	0.64	1.30
Wheat		0	0.22	0.5
Urban			0	0.26
Sand				0

Average = 0.547

(e) JM DISTANCE MATRIX

TABLE 6.5

SAMPLE SETS AND DISTANCE MATRICES OF TRAINING PIXELS OF CASE 3 (a) SITES (b) CLASSES (c) DIVERGENCE MATRIX (d) TRANSFORMED DIVERGENCE MATRIX (e) JM DISTANCE MATRIX

Site	Class	# Pixels
1	Crop	792
2	Tree	459
3	Crop	192
4	Urban	216
5	Urban	722
6	Tree	210
Total # Pixels		2591

(a) SITES

Sample	Number of Pixels in Class		
	Crop	Tree	Urban
Total	984	669	938
Training	656	446	626
Test	328	223	312

(b) CLASSES

Class	Crop	Tree	Urban
Crop	0	16	10
Tree		0	2
Urban			0

Average = 9.33

(c) DIVERGENCE MATRIX

Class	Crop	Tree	Urban
Crop	0	87	70
Tree		0	21
Urban			0

Average = 59.33

(d) TRANSFORMED DIVERGENCE MATRIX

Class	Crop	Tree	Urban
Crop	0	1.55	1.00
Tree		0	0.21
Urban			0

Average = 0.92

(e) JM DISTANCE MATRIX

TABLE 6.6

SAMPLE SETS AND DISTANCE MATRICES OF TRAINING PIXELS OF CASE 4 (a) SITES (b) CLASSES (c) DIVERGENCE MATRIX (d) TRANSFORMED DIVERGENCE MATRIX (e) JM DISTANCE MATRIX

Site	Class	# Pixels
1	Crop	792
2	Tree	459
3	Crop	204
Total # Pixels		1455

(a) SITES

Sample Set	Number of Pixels in Class	
	Crop	Tree
Total	996	459
Training	664	306
Test	332	153

(b) CLASSES

Class	Crop	Tree
Crop	0	22
Tree		0

Average = 22

(c) DIVERGENCE MATRIX

Class	Crop	Tree
Crop	0	94
Tree		0

Average = 94

(d) TRANSFORMED DIVERGENCE MATRIX

Class	Crop	Tree
Crop	0	1.57
Tree		0

Average = 1.57

(e)

As the data is represented in four bands (or channels), we get 4 single bands, 6 combinations of two bands, 4 combinations of three bands, and 1 combination of the four (all) bands. In this research work, processing has been performed on all 15 combinations of the bands.

For each case, the pixels of the sample sites representing same class are listed. From this list, 2/3 of total pixels are taken as training pixels and 1/3 of total pixels as test pixels. For this purpose, first two pixels are taken as training pixels, and third pixel as test pixel, then fourth and fifth pixels are taken as training pixels, and the sixth pixel is taken as test pixel and so on. Finally the remaining (if any) one or two pixels are placed in the set of training pixels. The number of pixels in each class in cases 1, 2, 3, and 4 are tabulated in Tables 6.3 (b), 6.4 (b), 6.5 (b), and 6.6 (b) respectively.

In all 1200 executions are performed for all 20 algorithms, all the combinations of the 1,2,3 and 4 bands and for each of the four cases. The following four tasks are performed for each execution:

1. Depending on the method, class-signatures are calculated from the training pixels and then by using the signatures and the classification algorithm, the test pixels are classified. After the classification of test pixels, confusion matrices and Kappa Coefficients are calculated.
2. To have idea about the statistical separation of the sample pixels, divergence, transformed divergence, and JM distances between all possible class-pairs in training sets are calculated for each case.
3. To evaluate performance of an algorithm in terms of speed, three time-durations are recorded 1) time to find the signatures or statistics (Statistics Computing Time) abbreviated as SCT, 2) time to perform class assignment during classification, the Class Assignment Time, abbreviated as CAT, and finally 3) the total classification time (sum of SCT and CAT) abbreviated as CT.
4. The results obtained by 20 classifiers for each case are plotted. These results are given in Tables C1, C2, C3, and C4 in appendix C for the four cases respectively.

Finally, the classifier demonstrating either maximum OA or minimum CAT in any of the band combinations of 1, 2, 3, and 4 bands, for each of the four cases is selected inclusive of all tie situations and the classifier along with maximum OA and minimum CAT is recorded. Scatter plot between maximum OA (x-axis) and minimum CAT (y-axis) is given in Figs. 7.5(a), 7.5(b), 7.5(c), and 7.5(d), for the cases 1, 2, 3, and 4 respectively, for which detailed interpretation and analysis is given in next chapter.

6.2 Implementation

All implementations are done using *LPA WinProlog* and its associated tools [Steel 2000, Shalfield et al. 2003]. Prolog [Clocksin and Mellish 1989, Sterling and Shapiro 1996, Bratko 2001] is used in this work because the latest trend in the remote sensing data interpretation and GIS is to use Artificial Intelligence techniques not only at the classification stage but also at other stages including the integration of output of classification in GIS [McKeown 1987, Kartikeyan et al. 1995, Jia 2000]. Prolog and Lisp [Patterson 1990] are the two most commonly used languages in the Artificial Intelligence community. Prolog in general

have many added advantages, over conventional procedural programming languages, like meta-programming, rapid prototyping capabilities, code-reusability, smaller code-size, natural levels of abstraction and object orientation, backtracking and unification capability, etc. Apart from this, we have tools like Win32 programming [Steel 2000], Portable Dialog Manager (PDM), and Fuzzy Logic Toolkit (FLINT) [Shalfield et al. 2003] associated with *WinProlog*.

Using Prolog programming programs are developed to execute a number of functions. The program listing of these functionalities is given in Appendix E. The following functionality has been implemented using *LPA WinProlog and its associated tools*, on 300 GL IBM 500 MHz PC with 64 MB RAM running *WinProlog* under Windows 98.

- A program which can extract a rectangular or square area of interest from the four-band LISS-III image data and the converts smaller image in terms of Prolog facts and stores these facts in an external file.
- A program to display the remote sensing data in form of user-specified color composite by taking any three or less (but at least one) bands. The image generated by this program is stored as bitmap in an external file.
- A GUI based Win32 program to display two different color composites of images stored in external files as bitmaps is developed for selecting and marking sites in the color composites. A set of other programs permits addition. deletion and updating of training sample sites. It is also possible to view number of pixels in a list-box. Another program permits storage of the sample-sites as prolog facts in an external file. This set of programs support loading (for subsequent updates) the details of sample-sites already stored in an external file.
- Another GUI based application. developed using Portable Dialog Manager and Win32 Programming tools of *WinProlog*, provides state-of-the-art simulation and presentation tool which permit executing any algorithm on any set of training sites and data reduced (if required) in less number of bands. This application supports utilities to calculate and display statistics of selected sample pixels, statistical separation measures. confusion matrix, kappa coefficient, statistics computing time, class assignment time, total classification time, histogram with mean and standard deviation details. This GUI also supports merger of sites of same classes and then spilt the pixels of classes in user specified ratio of training and test pixels.
- A program, which runs all the algorithms on data of all 15 possible combinations of bands. The results of this program can be tabulated in two external files. First file records data regarding classification accuracies, and classification times in the form of Prolog facts, and the second file contains details of confusion matrices in form of steam of text.
- A sophisticated GUI application prints the plots of Overall Accuracies (OA), Kappa Coefficients (KHAT), Statistics Computing Time (SCT), Class Assignment Time (CAT), and Total

Classification Time (CT) against algorithms. This can be done in any combination of bands and for any sample-sites. Four same types of plots (in 4 types of band combinations) can also be generated on one screen. Four different types of graphs can be generated on one screen if required. Only one selected graph can also be generated and printed on paper.

- Programs are also developed to handle routinely used matrix-related calculations, and statistical calculations.
- Needless to say that programs are also developed for each of the 20 classification algorithms. Fourteen of these algorithms, use fuzzy logic and fuzzy set theory and hence programs are developed to find fuzzy membership to apply various operations on these fuzzy memberships.
- In this work Caley-Hamilton Theorem [Pipes 1958] has been implemented for matrix-inversion and finding determinant of a matrix.
- Programs to calculate gamma function, beta-function, univariate normal distribution and multivariate normal distribution are also developed.
- Programs to calculate divergence, transformed divergence and JM-distance are also implemented.

In the next chapter, the results obtained after executing the 20 classifiers on the data of each of the four cases represented in all combinations of 1, 2, 3, and 4 bands, are analyzed to evaluate performance of each of the classifier with respect to ten criteria.

CHAPTER 7

Interpretation and Analysis of Results

7.1 Introduction

In the last four decades of successful application of the remote sensing technology and GIS to monitor earth resources and to solve various problems, it was always felt to design new methodologies for image classification. The conventional method to perform this task is Maximum likelihood classifier (MLC), which assumes that training samples follow multivariate unimodal normal distribution. Generally, performance of a new method is compared with respect to MLC. This chapter details out a comparative performance of the classification algorithms proposed or improved in the present research.

7.2 Interpretation of the Results

Three types of training sites can be distinguished in the terms of overlapping among the pair of classes. Those, which have high overlap and mixing among the training samples, depict less divergence (less transformed divergence and less JM-distance). Second, statistically separable training samples depict large values of divergence between all possible pairs of classes. In the third type there are situations when some class-pairs are separable and other class-pairs are not separable. These types of samples are less separable. The effects of divergence on statistical separability are not well established. So, clear guidelines cannot be established to infer something about separability, by merely using divergence as the measure of the separability. What can be said is if more the divergence more will be the separability [Mather 1987]. However, average divergence can be calculated for the set of all possible pairs of training classes and this can be compared with the average of another set of classes. The set of classes showing higher average divergence can be considered relatively more separable than the other set [Swain and Davis 1978]. Transformed divergence, however, indicates clear separability between two classes if its value for all the possible pairs of classes is about 80 or more on the scale of 100. It is also possible to decide about relative separability by using average transformed divergence [Swain and Davis 1978, Mather 1987]. Another important measure of statistical separation is Jaffrie-Matusita (JM) distance. Its value between 1.4 and 2.0 shows high separability and for the JM-distance less than 1.4 the separability further decided by using covariance matrices of the classes. The covariance matrix is one of the most important mathematical concepts in the analysis of multi-spectral remote sensing data. In covariance matrix it can be seen that if there is correlation between the responses in a pair of spectral bands, the corresponding off-diagonal element in the covariance matrix will be large as compared to the major diagonal terms. If there is a little correlation, the off-diagonal terms will be

close to zero. In the situation of less correlation, if JM distance is less than 1.4 then the separation can be less [Swain and Davis 1978].

In the first stage of interpretation relative separation among the four cases is estimated by considering either the values of divergence, transformed divergence and JM-distance or by considering the average value of these measures. However, the statistical separability also depends on the set of bands selected for the representation of the data. Most accurate estimates of the separability measures can be obtained by using all bands for the calculation of these measures. So, the separability measures are calculated by representing training sites in all the four bands. In the second stage, the efficiency of classification algorithms is compared for three types of training-sets, represented in four cases as detailed out in chapter 6. The cases 1 and 2 have more overlapping and mixing of the classes than the other two cases. The case 3 represents less separation because only some class pairs are separable and others are not separable. The case 4 has clearly separable classes. The performance of the classifiers, when executed on test data (considered in combinations of 1, 2, 3, and 4 bands) of four cases, is interpreted using the following criteria:

1. Overall accuracy (OA %) and KHAT
2. Statistics and Signature Calculation Time (SCT)
3. Class Assignment Time (CAT)
4. Classification Time (CT)
5. Maximum Overall Accuracy and Minimum Class Assignment Time
6. Complexity of Coding the Algorithm
7. Integration of Algorithm with GIS
8. Feasibility of Algorithm for Online Classification
9. Scalability of Algorithm in Terms of Number of Bands
10. Effect of Fuzzy Membership Function and its Modulation

Interpretation and analysis are performed by identifying some trends in the results obtained from execution of 20 algorithms on the test samples of the four cases. **The coloured plots of these results are given in the Figs. 7.1, 7.2, 7.3, and 7.4 for the Case 1, Case 2, Case 3, and Case 4 respectively and coloured scatter plot of maximum overall accuracy and minimum class assignment time is given in Fig. 7.5.** Statistics for the sample sets of the four cases are given in appendix B. The results are also tabulated in appendix C. Confusion matrices and K-Matrices for case 1, when all bands are used for presentation of the sample sets, are given in appendix D.

7.2.1 Interpretations of the statistics and plots of Case 1

From Table 6.3 it is observed that the four classes are overlapping. Fig. 7.1(a) reveals the following:

- In single bands, a number of methods (FM, SF, MC, MM, AA, AI, XX, XN) achieve the maximum OA of around 45% in bands [3] and [4].
- In bands combination [3,4], the method IE improves the maximum OA to 55%; the methods FM and NN both achieve OA of 52%; and MLC achieves only 50% maximum OA.
- The nature of the results obtained in combinations of three bands is more or less similar to the results obtained in combinations of two bands. However, an important feature is that the maximum OA increases for all the algorithms and IE achieves the maximum OA of about 56% in the bands combination [1,3,4].
- The methods FM, NX and NN achieve the maximum OA of 57% in the combination of four bands.

This indicates that the most separable bands are [3] and [4] for this case and in single bands a number of methods give comparable OA. In combinations of multi-bands the methods IE, FM, NX, and NN achieve the maximum OA, which is 2% - 6% higher than the OA achieved by MLC. Fig. 7.1 (b) and sub-section B1 of appendix B also support the observations of Fig. 7.1 (a) as follows:

- Single bands [3] and [4], bands combination [3.4], and band-combinations [1,3.4] and [2,3,4] are the most separable combinations of the single, two and three bands respectively. The maximum value of KHAT is 45% in single bands [3] and [4]. 54% in bands combination [3.4], 57% in bands combination [1,3,4] and 57% in combination of four bands.
- The covariance and corelation matrices (sub-section B1 of appendix B) of the classes also consistently confirm the separation of data in the corresponding bands as observed above.

The following are the observations with regard to Figs. 7.1 (c)-(e):

- In single bands, the methods FM, FD, SF, XX, XN, NX, and NN take more SCT as compared to other methods. CAT required by these methods varies. Hence, CT is dependent mainly on CAT.
- The nature of variations in CAT remains similar in all the combinations of multi-bands. Only the duration of time changes. As expected, SCT, CAT and CT increase with increase in the number of bands. The methods IE and RF require minimum CT. As we increase the number of bands, CT for all other methods increases significantly except for the method IE for which it increases slightly.
- From Figs. 7.1(a) and 7.1(d) it can be observed the bands combination [1,3,4] gives highest OA with the methods IE (55%) followed by NX (52%), FM (52%), and ML (50%). However, the time taken by the methods FM and NX to achieve 2% higher OA as compared to MLC is twice that of time taken by MLC. The method IE increases OA by 5% over MLC and it is 9 times faster than MLC. In combination of four bands the method IE is about 9 times faster than MLC.

Keeping accuracy and time as selection-criteria, it is inferred that the method IE is the most suitable method for a classification problem similar to case 1.

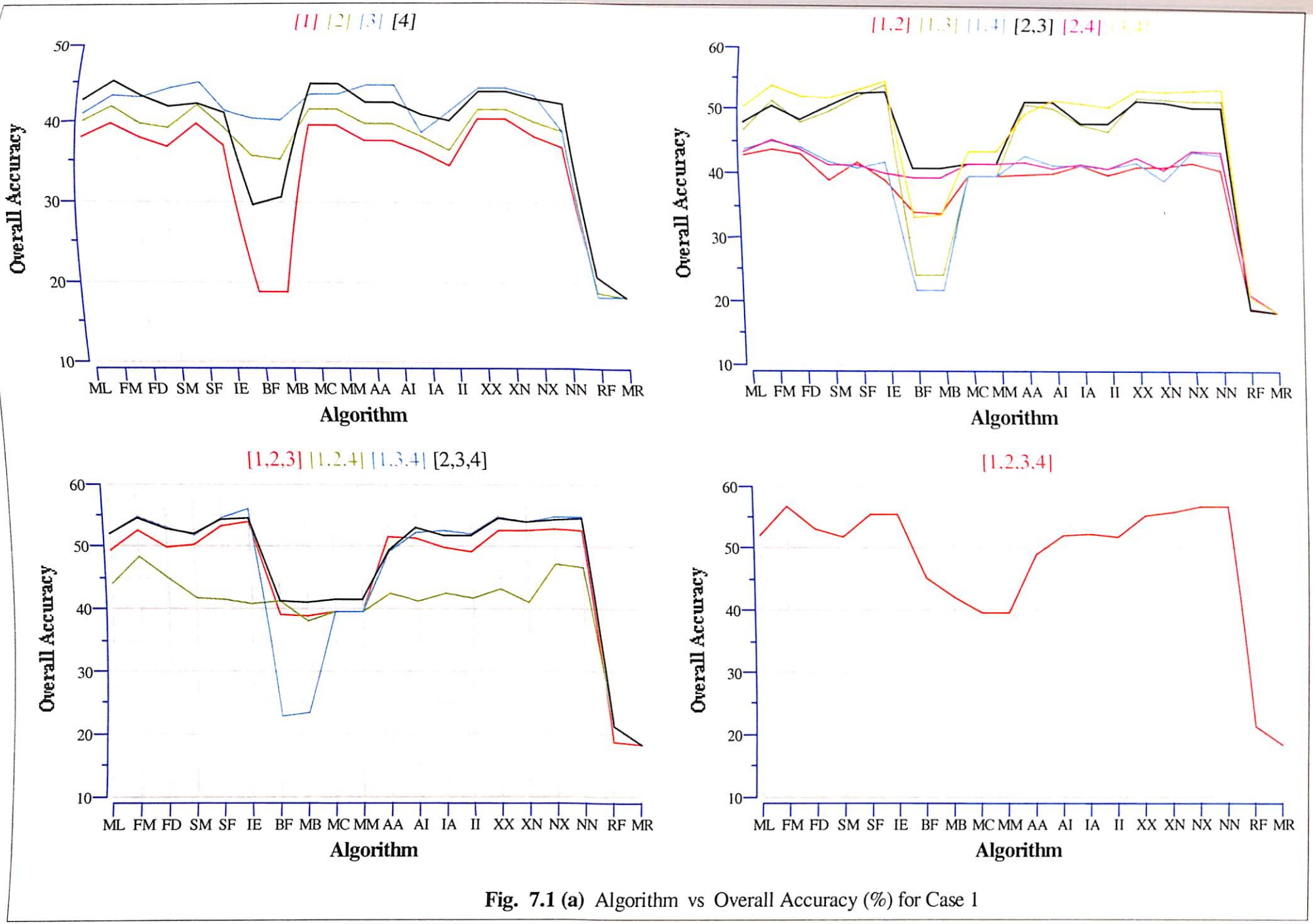


Fig. 7.1 (a) Algorithm vs Overall Accuracy (%) for Case 1

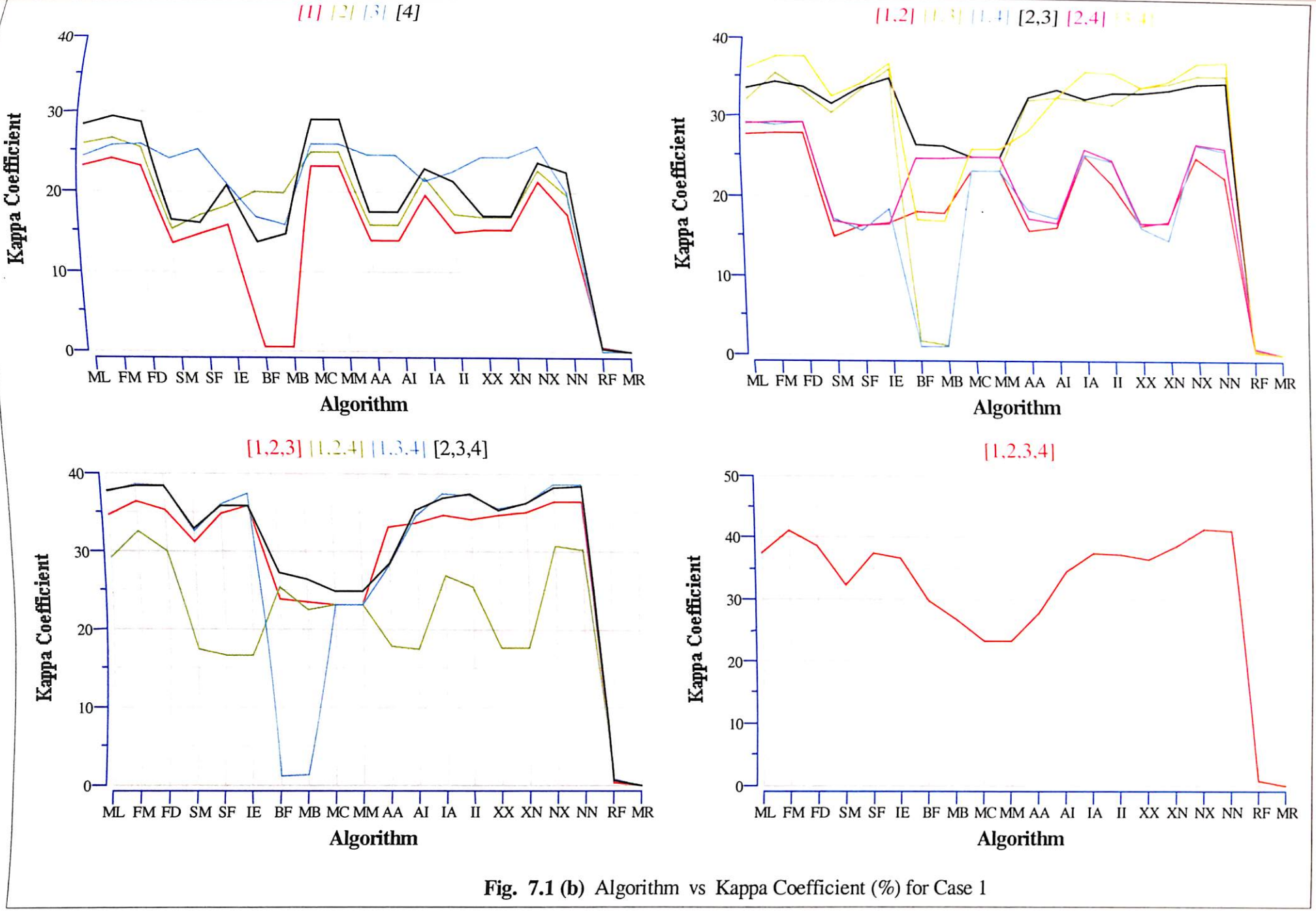


Fig. 7.1 (b) Algorithm vs Kappa Coefficient (%) for Case 1

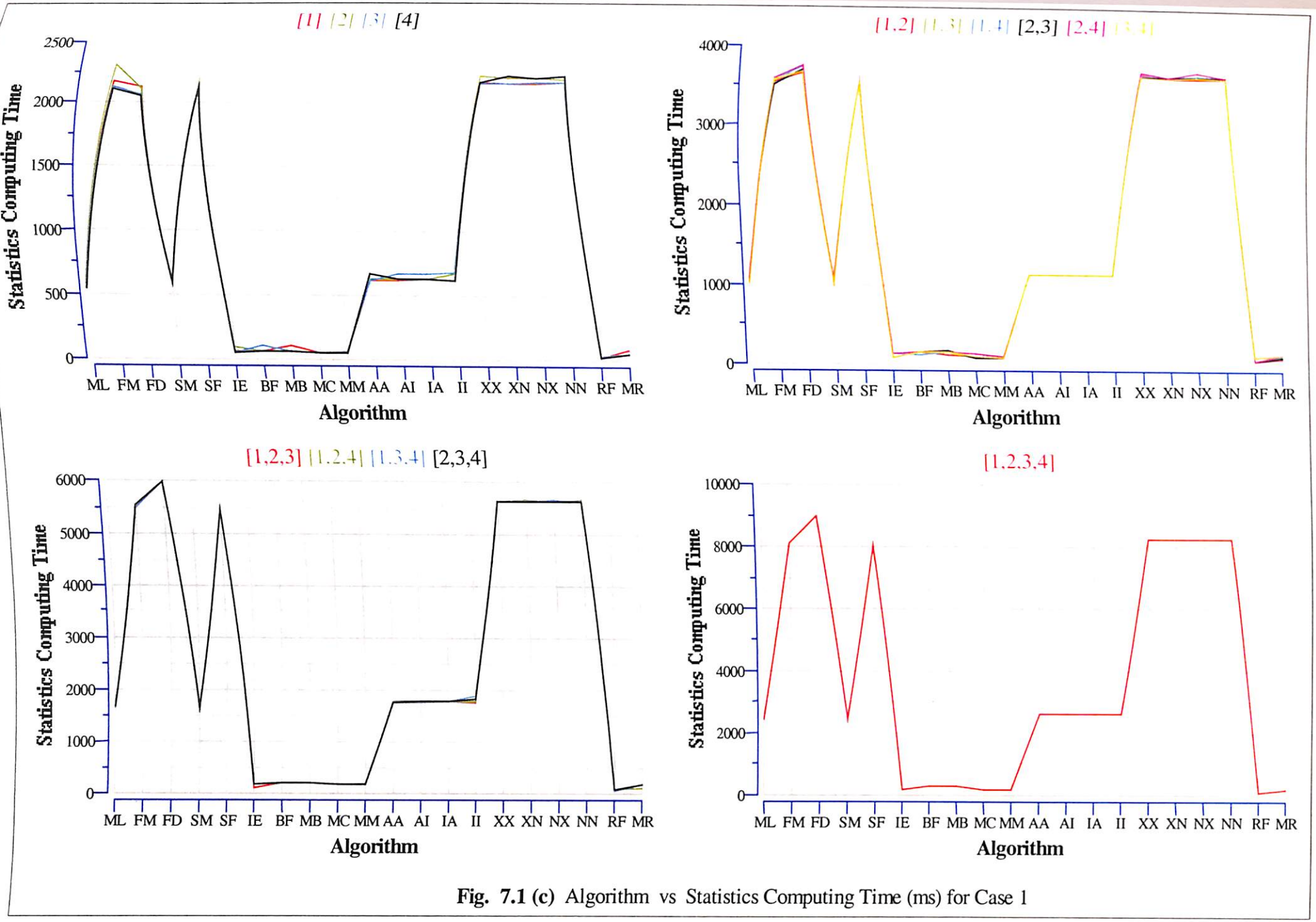


Fig. 7.1 (c) Algorithm vs Statistics Computing Time (ms) for Case 1

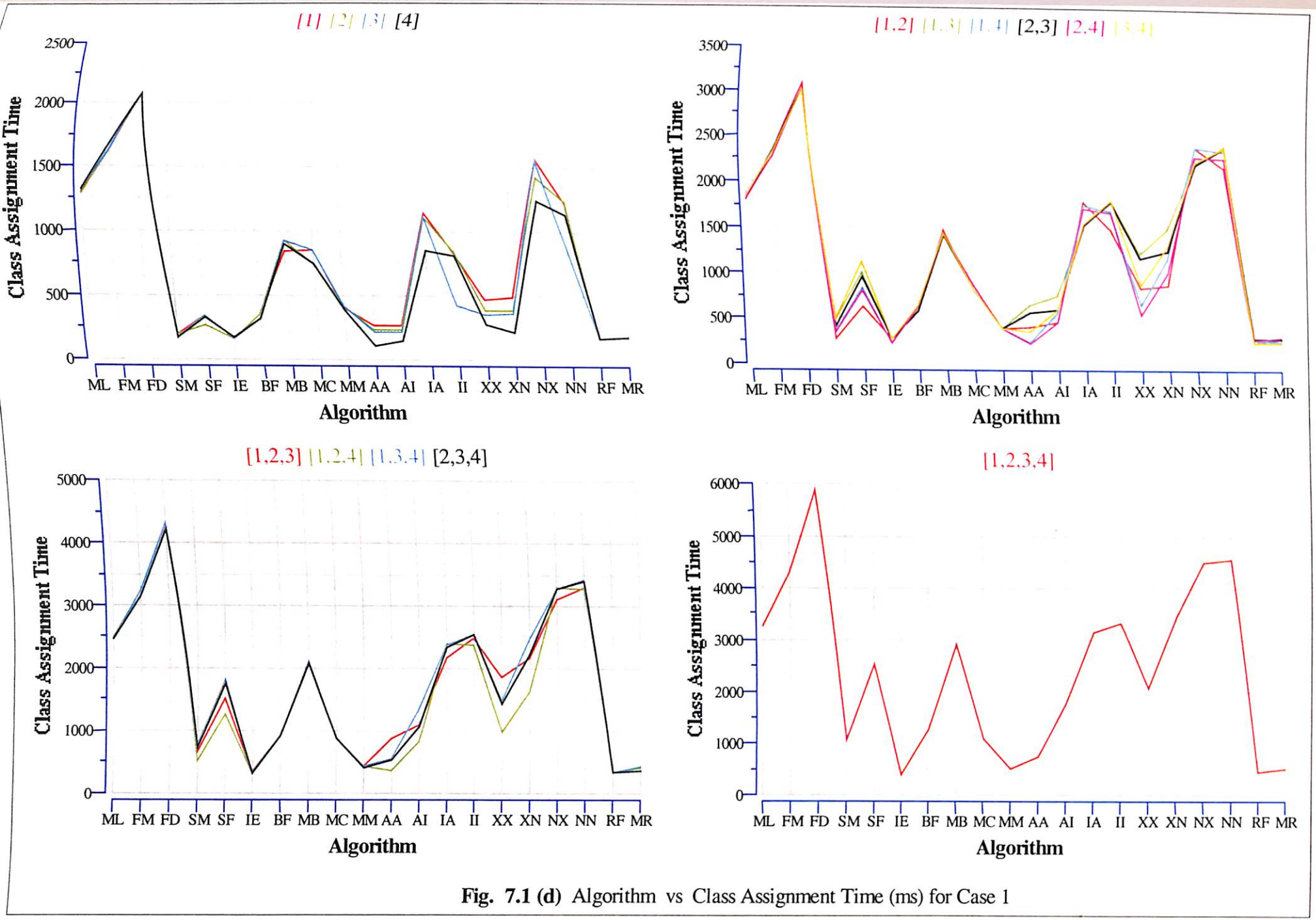


Fig. 7.1 (d) Algorithm vs Class Assignment Time (ms) for Case 1

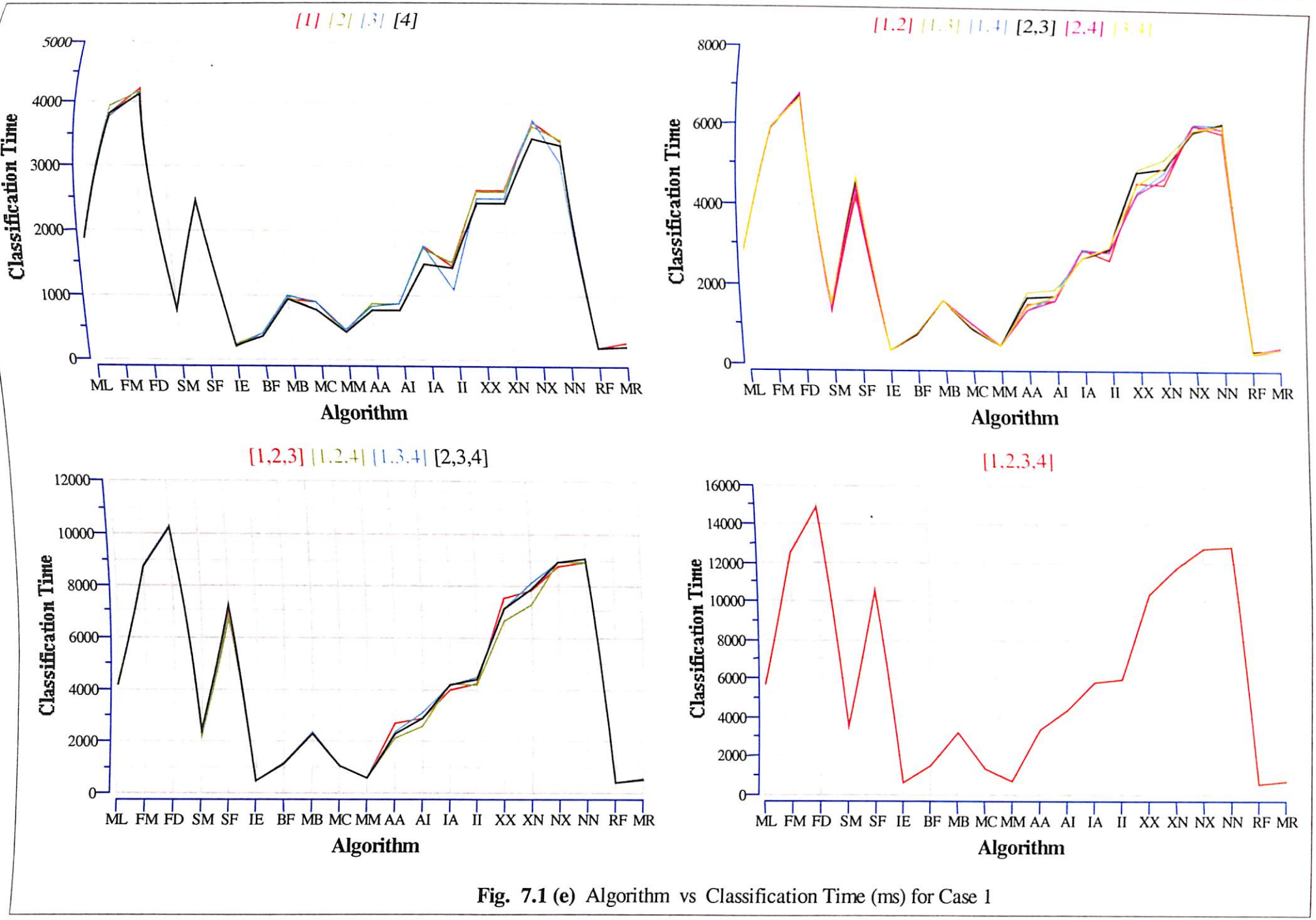


Fig. 7.1 (e) Algorithm vs Classification Time (ms) for Case 1

7.2.2 Interpretations of the statistics and plots of Case 2

From Table 6.4 it is observed that four classes are overlapping and from Tables 6.3 and 6.4 it is observed that the training sites of case 1 are relatively less overlapped than the training sites of case 2. Fig. 7.2 (a) reveals the following:

- In single bands, a number of methods (ML, FM, SF, IE, MC, MM, IAA, XX, XN, NX, NN) achieve the maximum OA of around 50% in single band [4].
- In combinations of two bands, the method IE increases the maximum OA to around 58% in bands combination [3,4]. The methods IE, FM, XN, NX, NN and ML achieve 57%, 54%, 54%, 53%, 53% and 52% OA respectively in bands combination [1,4]. The band-combinations [3,4] and [1,4] are almost equally discriminate.
- In combinations of three bands, the results are more or less similar to the results obtained in combinations of two bands. An important feature is that maximum OA does not increase significantly. A number of methods (FM, IE, XN, NX, NN) achieve the maximum OA of about 58% in band-combinations [1,3,4] and/or [2,3,4]. However, as noted above IE achieves same 58% OA by using three bands also.
- In combination of four bands, FM, XN, NX and NN achieve the maximum OA of around 59% followed by IE (58%) and SF (57%).
- These observations indicate that the OA achieved by IE is comparable to the OA achieved by FM and NN, and it is higher by a significant factor (4%-6%) as compared to MLC. It is also indicated that single bands [3] and [4] are the most separable, and in combinations of single band, a number of methods achieve maximum OA that is comparable with or higher than that of MLC. In combinations of multi-bands, IE, FM, XN, NX, and NN provide the higher maximum OA than that of MLC.

Fig. 7.2 (b) and sub-section B2 of appendix B also support the observations of Fig. 7.2 (a) as follows:

- The maximum value of KHAT is 37%, 40%, 41% and 44% in single band [4], and in band-combinations [3,4], [1,3,4] and [1,2,3,4] respectively. Band-combinations [1,3,4] and [2,3,4] give about the same results.
- The covariance and corelation matrices (sub-section B2 of appendix B) of the classes also consistently confirm the separation of data in the corresponding bands as observed above.

With regard to Figs. 7.2 (c)-(e) the observations are similar to the observations with case 1, and some important features are:

- In single bands, the methods SM and IE require almost same CAT, which is respectively, $1/20^{th}$ and $1/10^{th}$ of time taken by MLC. CAT taken by SF is $1/4^{th}$ of that of MLC. CAT taken by IE is $1/5^{th}$ of CAT taken by SF.

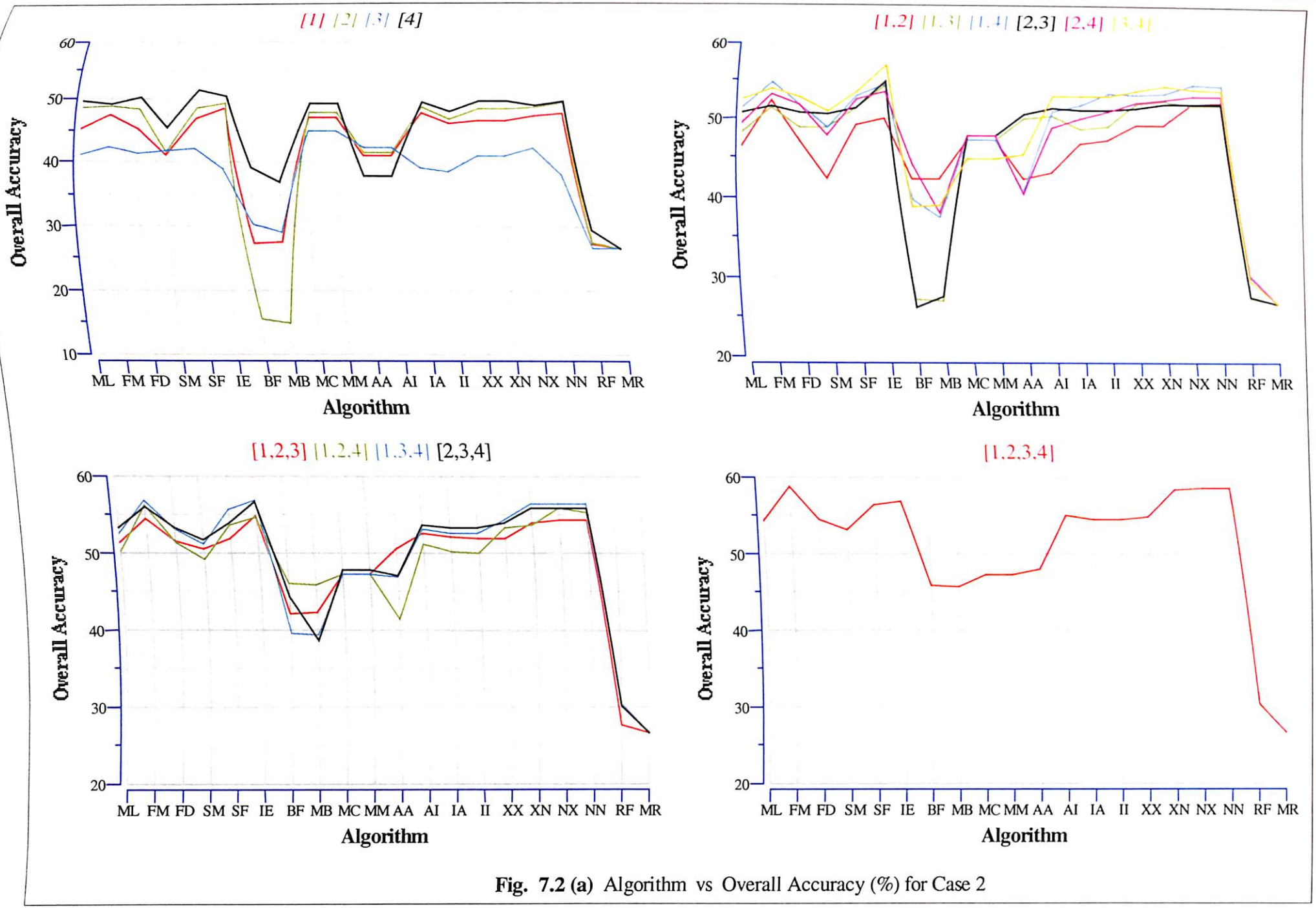


Fig. 7.2 (a) Algorithm vs Overall Accuracy (%) for Case 2

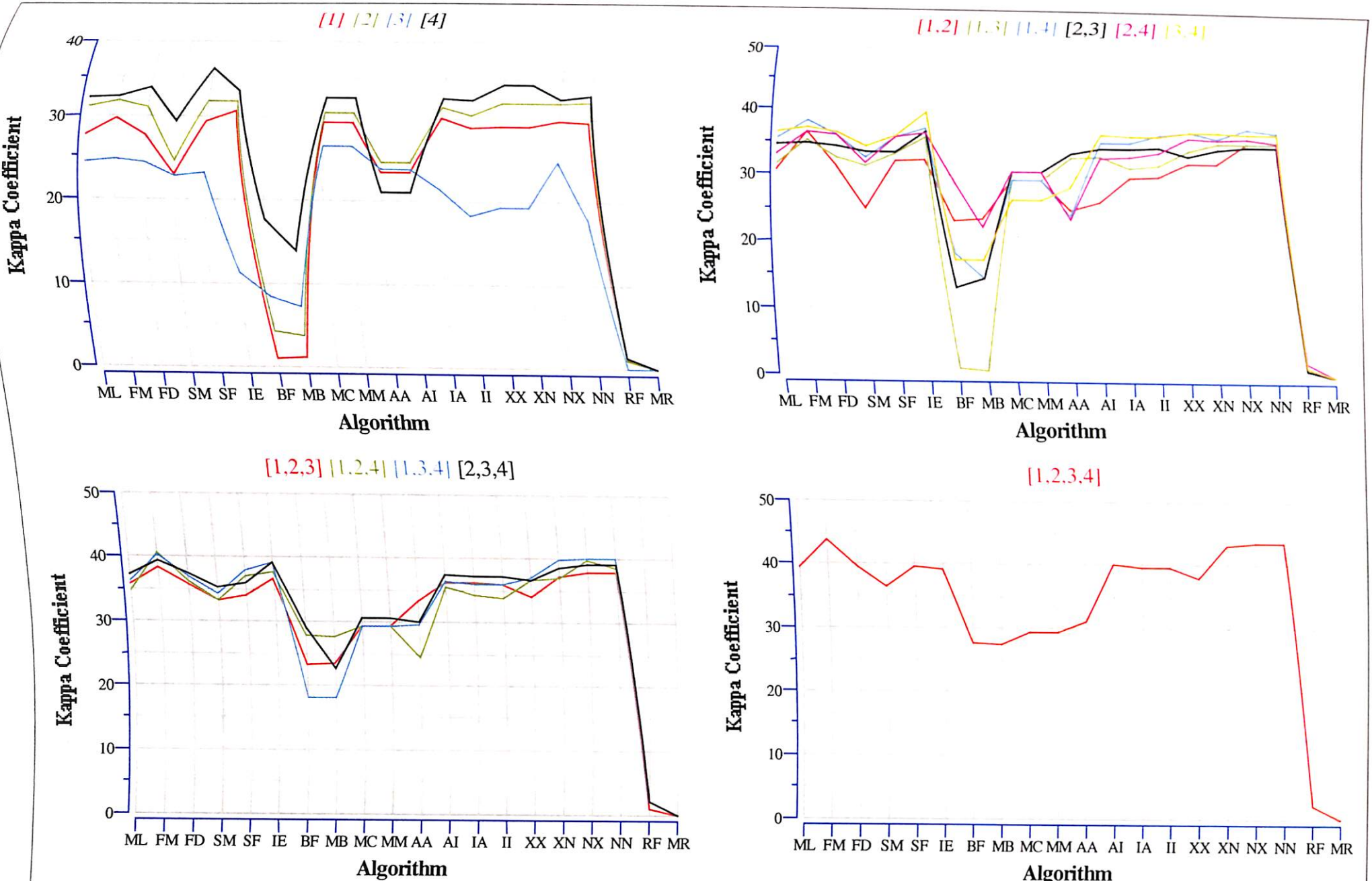


Fig. 7.2 (b) Algorithm vs Kappa Coefficient (%) for Case 2

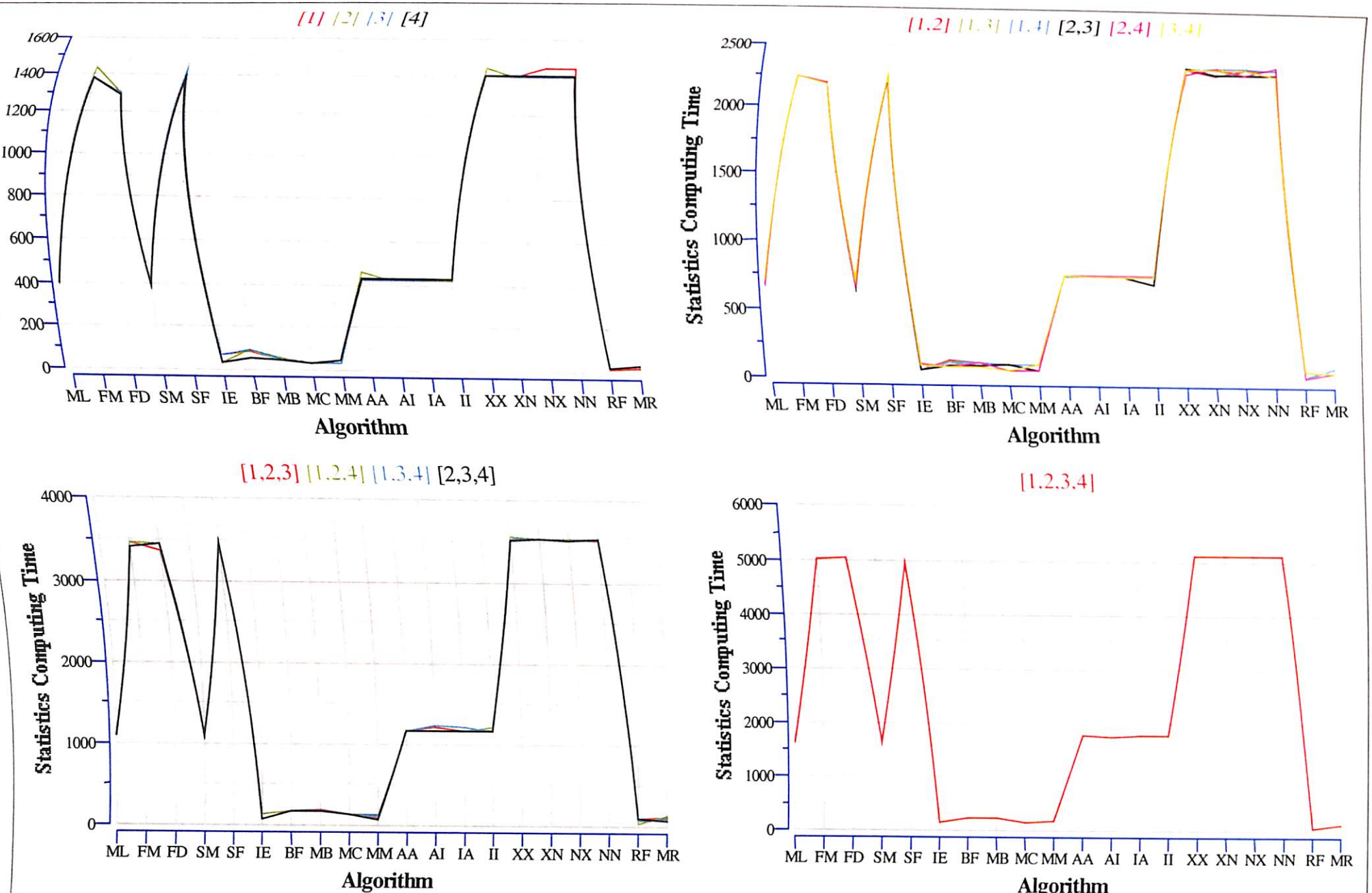


Fig. 7.2 (c) Algorithm vs Statistics Computing Time (ms) for Case 2

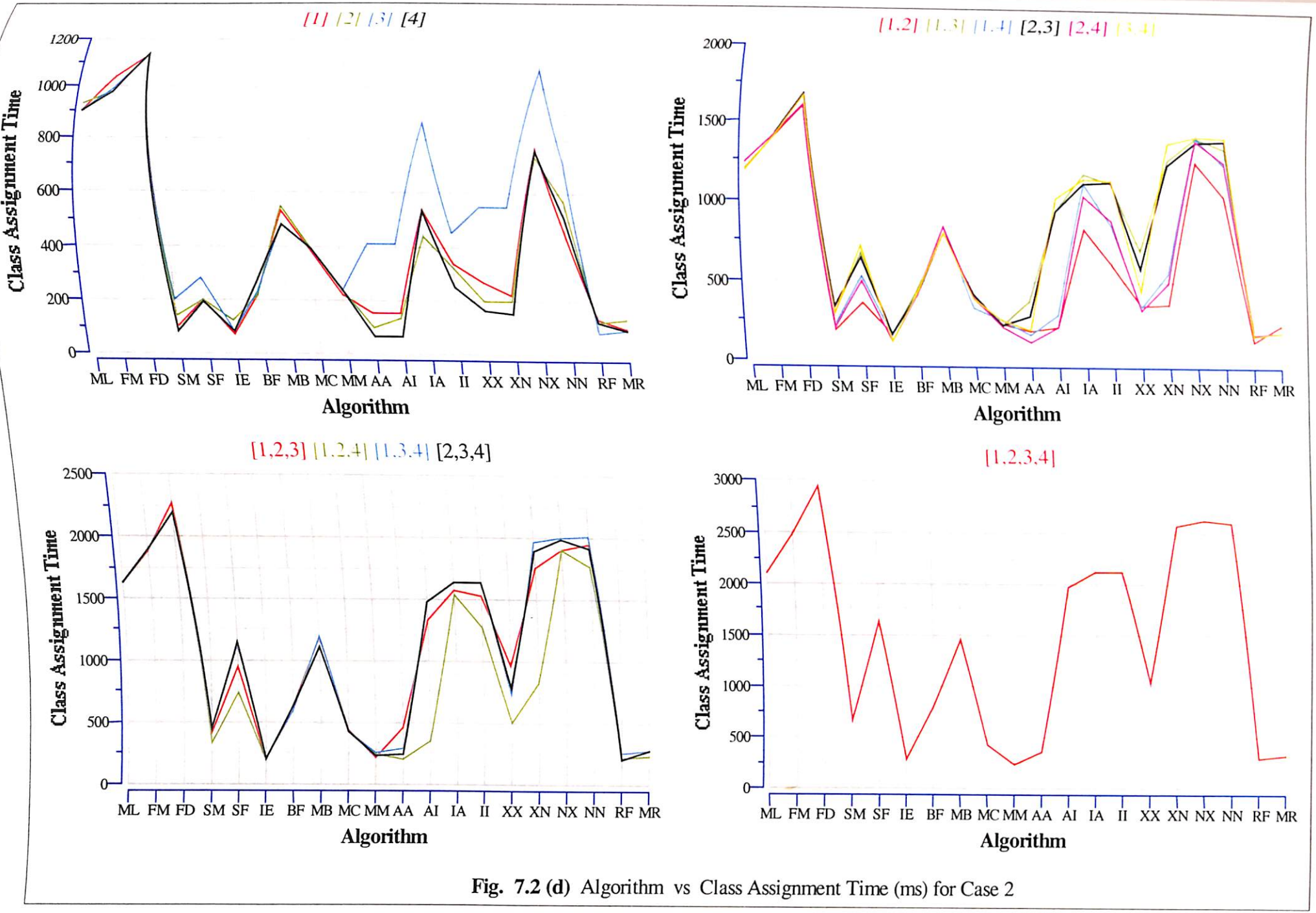


Fig. 7.2 (d) Algorithm vs Class Assignment Time (ms) for Case 2

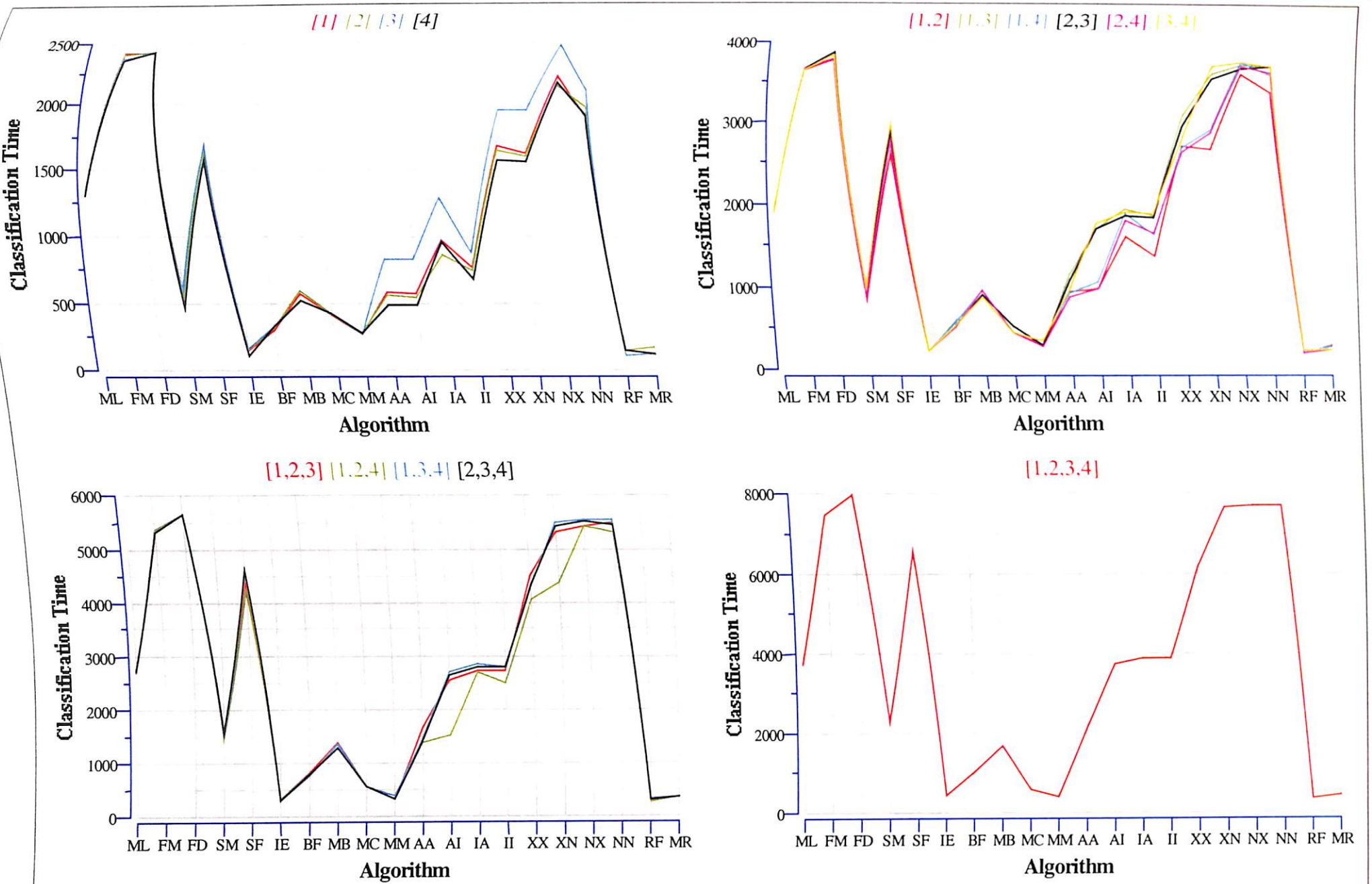


Fig. 7.2 (e) Algorithm vs Classification Time (ms) for Case 2

- In combinations of two bands, the methods IE and SF are faster by 12 and 2 folds respectively as compared to MLC. In combinations of three bands, the methods IE and SF are respectively 8 and 1.5 folds faster than MLC. In the combination of four bands IE and SF are respectively 10 and 4/3 times faster than MLC.

Keeping accuracy and time as selection-criteria, it is revealed that IE is the most suitable method followed by SF, NX, and NN for discriminating a classification problem similar to case 2.

7.2.3 Interpretations of the Statistics and Plots of Case 3

From Table 6.5, it is noted that the training sites of case 3 are less separable because only one class pair (Crop and Tree) out of three pairs is clearly separable. Fig. 7.3 (a) reveals the following:

- In single bands, a number of methods (FM, SF, XX, XN, NX and XN) achieve the maximum OA of 65% in band [3]. In combinations of two bands, the OA increased to around 71% in FM, XX and NX in bands combination [2,3]. In combinations of three bands, the methods FM, NX, and NN achieve maximum OA of around 76% in bands combination [1,3,4]. In combination of four bands, the method FM along with NX and NN gives the maximum OA of around 78%.
- These observations indicate that the band [3] is the most separable band and in combinations of single band, a number of methods give comparable OA which is 3% - 4% higher than the OA achieved by MLC, in combinations of multi-bands the methods FM, NX, and NN achieve the maximum OA which is 1% - 3% higher than the OA achieved by MLC.

Fig. 7.3 (b) and sub-section B3 of appendix B also support the observations of Fig. 7.3 (a) as follows:

- The maximum value of KHAT is 58%, 58%, 63% and 63% in single band [3], in bands combination [2,3], [1,3,4] and [1,2,3,4] respectively.
- Data is more separable in bands 3 and 4.
- The covariance and corelation matrices (sub-section B3 of appendix B) of the classes also consistently confirm the separation of data in the corresponding bands as observed above.

With regard to Figs. 7.3 (c)-(e) the observations are similar to the observations with cases 1 and 2. However, an interesting finding is that in bands combination [1,3,4], the methods FM and NN provide results better than or as good as those achieved by MLC in all four bands. In this case FM and NN takes 25 ms less CAT than that taken by MLC. Keeping accuracy as selection-criteria, it is revealed that methods FM and NN are equally suitable for discriminating a classification problem similar to case 3.

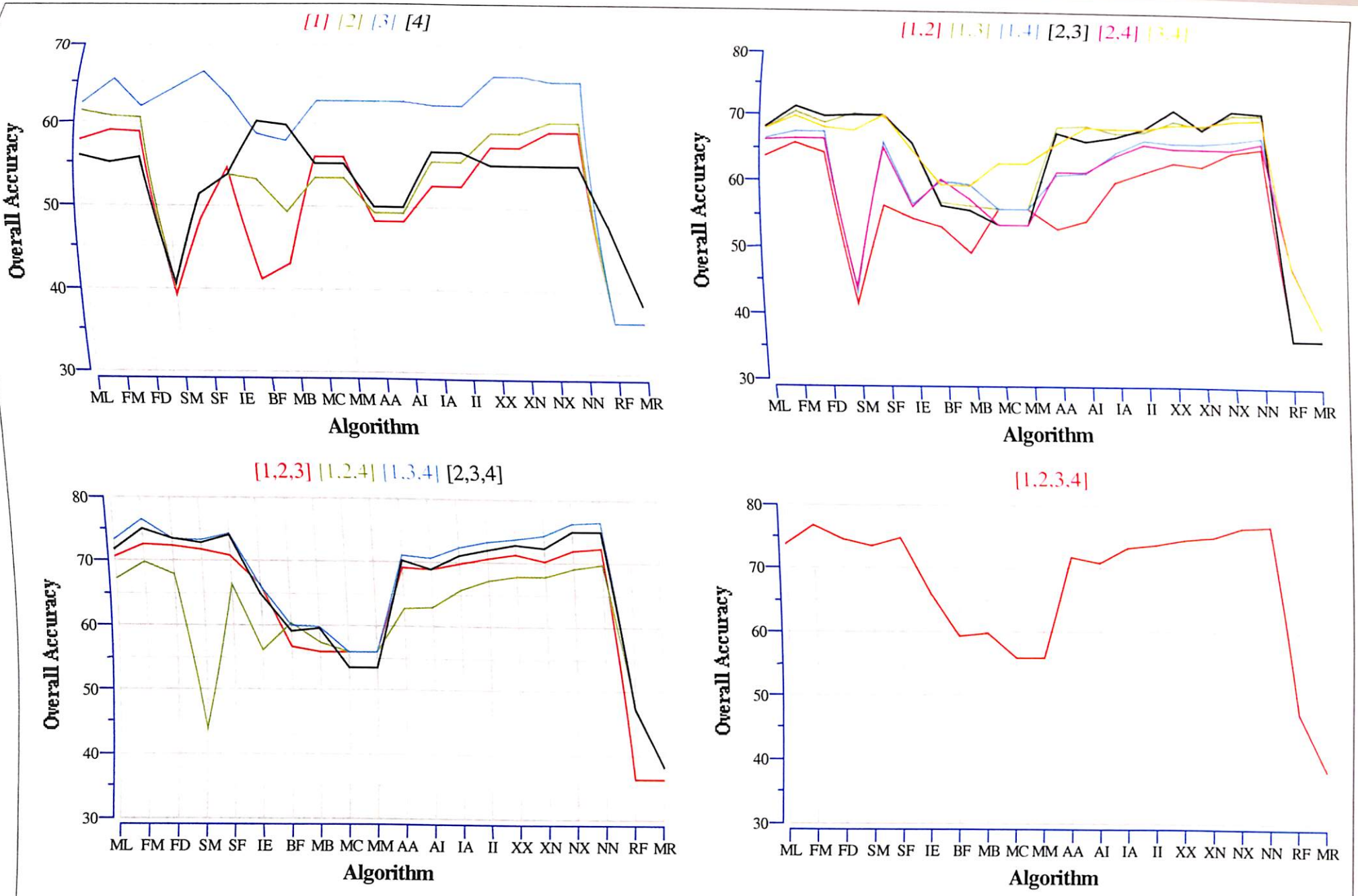


Fig. 7.3 (a) Algorithm vs Overall Accuracy (%) for Case 3

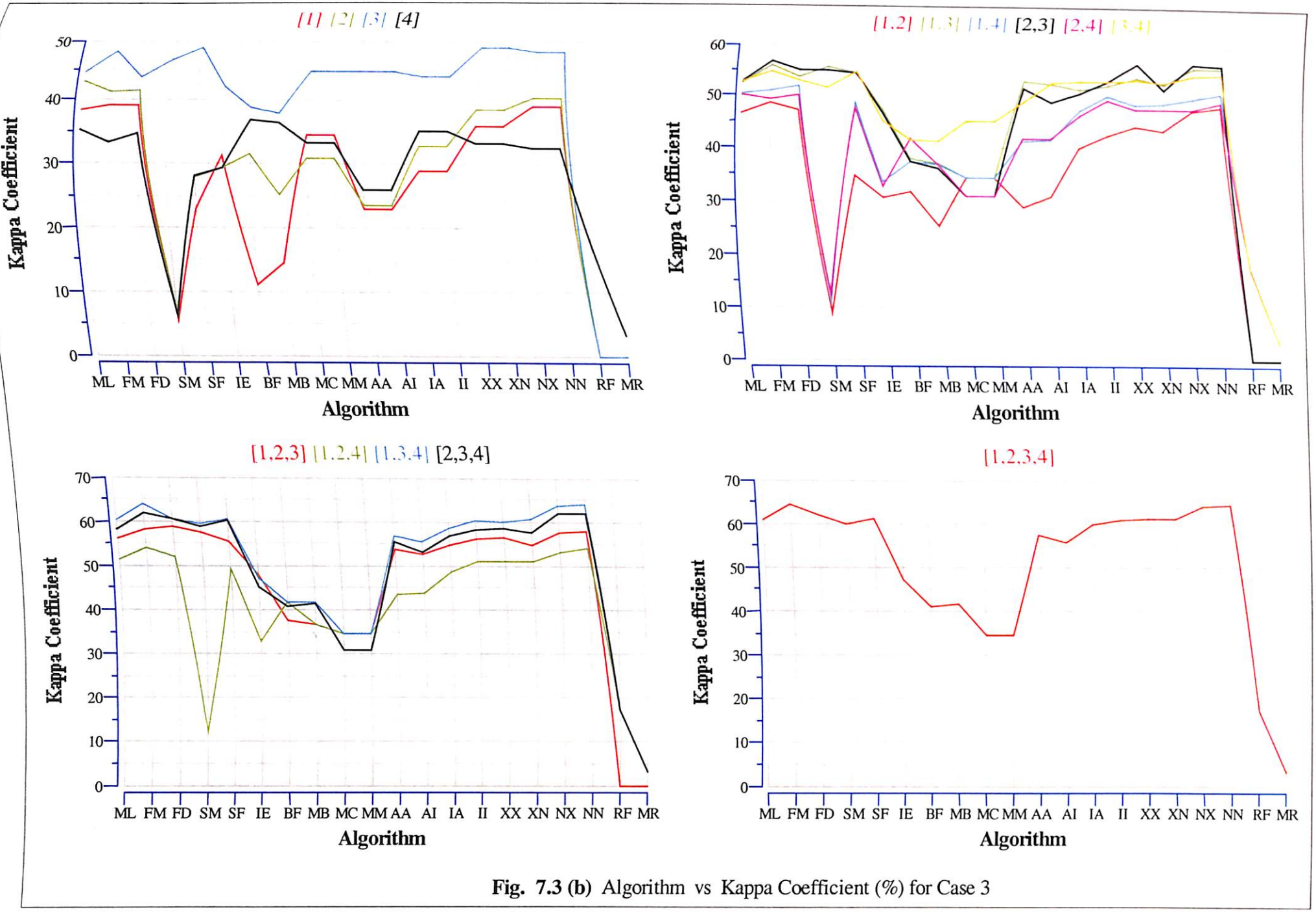


Fig. 7.3 (b) Algorithm vs Kappa Coefficient (%) for Case 3

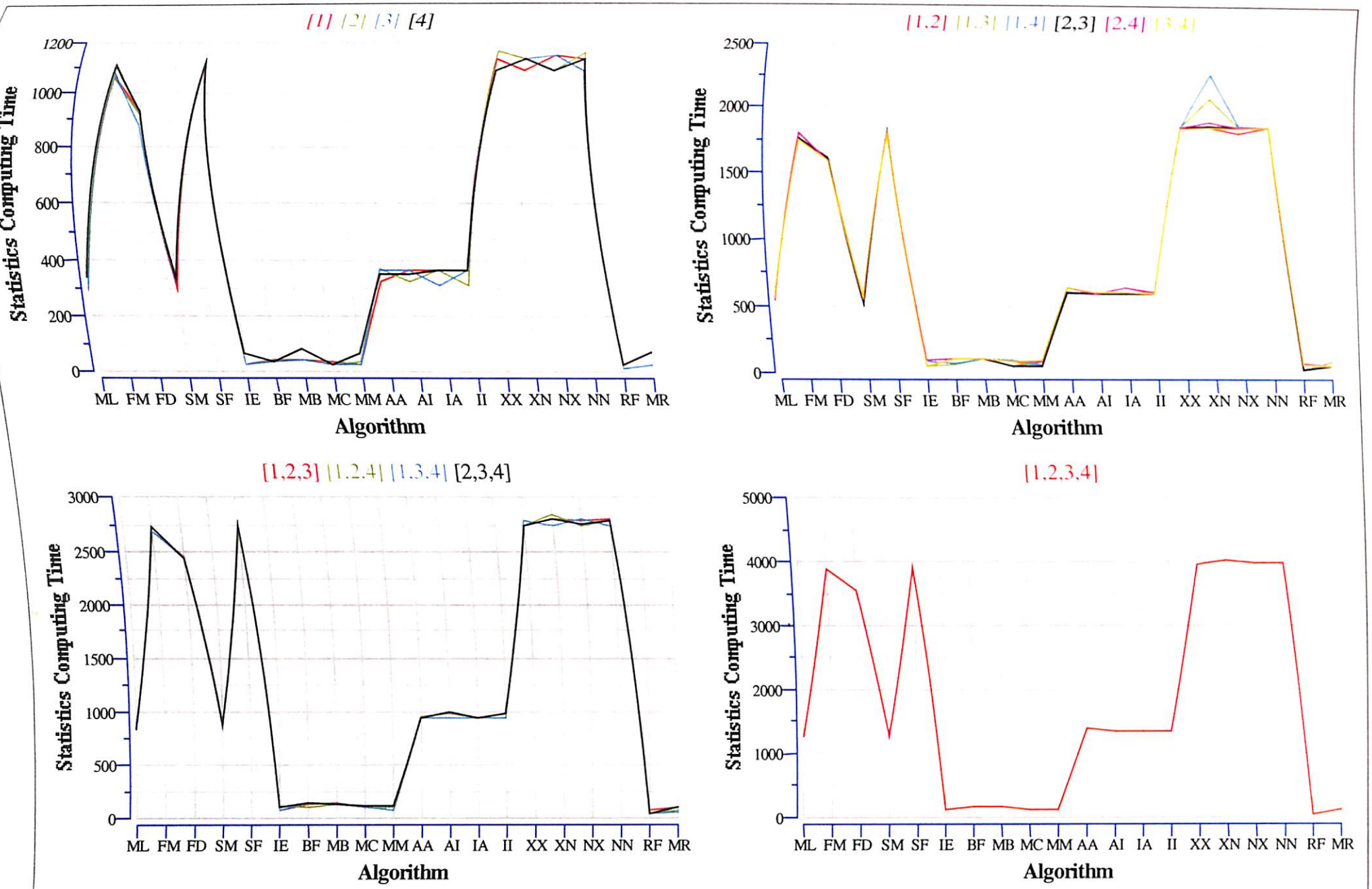


Fig. 7.3 (c) Algorithm vs Statistics Computing Time (ms) for Case 3

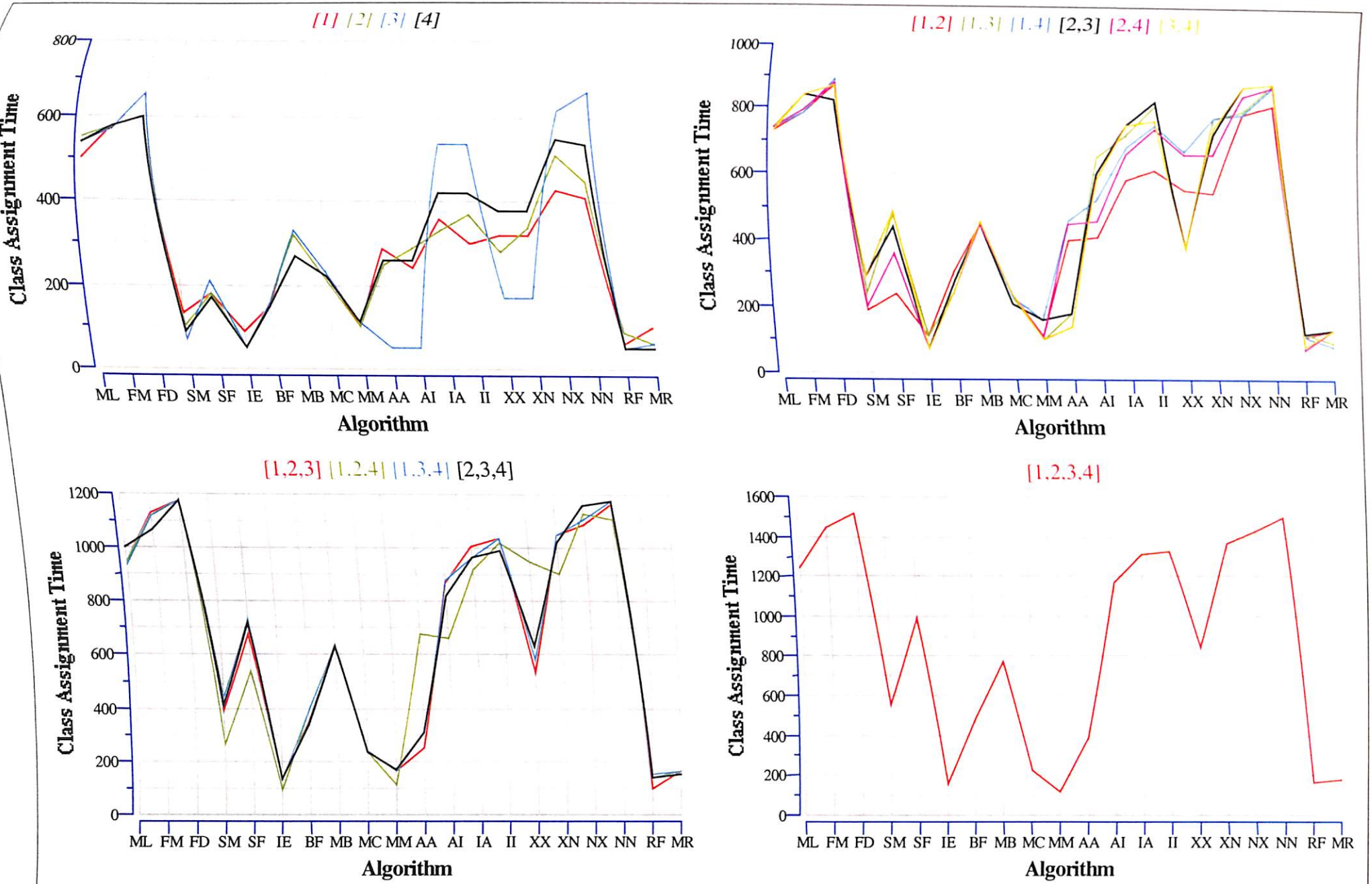


Fig. 7.3 (d) Algorithm vs Class Assignment Time (ms) for Case 3

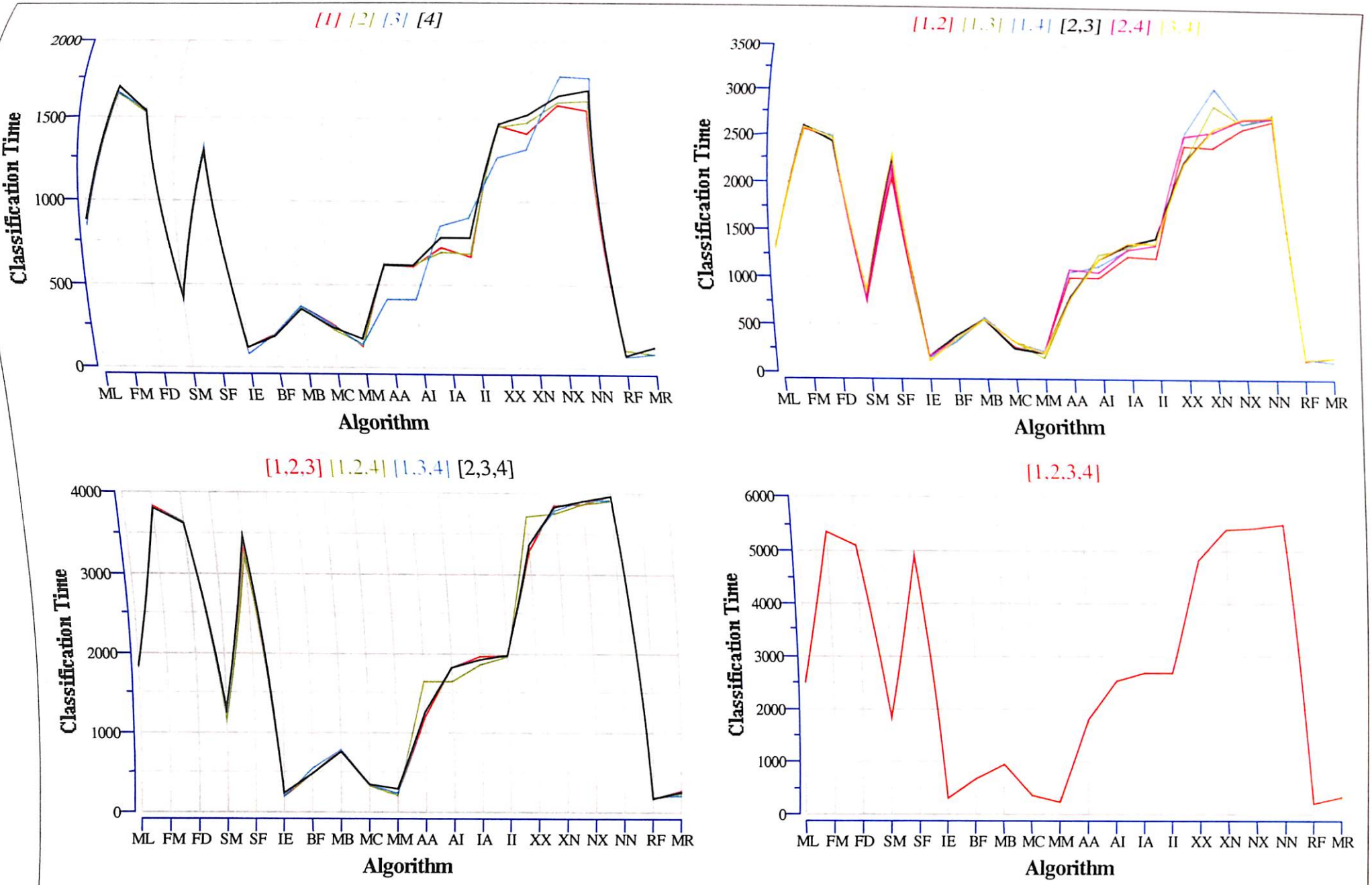


Fig. 7.3 (e) Algorithm vs Classification Time (ms) for Case 3

7.2.4 Interpretations of the statistics and plots of Case 4

From Table 6.6, it is noted that the training sites of case 4 show transformed divergence of 94% and hence, case 4 is a clear case of separable classes. Fig. 7.4 (a) reveals the following:

- In single bands, almost all the methods achieve the maximum OA of 90% except the methods SM, RF, and MR. The most separable band is [3].
- In combinations of two bands, the OA increased to around 95% by almost all the methods except SM, RF, and MR. The most separable bands combination is [3,4].
- In combinations of three bands, except SM, RF, and MR all other methods achieve maximum OA of around 95% in band-combinations [2,3,4] and [1,3,4].
- In combination of four bands, almost all the methods except SM, RF, and MR give the maximum OA of around 95%.
- In this case of clear separation of classes, it is seen that many methods show almost same OA. This is due to high separability of two classes, which is 94% as depicted by transformed divergence. In most of the band-combinations the univariate fuzzy methods BF and EF achieve OA, which is comparable to the OA achieved by MLC in the corresponding band combinations.

Fig. 7.4 (b) and sub-section B4 of appendix B also support the observations of Fig. 7.4 (a). The maximum KHAT achieved is around 80%, 90%, 90% and 88% in single band [3], and in band-combinations [3,4], [2,3,4] and [1,2,3,4] respectively. The covariance and corelation matrices (sub-section B4 of appendix B) of the classes also consistently confirm the separation of data in the corresponding bands as observed above. With regard to Figs. 7.4 (c)-(e) the observations have almost the same nature as case 3. However, some interesting findings are:

- The univariate fuzzy methods, BF and IE are faster in assigning classes than MLC by 3 and 10 folds respectively. In single bands, FM takes only 25 ms extra CAT than that taken by MLC.
- In combinations of single bands, IE is about 10 times faster than MLC and OA achieved is 2% higher than that achieved by MLC. In band-combinations, [2,3,4] and [1,3,4] OA achieved by IE is 2% less as compared to MLC, but speed up gained by IE is 11 folds and in combination of four bands IE achieves 3.5% less OA but achieves a speed up of 12 folds as compared to MLC.

Keeping accuracy and time as selection-criteria, it is revealed that IE is the most suitable method followed by MM, MC, IA and AA for discriminating a classification problem similar to case 4. It is also revealed that processing in multi-bands improves OA only by small amounts (1%-2%) as compared to processing in single band. The OA achieved by a number of methods (FM, FD, XX, XN, NX, NN) in combinations of two bands is almost equal to the OA achieved by these methods in combinations of three bands.

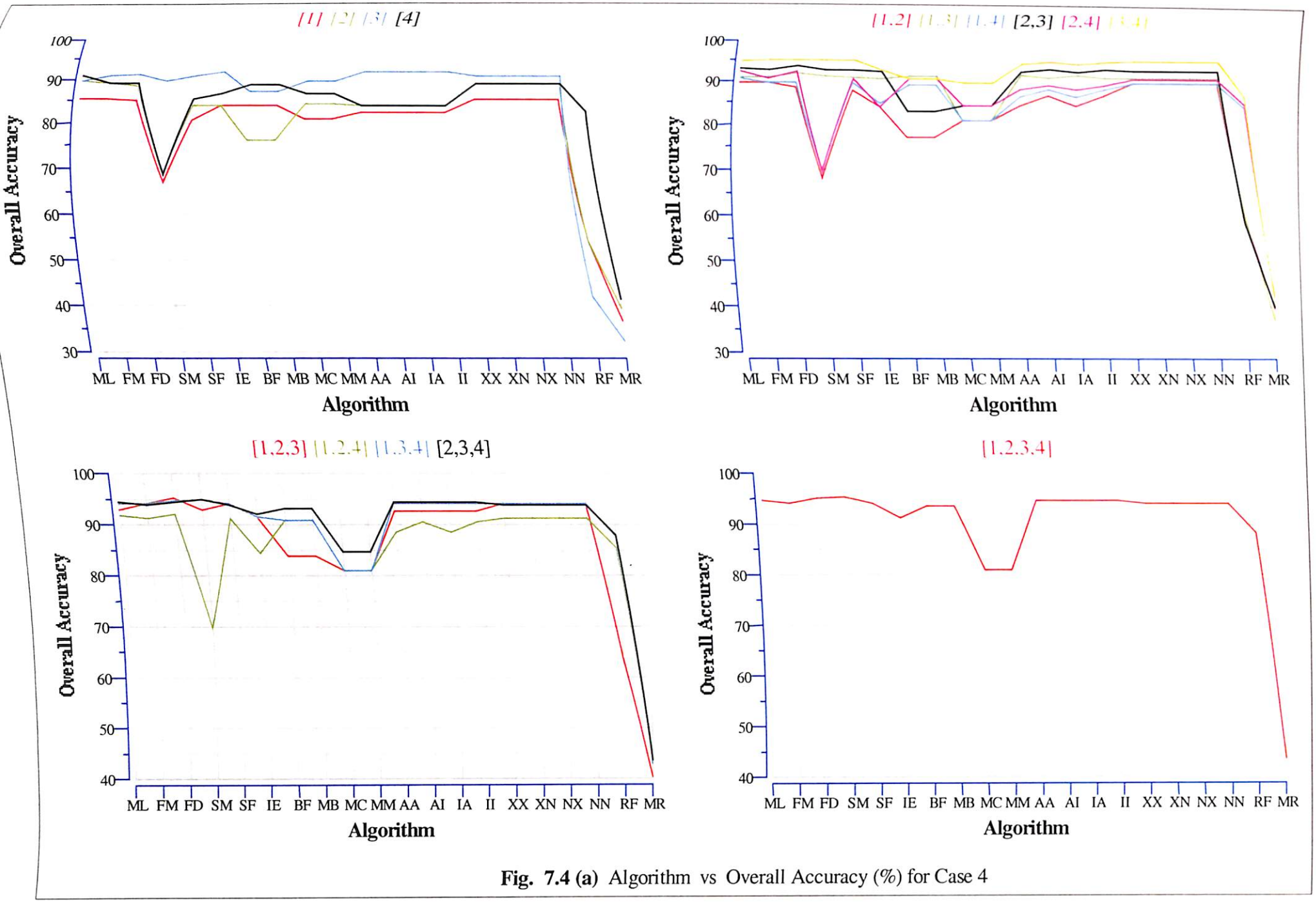


Fig. 7.4 (a) Algorithm vs Overall Accuracy (%) for Case 4

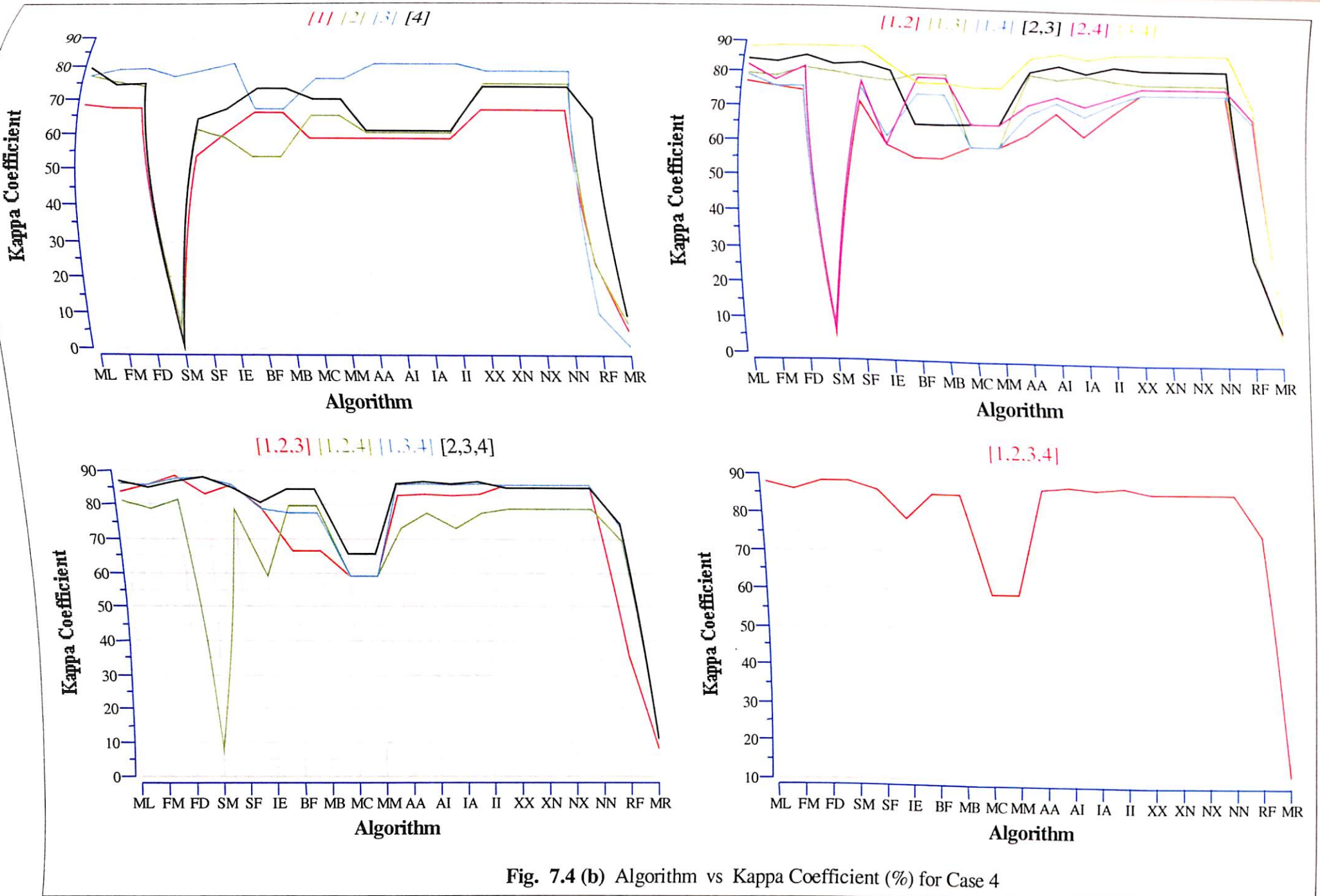


Fig. 7.4 (b) Algorithm vs Kappa Coefficient (%) for Case 4

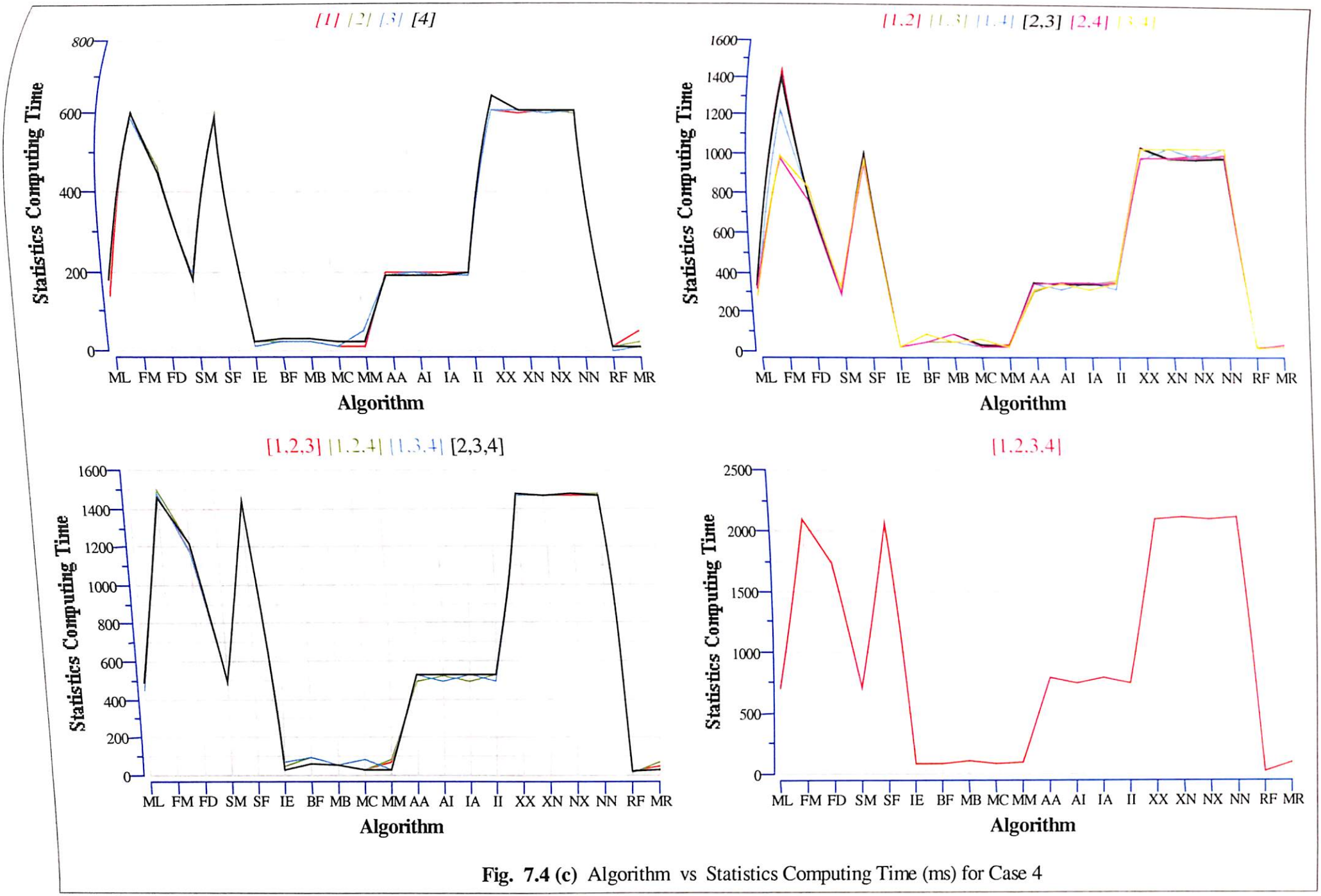


Fig. 7.4 (c) Algorithm vs Statistics Computing Time (ms) for Case 4

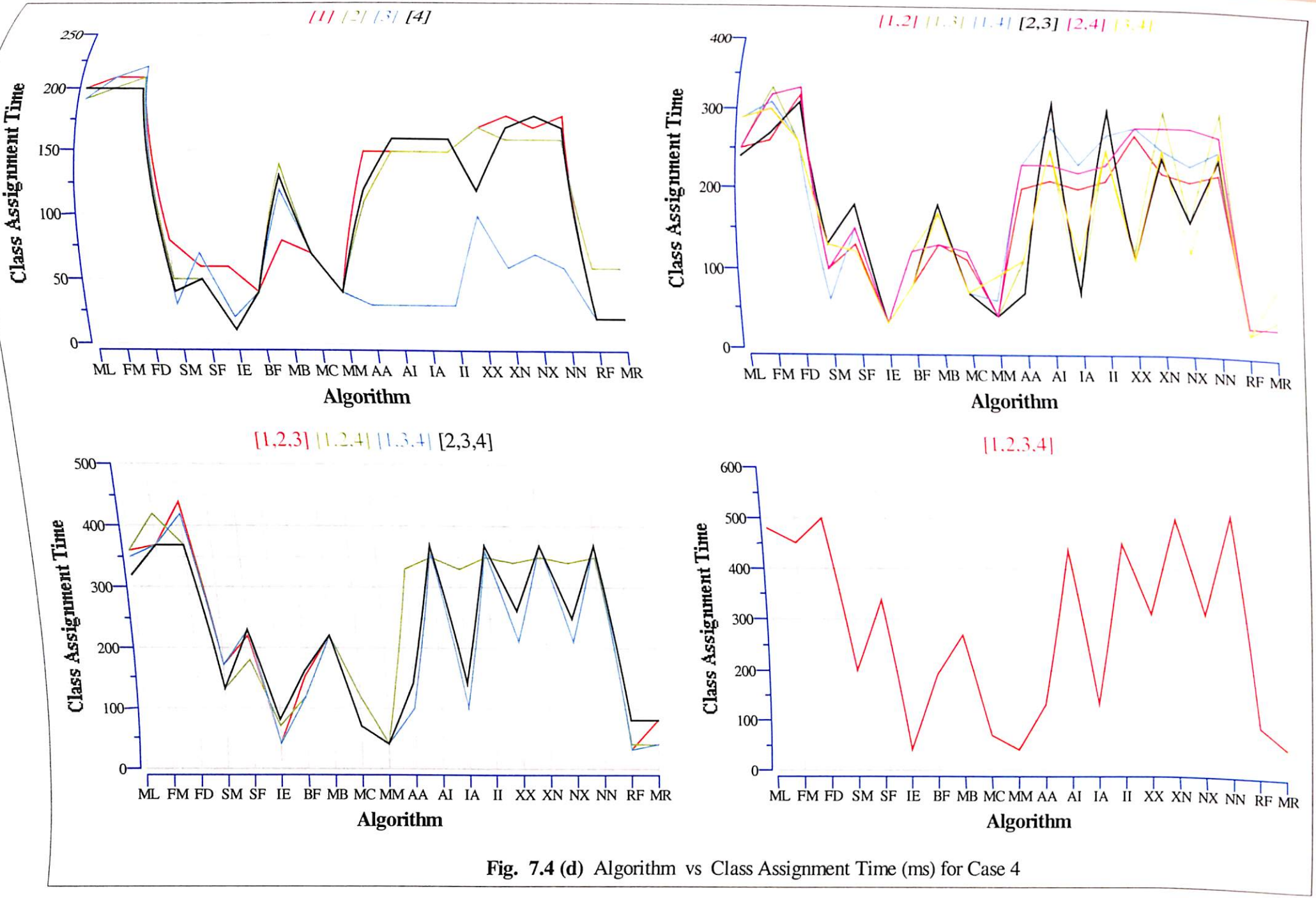


Fig. 7.4 (d) Algorithm vs Class Assignment Time (ms) for Case 4

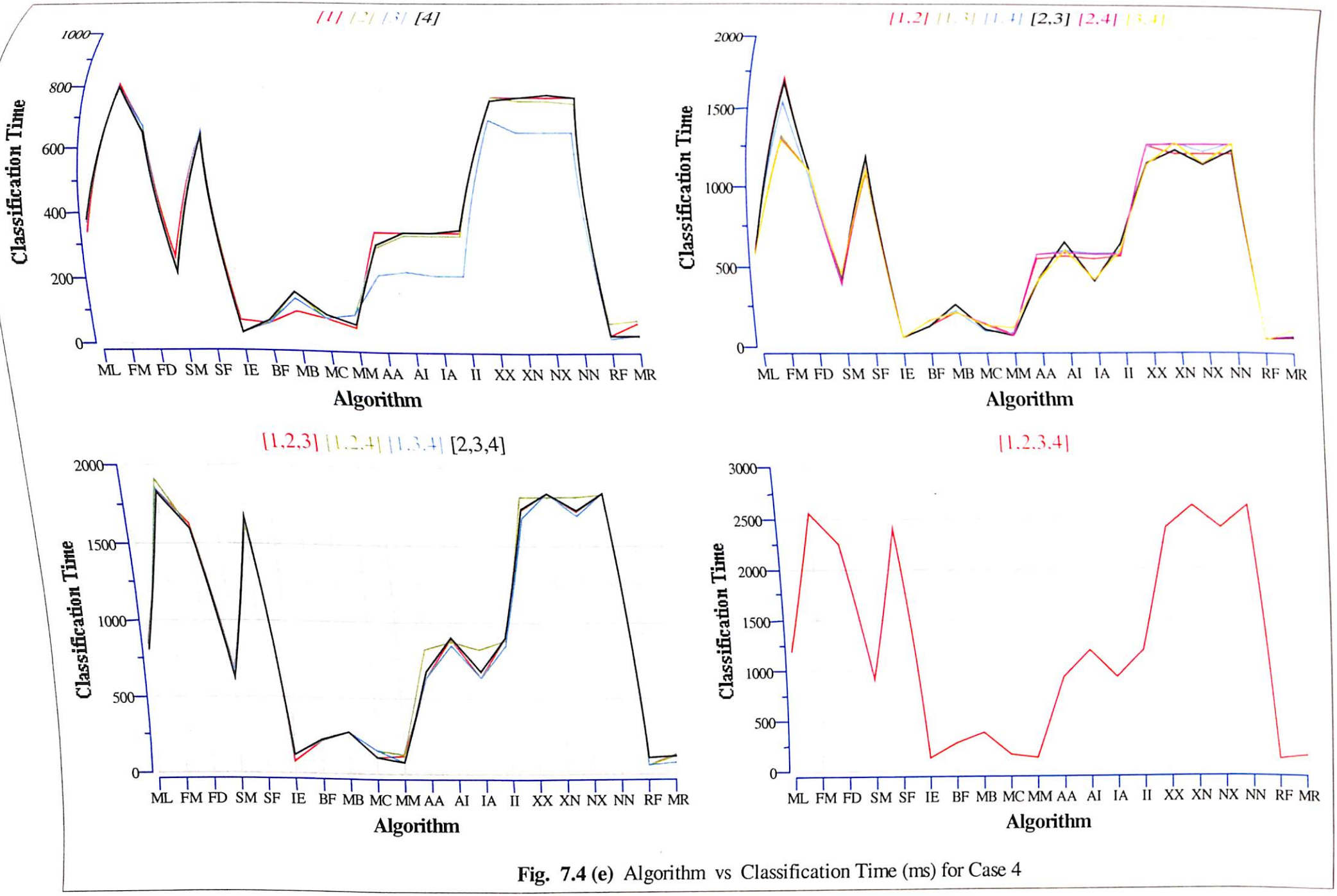
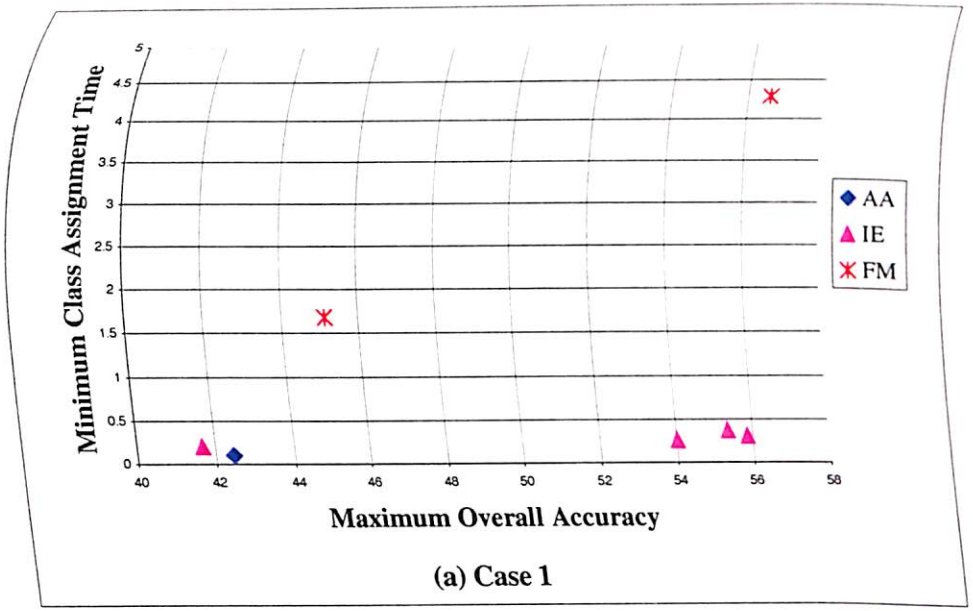


Fig. 7.4 (e) Algorithm vs Classification Time (ms) for Case 4

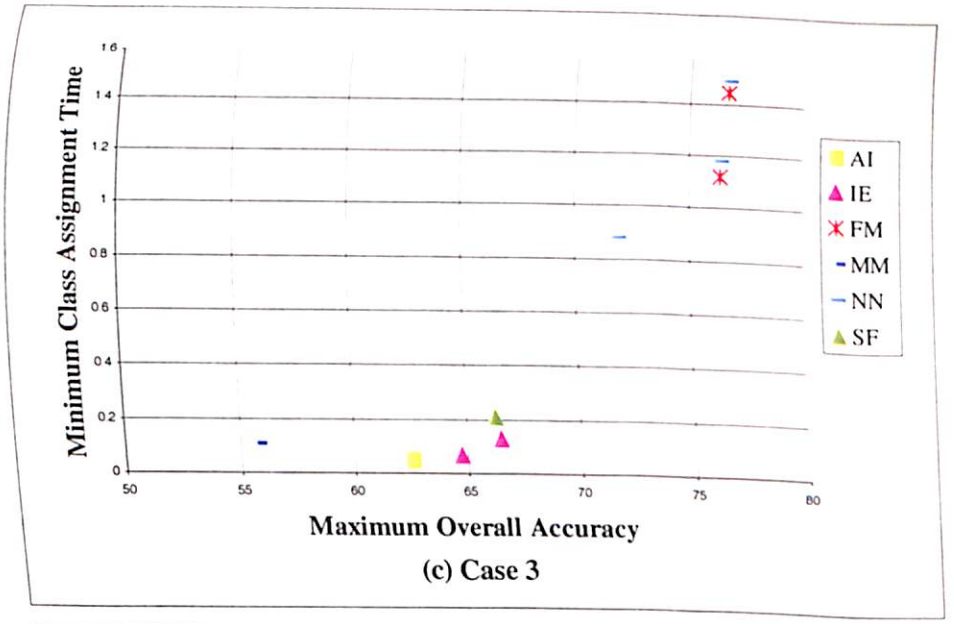
7.2.5 Interpretations on Maximum Overall Accuracy and Minimum Class Assignment Time

For the four cases, investigation is performed by observing Figs. 7.5(a), 7.5(b), 7.5(c) and 7.5(d), which have been generated for the case 1, 2, 3 and 4 respectively by selecting the method demonstrating either maximum OA or minimum CAT (inclusive of all tie situations) in band combinations of 1, 2, 3, and 4 bands, and then generating a scatter plot between maximum OA (x-axis) and minimum CAT (y-axis). Various band combinations are not indicated in this plot because whether we get maximum OA using less or more bands; maximum OA with minimum CAT using less or more bands, or minimum CAT using less or more bands the algorithm is a good classifier. The following are some observations from Figs. 7.5 (a-d):

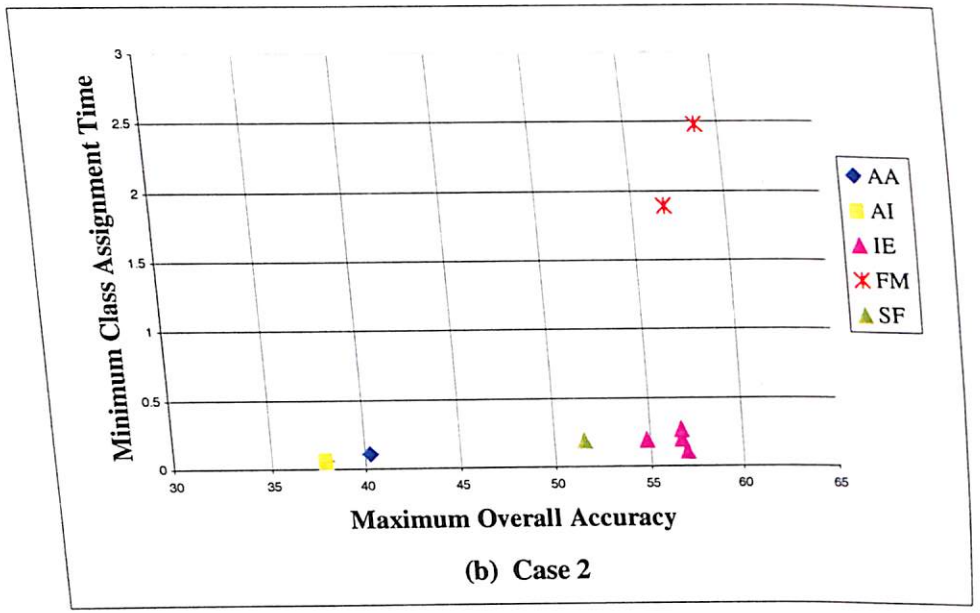
- The method, IE achieves OA in the range 54%-56% in case 1 and requires CAT of less than 0.4 sec. The methods IE and SF achieve OA ranging between 52%-58% in case 2 and require CAT of less than or about 0.3 sec. In case of less separable classes, the methods IE, AI, SF, MM, NN, and FM achieve OA between 56%-77% and require CAT of less than 1.0 sec, except FM and NN for which it is 1.0 and 1.5 sec respectively. The methods IE, AI, and SF require CAT less than 0.25 sec.
- In the separable case most of the classifiers achieve OA above 90%. It is also observed that in the cases of overlapping and less separable classes, multivariate method FM achieves OA about 57%-59% but it requires high CAT (2-2.5 sec). On the other hand, the method SF, which first try to classify a pixel by using univariate analysis and if the pixel is labeled as 'unclassified' by univariate processing, then use of multivariate processing is made, takes CAT near about 0.3 sec.
- The univariate fuzzy methods RF and MM, in the case of separable classes, achieve OA, which is comparable with the OA achieved by multivariate methods FM and FD. This suggests that in clearly separable cases fuzzy classifiers discriminate classes simply by using fuzzy membership functions defined in terms of lower bounds, upper bounds and mean in various bands. As compared to MLC, higher speeds of classification and comparable OAs are achieved, irrespective of number of classes as long as the classes are clearly separable.
- The OA achieved by FD, FM, XX, XN, NX, and NN in case 4 is around 90% which is very close to the OA achieved by MLC. This indicates that in case of separable classes we can achieve OA around 90% simply by processing data in single bands. This reduces classification time over MLC.
- Absence of MLC in the Figs. 7.5 (a-d) reveals that MLC neither achieves maximum OA nor performs classification with minimum CAT. Hence, as compared to most of the fuzzy classifiers proposed in the present research, performance of MLC is inferior both in terms of OA and CAT.
- It is also indicated that the methods IE, AI and SF achieve OAs of order 50-70% by classifying 863-1555 pixels within time-span of 0.1-0.5 sec in cases of overlapping and less separation of classes. In case of clear separation, a number of fuzzy classifiers (FM, SM, XX, XN, NX, and NN) achieve OA of order 92%-95% by classifying 485 pixels in a time-span of 0.05-0.5 sec. The classifiers IE, AI and XX are very fast because they require time-span of only 0.05–0.1 sec to classify 485-1555 pixels.



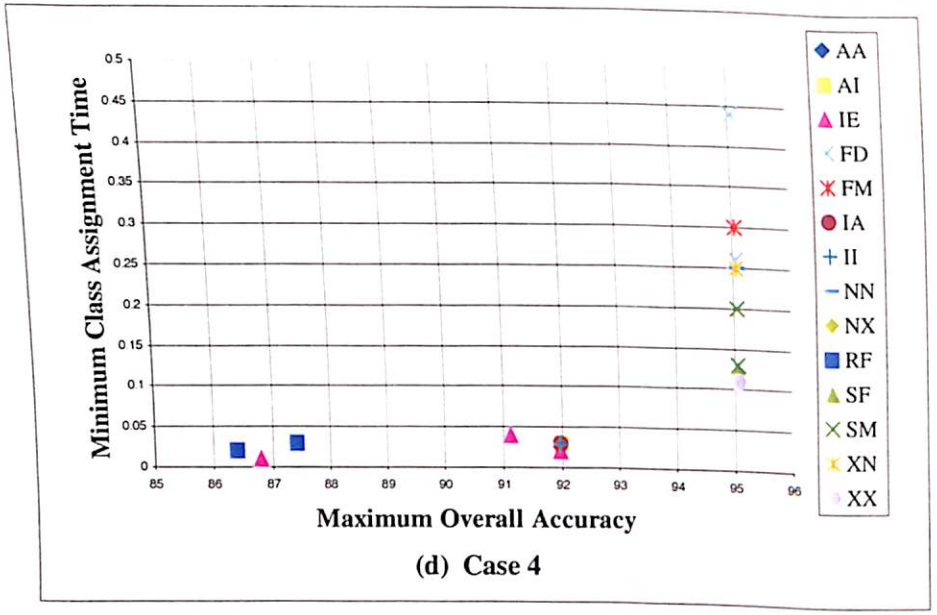
(a) Case 1



(c) Case 3



(b) Case 2



(d) Case 4

Fig. 9.5 Performance of Classifiers with respect to Maximum Overall Accuracy and Minimum Class Assignment Time

7.3 Analysis of the Results

Based on interpretations presented in the section 7.2 we can have analysis with respect to ten criteria as follows.

7.3.1 Overall Accuracy (OA %) and KHAT

One variant, where the coefficient of overlapping, k, is set to unity, of the proposed statistical classifier achieves 1% to 3% more overall accuracy than that of MLC. The overall accuracy achieved by other four variants of statistical classifier is comparable with that of MLC and these are 3 folds faster in classification than MLC.

With the use of fuzzy logic techniques, a fuzzy classifier assigns partial memberships to the pixels, enabling more accurate statistical parameters. These classifiers perform better in classification of mixed pixels and overlapping classes. With respect to the fuzzy classifier which is implemented in five variants, three variants, where the coefficient of overlapping, k, is set by using MaxMin or MinMax or MinMin rule, of the proposed new fuzzy method perform better than MLC with an improvement in overall accuracy ranging in 3% to 5%. In combinations of three bands, upper bound is 5.5 %. The variant that sets the coefficient of overlapping, k by MaxMax rule, demonstrates only marginal improvement in the overall accuracy as compared to MLC. Except, where k is set to unity in separable case, all other variants achieve better overall accuracy in the corresponding case. The exceptional case demonstrates less accuracy due to high degree of statistical separation. MLC has advantage to produce an efficient adaptation to the training sites due to the use of second order statistics (covariance matrices) [Melgani et al. 2000]. Present work also supports this result and also proves that the adaptation increases with increasing statistical separation among the classes. This demonstrates that the proposed method performs at least as good as MLC in terms of overall accuracy and the variants which use MaxMin or MinMax or MinMin rule demonstrate better overall accuracy as compared to MLC.

The beta-distribution based univariate classifiers are believed to encompass any shape of the histogram and hence probability distribution. However, the results of overall accuracy are significantly lower (with an exception of the results obtained by including two most discriminating bands in the cases of clear separation of classes) than that of MLC. This can be attributed to three reasons. First, the more adaptation of MLC to the training sites (even when less number of sites are used) as compared to univariate beta-distribution based fuzzy classifier. Second, the estimation of the parameters of beta-distribution solely on the basis of lower bound, upper bound, mean and standard deviation, and third, probably the errors due to classification of

7.3.5 Maximum Overall Accuracy and Minimum Class Assignment Time

From the discussion in section 7.2.5 and from Fig. 7.5, it is revealed that the methods IE, SF, AI, NN, RF, SM, II, NX, XN, and XX does not achieve OA higher than some of multivariate classifiers but these methods require CAT less than one second in some of the band combinations. An observation regarding the case of separable classes is that the performance of fuzzy classifiers is consistent (achieving almost same OA in many of the band combinations) and the time taken is less than 0.5 sec.

The method AA, though very fast and at one point in the Fig. 7.5 it shows OA around 40%. One should keep in mind that this is the maximum OA achieved by this method among the 20 methods in some band combination. Similarly, AI achieves about 38% OA. This type of consistent performance can be attributed to their implementations in fuzzy environment. The methods SF and SM are faster by 4 and 4.5 folds respectively than the method FM. Further, the methods RF and MM although can not resolve overlapping cases but in the separable case these methods achieve OA which is comparable with OA achieved by FM and FD. These being univariate are about 8 to 10 times faster than multivariate methods FM and FD. Further, it can be noted that the conventional method MLC does not appear in Fig. 7.5. This means that MLC neither demonstrate maximum OA nor classifies pixels in less time as compared to many methods developed in the present research.

7.3.6 Complexity of Coding the Algorithm

The classification algorithms operating in multivariate mode and the algorithms based on beta-distribution are complex. Algorithms operating multivariate mode are complex due to involvement of operations on matrices in multivariate. The algorithms based on beta-distribution are complex due to two main reasons. First, computationally intensive calculation of gamma function, which raises overflow error while computing gamma of a number greater than 149 even when we use 64-bit floating-point arithmetic. So to handle this overflow beyond this limit the value of gamma function is approximated. Second, as these algorithms make use of the range (lower and upper bounds of pixel-values in various bands) associated with the training samples. The range of pixels values training samples in various bands does not necessarily be the range depicted by the test pixels. In the cases when lower limit of pixels-values of test samples in a band is lower (in corresponding band) than that of training samples and similarly, when upper limit of the pixel-values of test samples in a band is higher (in corresponding band) than that of training samples we can not use gamma distribution. In such situations, instead of beta-distribution, normal distribution is used only for the pixels outside the ranges depicted by training samples. The fuzzy methods based on multivariate normal distribution are also difficult to code because of calculation and storage of initial fuzzy memberships, which requires calculations in multivariate. All other methods are easier to code.

7.3.7 Integration of Algorithm with GIS

In general, the spatial data of GIS is organized in different layers and hence, the algorithms based on univariate analysis can easily be integrated with GIS. GIS is a useful tool to simulate what-if type of scenario and analysis, and during this simulation it is expected to perform its computations fast and accurately. So the suitable algorithms for such type of tasks are the univariate analysis based algorithms due to their simplicity, better scalability and high speed. However, one has to choose only the methods, which provide OA comparable with or higher than that of MLC. Only two algorithms have these properties, namely improved explicit method and the classifiers based on beta-distribution (only in certain band combinations). These are fast and individual bands can represent different layers to be used in GIS applications. If more discriminating bands are selected then processing only in these more discriminating bands can further improve the speed with only marginal loss (if any) in the classification accuracy. The improved explicit fuzzy method can demonstrate 12 folds of speed up over MLC in case of single bands and still accuracy is comparable to that achieved by MLC.

7.3.8 Feasibility of Algorithm for Online Classification

In the recent past some faster methods [Melgani et al. 2000, Jia and Richards 2003] for image classification have been proposed and they can further be improved both in terms of overall accuracy and classification time by using contextual information. But, the measures used for establishing degree of the proximity are very complex and nature of their effect is not well established [Gonzalez and Woods 1993]. This suggests that to make classification faster, one can attempt to implement classifiers in univariate mode. Melgani et al. [2000] illustrate an example of such type of classifier. However, to establish feasibility of an algorithm for online classification there are other factors like image-size, number of bands, selection of a subset of available bands, intended use of classified data, etc. apart from speed of the classifier. Hence, the algorithms developed in the present work are all unsuitable for on-line classification. However, the improved explicit classifier can be used for on-line classification if it is supported by an expert system. Moreover, as the univariate algorithms, being more modular and as they operate in each band independently at several stages, their implementation on parallel machines is favored, and thus, classification can achieve further speed-ups.

7.3.9 Scalability of Algorithm in Terms of Number of Bands

In general, the univariate analysis and fuzzy algorithms are less sensitive to scaling in the terms of bands. The efficiency of an algorithm also depends on the subset of bands and the number of bands selected for data representation for subsequent classification. It is observed from the Figs. 7.1 – 7.4 that the improved explicit fuzzy method and algorithms based on beta-distribution are comparatively less sensitive to scaling than other

univarite based algorithms and multivariate based algorithms. Other univariate fuzzy algorithms are also less sensitive to scaling as compared to multivariate statistical classifiers. As seen for the cases 1 and 2, the improved explicit fuzzy method achieves higher accuracy when only three, instead of four, bands are taken. It achieves higher (5.5%) overall accuracy than that achieved by MLC. This significant gain shows that a suitable selection of bands can increase the overall accuracy in the case of overlapping classes. An advantage of improved explicit method is that in most separable 2 bands, it performs better than other methods and also the speed-up gained is of high order. For example, OA achieved by it is 2% more than that achieved by MLC and sped up is around 10 folds as compared to MLC.

7.3.10 Effect of Fuzzy Membership Function and its Modulation

A suitable choice of the fuzzy membership function can not only improves the overall accuracy significantly and makes the classifier faster both in the terms of calculation of signatures of training data and CAT. The improved explicit fuzzy method when applied on overlapping pixels represented in the combination four bands, an improvement between 3.5% and 4.5% in the overall accuracy has been observed as compared to overall accuracy achieved by MLC. The CAT taken by improved explicit fuzzy method is as low as one-ninth of the time taken by ML classifier. Hence, speed up of nine folds over MLC is achieved when we incorporate the proposed improvement. This happens due to modulation of standard derivation and better estimates of the parameters due to the proposed improvement.

On the other hand, the method MM (not using any modulation) does not perform well as compared to MLC. This is because there is no modulation of the parameters. The methods where range has been used to define fuzzy membership function do not perform well because parameter estimation is based only on ranges and/or mean. So, we do not get better estimates of the fuzzy memberships and hence overall accuracy achieved is lower than that achieved by MLC.

CHAPTER 8 Conclusions

In the present work, the problem of classification of remote sensing data has been investigated, and five new statistical supervised classifiers, 13 new fuzzy supervised classifiers and an improved explicit fuzzy supervised classifier are proposed and implemented using *LPA WinProlog* and its associated tools. Performance of these proposed classifiers is compared with the conventional multivariate classifier, namely, Maximum Likelihood Classifier (MLC), to overcome the problems caused by mixed and overlapping classes. These classifiers are compared in context of their practical applications in Geographical Information Systems (GIS). The overlapping among the class-pairs is decided by transformed divergence taken to the scale of 100. Divergence, and Jaffrie-Matusita (JM) distances are used to further verifications the separability. Based on the separability four cases are investigated – case 1 and 2 contain overlapping classes, case 3 contains less separable classes and classes in case 4 are clearly separable. The following conclusions are drawn from the study:

- With regard to the statistical classifier proposed in five variants, the fifth variant, where the coefficient of overlapping, k, is set to unity, performs 1% to 3% better in terms overall accuracy than MLC in all cases. The overall accuracy achieved by other four variants is comparable with that of MLC and they are about 3 folds faster in classification than MLC in all cases.
- Three variants of the proposed fuzzy classifier, where MaxMin or MinMax or MinMin rules set the coefficient of overlapping, k, perform better than MLC in all cases, with an improvement in overall accuracy ranging between 3% to 5.5%. The variant that sets coefficient of overlapping, k, by MaxMax rule, demonstrates only marginal improvement in the overall accuracy as compared to MLC in all the cases. Except when k is taken as unity in the separable case, all other variants demonstrate overall accuracy higher than or equal to the overall accuracy achieved by MLC in the corresponding case. The proposed method when uses $k = 1$, performs at least as good as MLC and the variants which make use of the MaxMin or MinMax or MinMin rules to set k demonstrate higher overall accuracy than MLC.
- The classifiers based on the univariate beta-distribution do not perform better than MLC in the terms of overall accuracy in all the cases except the separable case where their performance is comparable to that of MLC. However, these speed up the task of classification by three fold as compared to MLC.
- The classifiers based on univariate normal distribution, achieves less overall accuracy than MLC except in certain band combinations in separable case. However, these methods are 9 to 20 times faster than MLC by considering their performance in various band combinations.
- Five important advantages of the improved explicit fuzzy classifier are observed. First, in case of overlapping and mixed classes it achieves higher overall accuracy, which ranges between 3.5% and

5.5%, as compared to MLC. Second, it is about nine times faster than MLC. So, it is suitable for on-line classification. Third, if all classes are overlapping in all the bands, it demonstrates very high accuracy (as compared to MLC) when all bands are used. Fourth, if there are few bands where there is no overlapping or there is less overlapping, improved explicit fuzzy classifier performs better than MLC. Fifth, this method being univariate based, one can upgrade the signatures of training samples by simple calculations, and this improves the performance of the method.

- Due to modulation of standard deviation and better estimates of the expected extends of classes in various bands, the improved explicit fuzzy classifier achieves a significantly high speed up over MLC and still significantly improves overall accuracy of overlapping classes. The selection of suitable fuzzy membership function and then its modulation both tasks are important for fuzzy classifiers.
- Proposed statistical classifier achieves overall accuracy slightly higher than or comparable to that of MLC. However, except the fifth variant and the variant where MaxMax rule sets the value of coefficient of overlapping, the remaining three variants are slower than MLC. Multivariate fuzzy classifiers also achieved similar types of results. However, the range of improvements in overall accuracy over MLC is slightly higher than that of five variants of proposed statistical classifiers.
- The univariate classifiers are 3 to 12 folds faster than multivariate classifiers and the classifiers like improved explicit fuzzy, fifth variants of proposed statistical and fuzzy classifiers achieves 2%-5.5% more overall accuracy than that of MLC. In case of clear separation of classes almost all univariate classifier achieve overall accuracy approximately equal to that achieved by MLC.
- The improved explicit fuzzy method achieves higher overall accuracy in three band-combinations as compared to the overall accuracy achieved by this method in the four bands combination. The overall accuracy achieved by the univariate fuzzy classifiers that make use of beta-distribution, normal distribution or ranges for defining fuzzy memberships are inferior to MLC, even when more bands are used.
- An important observation with respect to the selection of fuzzy membership functions is that only a suitable fuzzy membership function improves the overall accuracy of classification. Use of any arbitrary function, which evaluates fuzzy membership value in the range [0, 1], degrades performance of the classifiers. This is exactly the case with the classifiers, making use of the ranges for finding the fuzzy memberships.
- The univariate beta-distribution based algorithms are believed to encompass diverse shapes of probability distributions of the training samples. However, the obtained results of overall accuracy are significantly lower (with an exception of the results obtained by including two most discriminating bands in the cases of clear separation of classes) than that of MLC. This is mainly attributed to the less accurate estimates of the parameters of beta-distribution, and hence, estimates of fuzzy membership, solely on the basis of lower bound, upper bound, mean and standard deviation.

- Use of Prolog offers added advantages in integrating the algorithms with the both knowledge-based systems and with the systems that also utilize the spatial context, in addition to spectral data, for classification of remote sensing data. Prolog permits easy derivation of new knowledge from existing remote sensing and GIS data even when the data is represented in different forms and available from different and contradictory sources.
- Based on the role played by classification in GIS, the suitability of the univariate classifier is better than multivariate classifiers. The classification is used in GIS for geographic visualization, creating new data, data reclassification, and spatial aggregation. The univariate classifiers, being faster, scalable, and less sensitive to selection of bands are better suited to such utilizations.

Thus, this work presents a wide spectrum of algorithms, which by using various approaches achieve varying overall accuracies with varying classification speeds, and the achieved overall accuracy and required classification time for a classifier both are consistent with the separability of the selected training sites and the bands used for data representation. To integrate the output of classification with GIS, one can choose a suitable classifier from this wide spectrum of classifiers, depending on the desired accuracy, speed, bands used, separability of data and intended application.

CHAPTER 9 *Critical Assessment of the Work and Suggestions for Further Research*

In this research work author proposes new solutions in the field of fuzzy and statistical classification of remote sensing data. Author experimentally and analytically investigates these new solutions for their efficiency and suitability in various conditions of class separability, selection of bands for representing data and selection of fuzzy membership function. In doing so, author has also achieved some unfavorable results. For example, univariate beta-distribution based fuzzy classifiers are expected to provide better results in terms of overall accuracy as compared to MLC. But this expected gain could not be observed from the results. An improvement can be done at this stage by implementing a multivariate beta-distribution. Some gains in overall accuracy are expected because normal distribution based classifiers perform better than the univariate beta-distribution based classifier even in the overlapping cases. Second possibility of the improvement is to modulate beta-distribution based algorithms along the guidelines of improved explicit fuzzy method. However, both of these improvements are complex to implement because in case of multivariate beta-distribution we have to estimate parameters similar to covariance matrices and in case of modulation we have to use two modulation parameters to encompass diverse shapes of probability distributions. Yet another problem is, if data is not unimodal then certain failures are going to occur. In present work, the parameters are estimated by using mean, standard deviation, lower bounds and upper bounds in various bands but by using moments about mean one can do better estimations of the parameters and can achieve higher overall accuracies.

One can argue that when there are plenty of training samples the assumption of normal distribution is appropriate and reasonable and MLC alone can provide more overall accuracy. But, we do not have well established guidelines on the count of training pixels in each class so that the training classes can be treated as normally distributed. Apart from this the distribution should not be multi-modal. The beta-distribution can encompass variety of shapes of the probability distributions simply by using variations in its two parameters. Beta-distribution is expected to handle the situations more accurately than MLC when samples are not unimodal. In this research, no explicit attempts are made to investigate the performance of classifiers for such situations. This can be a topic of further research.

Using some knowledge-based expert systems for classification, one can do other improvements. This is easily possible in present implementation as Prolog programming supports easy & manageable integration of knowledge-bases. As Prolog supports easy integration of contextual information as compared to other

procedural languages like C, Fortran, Pascal etc. one can improve the classifiers by utilizing contextual information.

The fuzzy classifiers using fuzzy membership defined on the basis of range and/or mean demonstrate overall accuracy less than the half of the overall accuracy achieved by that of MLC. This situation can be improved by using more training sites and by assuming univariate normal distribution for the pixels with values beyond the ranges in the respective bands. We can get overall accuracies comparable to that of MLC in separable cases in lesser time because these methods are about 11 times faster than MLC.

It is also possible to calculate other attributes from the existing band data and add these attributes to increase the dimensionality of data. Similarly, one or more bands may be eliminated. Programming has been done to take into account these types of changes in the data. This type of experimental study [Kartikeyan et al. 1995] is expected to give positive results, where the Kartikeyan et al. [1995] have developed an expert system and the results obtained are comparable to MLC when feature expansion is used. In this case we are going to get overall accuracy only at cost of slightly more classification time if we make use of the proposed univariate methods. If we use feature reduction after applying proper feature selection methods like Eigen value method, we can further speed up the classification in the class assignment stage.

Further obvious improvements in classification time can be achieved by using faster methods for matrix inversion, matrix multiplication, and calculation of variance and calculation of covariance and exploiting results derived in matrix algebra like those for symmetric matrices.

16. Bryant, J. (March, 1989). "A fast classifier for image data." Pattern Recognition, 22(2), 45 - 48.

17. Cannon, R. L., Dave, J. V., and Bezdek, J. C. (February, 1986). "Efficient implementation of the fuzzy C-means clustering algorithms." IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(2), 248 - 255.

18. Chang, K. T. (2002). Introduction to geographic information systems. Tata McGraw-Hill Publishing Company Limited, New Delhi.

19. Chen, C. F. (November, 1999). "Fuzzy training data for fuzzy supervised classification of remotely sensed images." Proceedings of 20th Asian Conference on Remote Sensing, Hong Cong, China.

20. Choudhry, S., and Morad, M. (August, 1998). "GIS errors and surface hydrologic modelling: An examination of effects and solutions." Journal of Surveying Engineering, 124, 134 - 143.

21. Clocksin, W. F., and Mellish, C. S. (1989). Programming in Prolog. 3rd Ed., Narosa Publishing House, New Delhi.

22. Crane, R. (1997). A simplified approach to image processing: Classical and modern tecniques in c. Prentice-Hall PTR, New Jersey.

23. Curran, P. J. (1988). Principles of remote sensing. English Language Book Society/Longman, Essex.

24. Dangermond, J., and Schutzberg, A. (July, 1998). "Engineering, geographic information system and database: A new frontier." Journal of Computing in Civil Engineering, 12, 121 - 122.

25. Delbo, S., Gamba, P., and Roccato, D. (May, 2000). "A fuzzy clustering approach to recognize hyperbolic signatures in subsurface radar images." IEEE Transactions on Geoscience and Remote Sensing, 38(3), 1447 - 1451.

26. Devijver, P. A., and Kittler, J. (1982). Pattern recognition: A statistical approach. Prentice-Hall PTR, New Jersey.

27. Dobson, M. C., Pierce, E. L., and Ulaby, F. T. (January, 1996). "Knowledge-based land-cover classification using ERS-1/JERS-1 SAR composites." IEEE Transactions on Geoscience and Remote Sensing, 34(1), 83 - 99.

28. Duda, R. O., and Heart, P. E. (1973). Pattern classification and scene analysis. John Wiley and Sons, New York.

29. Eppler, W. (January, 1976). "Canonocal analysis for increased classification and channel selection." IEEE Transactions on Geoscience Electronics, 14(1), 26 - 33.

30. Friedman, J. H., Baskett, F., and Shustek, L. J. (October, 1975). "An algorithm for finding nearest neighbors." IEEE Transactions on Computers, 24(10), 1000 - 1006.

31. Fu, K. S. (January, 1976). "Pattern recognition in remote sensing of earth's resources." IEEE Transactions on Geoscience Electronics, GE-14(1), 10 - 18.

32. Fukuda, S., and Hirosawa, H. (September, 1999). "A wavelet-based texture feature set applied to classification of multifrequency polarimetric SAR images." IEEE Transactions on Geoscience and Remote Sensing, 37(5), 2282 - 2286.

33. Gath, I., and Geva, A. B. (July, 1989). "Unsupervised optimal fuzzy clustering." IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7), 773 - 781.

34. Goldberg, M., and Scilien, S. (February, 1978). "A clustering scheme for multi-spectral images." IEEE Transactions on Systems, Man and Cybernetics, SMC-8(2), 86 - 92.

35. Gonzalez, R. C., and Woods, R. E. (1993). Digital image processing. Pearson Education (Singapore) Pte. Ltd., Delhi.

36. Goodenough, D. J., Goldberg, M., Plunkett, G., and Zelek, J. (March, 1987). "An expert system for remote sensing." IEEE Transactions on Geoscience and Remote Sensing, 25(3), 349 - 359.

37. Gorte, B., and Stein, A. (May, 1998). "Bayesian classification and class area estimation of satellite images using stratification." IEEE Transactions on Geoscience and Remote Sensing, 36(3), 803 - 812.

38. Graybill, F. A. (1969). Introduction to matrices with applications in statistics. Wadsworth, California.

39. Gupta, R. (February, 2002). "Network flow model using temporal GIS." Proceedings of 5th Annual International Conference, Map India 2002, New Delhi, India, 63 - 66.

40. Gupta, R., and Rohil, M. K. (June, 2002). "Rehabilitation of water-distribution network using GIS." Journal of Indian Building Congress, 9(1), 216 - 223.

41. Gupta, R., Bhatt, M. K., and Rohil, M. K. (June, 2000). "Digital imaging in teaching construction process." Journal of Indian Building Congress, 7(1), 47 - 53.

42. Hair, J. F. J., Anderson, R. E., Tatham, R. L., and Black, W. C. (2003). Multivariate data analysis. 5th Ed., Pearson Education (Singapore) Pte. Ltd., Delhi.

43. Haralick, R. M., Shanmugam, K., and Dinstein, I. (October, 1973). "Textural features for image classification." IEEE Transactions on Systems, Man and Cybernetics, 3, 610 - 621.

44. Harsanyi, J. C., and Chang, C. I. (July, 1994). "Hyper spectral image classification and dimensionality reduction: An orthogonal subspace projection approach." IEEE Transactions on Geoscience and Remote Sensing, 32(4), 779 - 785.

45. Hassan, A. K. (July, 1998). "Data acquisition through emerging high-resolution satellite imagery." Journal of Computing in Civil Engineering, 12, 126 - 128.

46. Heywood, I., Cornelius, S., and Carver, S. (2002). An introduction to geographical information systems. 2nd Ed., Pearson Education (Singapore) Pte. Ltd., Delhi.

47. Hodson, M. E. (1998). "Reducing the computational requirements of the minimum-distance classifier." Remote Sensing of Environment, 25, 117 - 128.

48. Holdstock, D. A. (January, 1998). "Basics of geographic information system (GIS)." Journal of Computing in Civil Engineering, 12, 1 - 4.

49. Hord, R. M. (1982). Digital image processing of remotely sensed data. Academic Press, New York.

50. Jain, A. K. (1995). Fundamentals of image processing. Prentice-Hall of India, New Delhi.

51. Jensen, J. R. (1986). Introductory digital image processing: A remote sensing perspective. Prentice-Hall PTR, New Jersey.

52. Jensen, L. M. (June, 1990). "Knowledge-based classification of an urban area using texture and context information in Lansat-TM imagery." Photogrammetric Engineering and Remote Sensing, 56(6), 899 - 904.

53. Jensen, J. R. (2000). Remote senssing of the environment: An earth resource perspective. Prentice-Hall PTR, New Jersey.

54. Jia, X. (January, 2000). "IntelliGIS: Tool for representing and reasoning spatial knowledge." Journal of Computing in Civil Engineering, 14, 51 - 59.

55. Jia, X., and Richards, J. A. (March, 2002). "Cluster-space representation for hyperspectral data classification." IEEE Transactions on Geoscience and Remote Sensing, 40:3, 593 - 598.

56. Jia, X., and Richards, J. A. (May, 2003). "Efficient transmission and classification of hyperspectral image data." IEEE Transactions on Geoscience and Remote Sensing, 41:5, 1129 - 1131.

57. Johnson, R. A., and Wichern, D. W. (2002). Applied multivariate statistical analysis. 5th Ed., Pearson Education (Singapore) Pte. Ltd., Delhi.

58. Kaewpijit, S., Moigne, J. L., and Ghazawi, T. E. (April, 2003). "Automatic reduction of hyperspectral imagery using wavelet spectral analysis." IEEE Transactions on Geoscience and Remote Sensing, 41(4), 863 - 871.

59. Kartikeyan, B., Majumdar, K. L., and Dasgupta, A. R. (January, 1995). "An expert system for land cover classification." IEEE Transactions on Geoscience and Remote Sensing, 33(1), 58 - 65.

60. Kaufman, Y. J., and Remer, L. (May, 1994). "Detection of forests using Mid-IR reflectance: an application for aerosol system." IEEE Transactions on Geoscience and Remote Sensing, 32(3), 672 - 683.

61. Khazenie, N., and Crawford, M. M. (July, 1990). "Sptial-temporal autocorrelated model for contextual classification." IEEE Transactions on Geoscience and Remote Sensing, 28(4), 529 - 539.

62. Lee, D., and Landgrebe, D. A. (July, 1991). "Fast likelihood classification." IEEE Transactions on Geoscience and Remote Sensing, 29(4), 509 - 517.

63. Lillesand, T. M., and Kiefer, R. W. (2000). Remote sensing and image interpretation. 4th Ed., John Wiley and Sons, New York.

64. MacEachren, A. M. (1995). How maps work: Representation, visualization, and design. The Guilford Press, New York.

65. Marceau, M. J., Howrath, P. J., Dubois, J. M. M., and Gratton, D. J. (July, 1990). "Evaluation of the grey-level co-occurence matrix method for land cover classification using SPOT imagery." IEEE Transactions on Geoscience and Remote Sensing, 28(4), 513 - 519.

66. Martin, J. (1976). Engineering reliability techniques. Open University Press, London.

67. Mather, P. M. (1987). Computer processing of remotely sensed images. John Wiley and Sons, New York.

68. Maulik, U., and Bandyopadhyay, S. (May, 2003). "Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification ." IEEE Transactions on Geoscience and Remote Sensing, 41(5), 1075 - 1081.

69. McKeown, M. D. (May, 1987). "The role of artificial intelligence in the integration of remotely sensed data with geographic information systems." IEEE Transactions on Geoscience and Remote Sensing, GE-25(3), 330 - 348.

70. Mchldau, G., and Schiwengerdt, R. A. (June, 1990). "A C-extension for rule-based image classification systems." Photogrammetric Engineering and Remote Sensing, 56(6), 887 - 892.

71. Melgani, F., Hashemy, A. R. A., and Taha, S. M. R. (January, 2000). "An explicit fuzzy supervised classification method for multi-spectral remote sensing images." IEEE Transactions on Geoscience and Remote Sensing, 38(1), 287 - 295.

72. Middlekoop, H., and Jensen, L. L. F. (July, 1991). "Implementation of temporal relationships in knowledge-based classification of satellite images." Photogrammetric Engineering and Remote Sensing, 57(7), 937 - 945.

73. Moore, T. S., Campbell, J. W., and Feng, H. (August, 2001). "A fuzzy logic classification scheme for selecting and blending satellite ocean color algorithms." IEEE Transactions on Geoscience and Remote Sensing, 39(8), 1764 - 1776.

74. Morrison, D. F. (1976). Multivariate statistical methods. McGraw-Hill Book Company Inc., New York.

75. Mowle, D. P., and Dennetur, C. J. (June, 1991). "The Landsat-6 satellite: an overview." IEEE ASE Systems Magazine, 18 - 23.

76. Narendra, P. M., and Goldberg, M. (1977). "A non-parametric clustering scheme for Landsat." Pattern Recognition, 9. 207 - 215.

77. Nishii, R. (October, 2003). "A Markov random field-based approach to decision-level fusion for remote sensing image classification." IEEE Transactions on Geoscience and Remote Sensing, 41(10), 2316 - 2319.

78. Paole, J. D., and Schowegedr, R. A. (July, 1995). "A detailed comparison of back propagation neural network and ML classifiers for urban land use classification." IEEE Transactions on Geoscience and Remote Sensing, 33(4), 981 - 996.

79. Patterson, D. W. (1990). Introduction to artificial intelligence and expert systems. Prentice-Hall of India, New Delhi.

80. Pipes, L. A. (1958). Applied mathematics for engineers and physicists. 2nd Ed., McGraw-Hill Book Company Inc., New York.

81. Rohil, M. K. (April, 2000). "Natural language processing to query a geographic information system (India) knowledgebase." Proceedings of 3rd Annual International Conference, Map India 2000, New Delhi, India, GTT 4 - GTT 5.

82. Rohil, M. K. (November, 2000). "Exploring possible applications of fuzzy logic in simulation of three dimensional visualization using remote sensing data." Proceedings of Annual Symposium of Indian Society of Remote Sensing, IIT Kanpur, Kanpur, India, 125 - 134.

83. Rohil, M. K. (November, 2000). "An artificial intelligence based GIS for optimal routing for accessibility in rural regions." Proceedings of International Seminar on Accessibility & Rural Development Planning, B.I.T.S., Pilani, India.

84. Saitwal, K., Sadjadi, M. R. A., and Reinke, D. (May, 2003). "A multichannel temporally adaptive system for continuous cloud classification from satellite imagery." IEEE Transactions on Geoscience and Remote Sensing, 41(5), 1098 - 1105.

85. Schalkoff, R. J. (1989). Digital image processing and computer vision. John Wiley and Sons, New York.

86. Schnek, T., and Zilberstein, O. (June, 1990). "Experiments with a rule-based system for interpreting linear map features." Photogrammetric Engineering and Remote Sensing, 56(6), 911 - 917.

87. Serpico, S. B., and Bruzzone, L. (July, 2001). "A new search algorithm for feature selection in hyperspectral remote sensing images." IEEE Transactions on Geoscience and Remote Sensing, 39(7), 1360 - 1367.

88. Serpico, S. B., and Roli, F. (May, 1995). "Classification of multisensor remote-sensing images by structural networks." IEEE Transactions on Geoscience and Remote Sensing, 33(3), 562 - 578.

89. Sethi, I. K. (January, 1981). "A fast algorithm for recognizing nearest neighbours." IEEE Transactions on Systems, Man and Cybernetics, 11(1), 245 - 248.

90. Shackelford, A. K., and Davis, C. H. (September, 2003). "A hierarchical fuzzy classification approach for high-resolution multispectral data over urban areas." IEEE Transactions on Geoscience and Remote Sensing, 41(9), 1920 - 1932.

91. Shalfield, R., Spenser, C., Steel, B. D., and Westwood, A. (2003). LPA Win-Prolog 4.320 user guide. Logic Programming Associates Ltd., London.

92. Siegal, B. S., and Gillespire, A. R. (1980). Remote sensing in geology. John Wiley and Sons, New York.

93. Simhadri, K. K., Iyengar, S. S., Holyer, R. J., Lybanon, M., and Zachary, J. M. (May, 1998). "Wavelet-based feature extraction from oceanographic images." IEEE Transactions on Geoscience and Remote Sensing, 36(3), 767 - 788.

94. Solberg, A. H. S., Taxt, T., and Jain, A. K. (January, 1996). "A Markov random field model for classification of multisource satellite imagery." IEEE Transactions on Geoscience and Remote Sensing, 34(1), 100 - 113.

95. Sonka, M., Hlavac, V., and Boyle, R. (2001). Image processing, analysis and machine vision. 2nd Ed., Thomson Learning Inc., Singapore.

96. Steel, B. D. (2000). LPA Win-Prolog Win32 programming guide. Logic Programming Associates Ltd., London.

97. Sterling, L., and Shapiro, E. (1996). The art of Prolog - Advanced programming techniques. Prentice-Hall of India, New Delhi.

98. Stromberg, W. D., and Farr, T. G. (September, 1986). "A fourier based textual feature extraction procedure." IEEE Transactions on Geoscience and Remote Sensing, 24, 722 - 731.

99. Swain, P. H., and Davis, S. M. (1978). Remote sensing: A quantitative approach. John Wiley and Sons, New York.

100. Szekielda, K. H. (1988). Satellite monitoring of the earth. John Wiley and Sons, New York.

101. Tomlin, C. D. (1990). Geographic information systems and cartographic modeling. Prentice-Hall PTR, New Jersey.

102. Tou, J., and Gonzales, R. (1974). Pattern recognition principles. Addison-Wesley Publishing Company, Massachusetts.

103. Treash, K., and Amaratunga, K. (January, 2000). "Automatic road detection in grayscale aerial images." Journal of Computing in Civil Engineering, 14, 60 - 69.

104. Tu, T. M., Chen, C. H., Wu, J. L., and Chang, C. I. (January, 1998). "A fast two-stage classification method for high-dimensional remote sensing data." IEEE Transactions on Geoscience and Remote Sensing, 36(1), 182 - 191.

105. Venkateswarlu, N. B., and Raju, P. S. V. S. K. (November, 1991). "Three stage ML classifier." Pattern Recognition, 24(11), 1113 - 1116.

106. Wang, F. (March, 1990). "Fuzzy supervised classification of remote sensing images." IEEE Transactions on Geoscience and Remote Sensing, 28(2), 194 - 201.

107. Watanable, S. (1969). Methodologies of pattern recognition. Academic Press, New York.

108. Windham, M. P. (April, 1982). "Cluster validity for the fuzzy C-means clustering algorithms." IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-4(4), 357 - 363.

109. Yoshida, T., and Omatu, S. (September, 1994). "Neural network approach to land cover mapping." IEEE Transactions on Geoscience and Remote Sensing, 32(5), 1103 - 1109.

110. Zaki, F. W., Konyaly, S. H., Fattah. A. I. A., and Enab, Y. M. (July, 1988). "An ensemble average classifier for pattern recognition machines." Pattern Recognition, 21(4), 327 - 332.

111. Zheng, P. Q., and Baetz, B. W. (December, 1999). "GIS-based analysis of development options from a hydrology perspecative." Journal of Urban Planning and Development, 125, 165 - 180.

APPENDIX A *Using Fuzzy Logic in WinProlog*

This appendix gives only a brief overview of the terminology of fuzzy logic and the structure of fuzzy logic program in WinProlog [Steel 2000, Shalfield 2003, http://www.lpa.co.uk/ind_pro.htm]. Here only the relevant functions (predicates) are outlined.

A1. Fuzzy Logic Terminology

There are two main components to a fuzzy logic program: fuzzy variables and fuzzy rules. The keywords in the following text are shown in bold. **Fuzzy variables** take a range of numeric values. The value of a fuzzy variable is referred to as a 'crisp' value. The range can be divided into several sub-ranges by defining qualifiers for the fuzzy variable. Fuzzy variable qualifiers consist of a name, known as a **linguistic qualifier**, and a **membership function,** which shows for each possible 'crisp' value of the fuzzy variable its **degree of membership** of the **fuzzy set**.

When a **'crisp' value** is assigned to a fuzzy variable this is converted into a degree of membership for each linguistic qualifier using its membership function. When a 'crisp' value is required back from the fuzzy variable, so that it can be used outside the fuzzy logic system, a **de-fuzzifying expression** is used. This expression is based on the degrees of membership of the fuzzy variable's qualifiers. A linguistic 'hedge' can affect the degrees of membership of a fuzzy variable qualifier. This has the effect of either concentrating or diluting the fuzziness of the qualifier.

A **Fuzzy Rule** is a rule that refers to one or more fuzzy variable qualifiers in its conditions and a single fuzzy variable qualifier in its conclusion. Rules are applied to fuzzy variables by a process called **propagation**. When a rule is applied it looks at the degrees of membership for the fuzzy variable qualifiers mentioned in its conditions and calculates the new degree of membership for the fuzzy variable qualifier mentioned in its conclusion. The calculation depends upon whether the conditions are formed by conjunctions (and), disjunctions (or), negations (not) or a mixture of these. Fuzzy rules can also be joined together into a **fuzzy matrix**, also known as a **fuzzy associative memory**.

A2. The Structure of a Fuzzy Logic Program

A fuzzy logic program can be viewed as a three-stage process:

- Stage 1 - Fuzzification, the 'crisp' input values are assigned to the appropriate input fuzzy variables. The 'crisp' input value is converted into a degree of membership for each of the qualifiers for the fuzzy variable it is assigned to.
- Stage 2 - Propagation, fuzzy rules are applied to the fuzzy variables and their qualifiers. When a fuzzy rule is applied to some fuzzy variables the degrees of membership for the qualifiers mentioned in the conditions of the rule are propagated to the qualifiers mentioned in the conclusions of the rule.
- Stage 3 - De-fuzzification, the resultant degrees of membership for the qualifiers of the output fuzzy variables are converted back into 'crisp' values.

The following diagram shows this three-stage process for an example fuzzy steam turbine program. Before entering the fuzzy logic program the 'crisp' values for the 'temperature' and 'pressure' of the turbine are found to be 200°C and 15Kpa respectively.

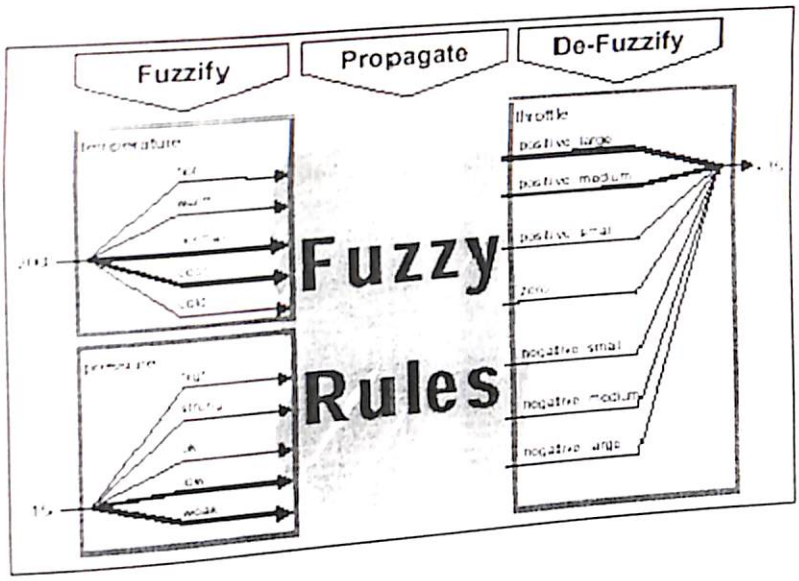


Fig. A1: Stages of a Fuzzy Logic Program

In the first stage of the fuzzy logic program these values are fuzzified into degrees of membership for the qualifiers of the 'temperature' and 'pressure' fuzzy variables. The 'temperature' qualifier with the highest degree of membership is 'normal' followed by 'cool'. The 'pressure' qualifier with the highest degree of membership is 'weak' followed by 'low'.

A3. Fuzzy Rules

The second stage is the fuzzy rule propagation this stage generates, using the fuzzy rules and the degrees of membership of the input fuzzy variable qualifiers, a resultant degree of membership for each of the qualifiers of the 'throttle' fuzzy variable. The 'throttle' qualifier with the highest degree of membership is 'positive_large' followed by
'positive_medium'.

The final stage is to take all the degrees of membership for the qualifiers of the 'throttle' fuzzy variable and calculate a 'crisp' output value. This value turns out to be 35. After exiting the fuzzy logic program with the 'crisp' value this can then be applied back in some way to the throttle for the steam turbine.

A4. Fuzzy Components

This section tells you how to define the components that make up a fuzzy logic program. The components fall into four groups: fuzzy variables, linguistic 'hedges', fuzzy rules and fuzzy matrices.

A4.1 Defining Fuzzy Variables

Fuzzy variables and their qualifiers are the basis for fuzzy logic programs. You can seed the input values for fuzzy variables prior to applying the fuzzy rules or retrieve values from them at the end of the process. To define a fuzzy variable you define its name, its range, all its qualifiers and a de-fuzzifying expression.

The Anatomy Of A Fuzzy Variable

The following diagrams contrast Prolog syntax fuzzy variable definitions. The parts in bold indicate keywords and syntax recognised by the fuzzy interpreter. The plain text indicates the names and parameters that are chosen by the programmer. The qualifiers for the variable are highlighted with a dotted border.

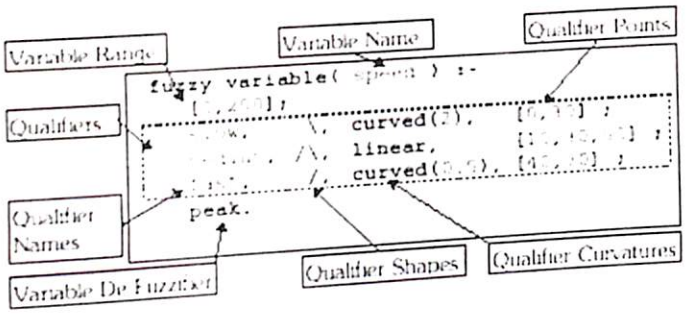


Fig. A2: Parts of a Prolog Fuzzy Variable

The Fuzzy Variable Name

The name of a fuzzy variable is an atom, usually descriptive of the quantity being defined, to be used whenever referring to the variable. An example of a fuzzy variable name could be 'temperature'. A fuzzy variable program that started would define this:

fuzzy_variable(temperature) :-

Setting a Fuzzy Variable's Lower and Upper Bounds

The lower and upper bounds of a fuzzy variable define the range of possible values that the variable can take. For example the fuzzy 'temperature' variable could have a lower bound of -100 and an upper bound of 150. This would be defined by a fuzzy variable program that started as follows:

fuzzy_variable(temperature) :- [-100,150] :

The explicit setting of lower and upper bounds for a fuzzy variable is optional. If this section is left out the fuzzy logic system works out the lower and upper bounds for the fuzzy variable from the shapes of its qualifier membership functions. For example our fuzzy 'temperature' variable could have two qualifiers, 'hot' and 'cold'. The 'cold' qualifier could specify that everything below -20 is definitely 'cold' and that everything above 10 is definitely not 'cold'. The 'hot' qualifier could specify that everything above 80 is definitely 'hot' and that everything below 10 is definitely not 'hot'. In this case, if the upper and lower bounds for the 'temperature' variable were not specified, the fuzzy logic system would assign the value -20 to the lower bound and 80 to the upper bound.

Defining Fuzzy Variable Qualifiers

Fuzzy variable qualifiers consist of a name, known as a 'linguistic qualifier', and a membership function, which shows for each possible 'crisp' value of the fuzzy variable its degree of membership of the set referred to by the qualifier.

Its overall shape, its curvature and some relevant points define the membership function. This could be represented by a graph where the *Y-axis* shows the degree of membership of the set and the *X-axis* the 'crisp' value of the fuzzy variable. The degree of membership is a value somewhere between 0 and 1. A degree of membership of 1 specifies that a value is definitely a member of the set. A degree of membership of 0 specifies that a value is definitely not a member of the set. Numbers in-between define in fuzzy terms varying degrees of membership of a set.

There are seven different overall shapes that can be used: an upwards slope, a downwards slope, an upwards triangle, a downwards triangle, an upwards trapezoid, a downwards trapezoid and a freehand shape. The following table shows the symbols and points needed to define each shape.

Table A1 - The Qualifier Membership Function Shapes

Symbol	Points	Description
\	[A,B]	A downward slope
/	[A,B]	An upward slope
/\	[A,B,C]	An upward pointing triangle
\/	[A,B,C]	A downward pointing triangle
/-\	[A,B,C,D]	An upward pointing trapezoid
\-/	[A,B,C,D]	A downward pointing trapezoid
?	[V1 /M1 ,V2 /M2,... Vk /Mk]	A freehand shape

Defining Fuzzy Variable De-Fuzzifiers

A de-fuzzifier is used to convert a fuzzy variable's current qualifier membership values into a single 'crisp' value for the variable as a whole. This is done when you need to get a value back from the variable. There are two built-in defuzzifiers, centroid, and peak. User-defined expressions may also be used.

The default de-fuzzifier is the **centroid** method. This works by finding the 'centre of gravity' of the collection of all the degrees of membership for all the qualifiers of the fuzzy variable. To illustrate the centroid method, let's take a fuzzy variable called 'rainfall' that has three qualifiers: 'light', 'medium' and 'heavy'. At the end of our fuzzy logic session these qualifiers have the following values: 'light' 0.75, 'medium' 0.25 and 'heavy' 0. The three qualifiers light, heavy and medium are shown in the following diagram:

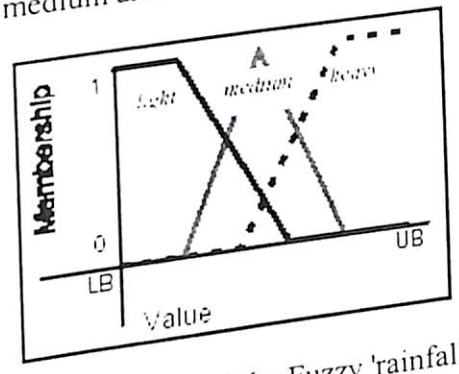


Fig. A3 - The Qualifiers of the Fuzzy 'rainfall' Variable

The centroid method then takes the membership function for each qualifier and cuts off any portion of the function, which contains values above the current value for the qualifier. So the 'light' qualifier in our example would have a cut-off point at 0.75. This is shown in the following diagram:

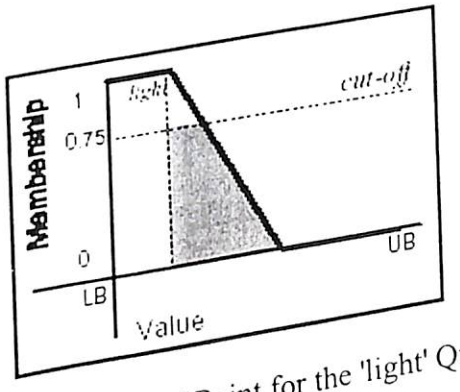


Fig. A4 - The Cut-off Point for the 'light' Qualifier

The shaded portion of the diagram is the shape that will be considered when calculating the centre of gravity. Note that twice both the 'light' and the 'medium' qualifiers have shaded part of the diagram. This portion of the shape is considered to be twice as dense as the rest and this is also taken into account when calculating

the centre of gravity. The centre of gravity indicates a single value on the x-axis, which is deemed to be the resultant 'crisp' value for the fuzzy variable.

The Peak Method

Another built-in de-fuzzier is the **peak** method. This involves finding the qualifier or qualifiers with the highest degree of membership and then calculating the mid-point of all the plateaux occurring at that cut-off point. To illustrate the peak method, let's take our fuzzy 'rainfall' variable again. This time we look for the qualifier, which currently has the highest membership value, which in this case is the 'light' qualifier. The membership function for this qualifier is then cropped at its current membership value and the mid-point of the resultant plateau is calculated. This is shown in the following diagram:

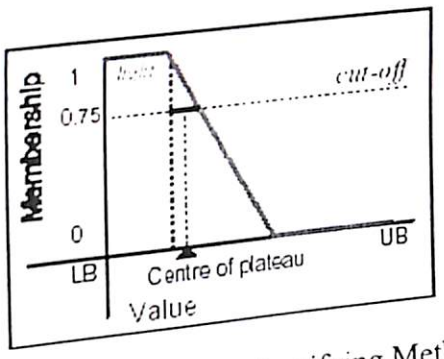


Fig. A5 - The Peak De-fuzzifying Method

The final method for de-fuzzifying a fuzzy variable allows you to define your own expression for calculating the resultant value. Usually this expression takes the current values for the fuzzy variables qualifiers into account. The following shows a de-fuzzifying expression that calculates the 'crisp' value by taking the sum of 10 times the 'light' qualifier, 20 times the 'medium' qualifier and 30 times the 'heavy' qualifier divided by the sum of the 'light', 'medium' and 'heavy' qualifiers.

$$\frac{30 * \text{light} + 20 * \text{medium} + 30 * \text{heavy}}{\text{light} + \text{medium} + \text{heavy}}.$$

A4.2 Defining Linguistic Hedges

A linguistic 'hedge' is used to either concentrate or dilute the characteristic of the membership function for a fuzzy variable qualifier. Hedges like 'extremely' or 'very' normally intensify the effect of the qualifier whilst their counterparts like 'slightly' or 'mildly' dilute the effect of the qualifier.

The Anatomy of a Linguistic Hedge

To define a linguistic 'hedge' you define its name and the formula to be used when the 'hedge' is applied. The Prolog anatomy of a linguistic 'hedge' definition is shown in the following diagram. The parts in bold indicate keywords and syntax recognized by the fuzzy interpreter. The plain text indicates the names and parameters that are chosen by the programmer.

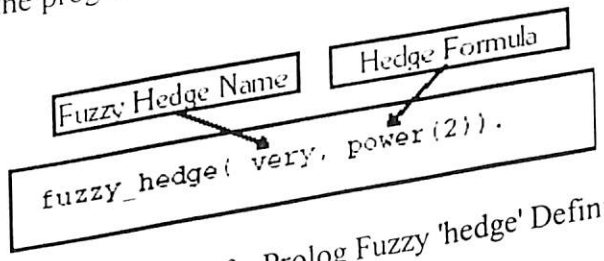


Fig. A6 - The Parts of a Prolog Fuzzy 'hedge' Definition

The Linguistic Hedge Name

The name of a linguistic 'hedge' is an atom to be used whenever the 'hedge' is applied. An example of a fuzzy 'hedge' could be 'slightly'. The following program could define this 'hedge':

```
fuzzy_hedge( slightly , power(0.5)).
```

Note that the 'slightly' hedge would then be 'universal' in that it could be applied to any qualifier.

The Linguistic Hedge Formula

This part of a linguistic 'hedge' program defines the formula to be applied whenever the 'hedge' is used. Hedges apply to the qualifiers of fuzzy variables and affect their membership functions. Currently the only formula supported is that of 'power'. The value of the power must be between 0.1 and 9.9.

A4.3 Defining Fuzzy Rules

Another major component of a fuzzy logic system is the set of rules that reason about the qualifiers of fuzzy variables. Rules consist of a set of 'if' conditions and one 'then' conclusion and an optional 'else' conclusion.

The Anatomy of a Fuzzy Rule

The anatomy of a Prolog fuzzy rule definition is shown in the following diagram. The parts in bold indicate keywords and syntax recognised by the fuzzy logic toolkit. The plain text indicates the names and parameters that are chosen by the programmer. The conditions for the rule are highlighted with a dotted border.

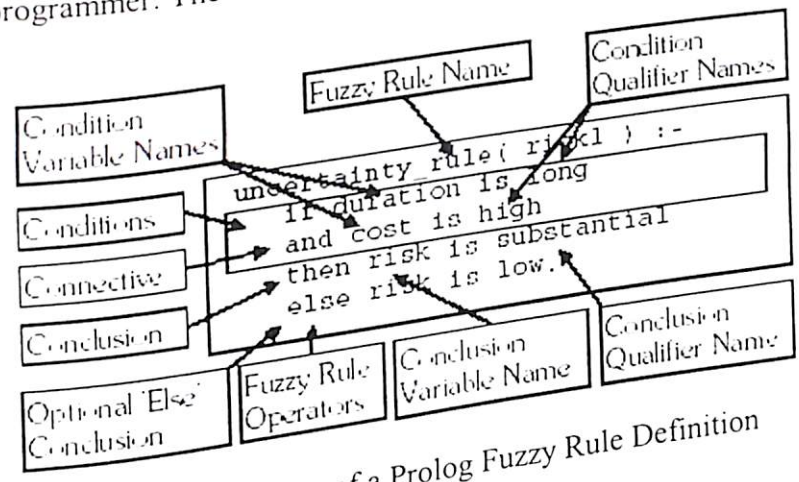


Fig. A7 - The Parts of a Prolog Fuzzy Rule Definition

The Fuzzy Rule Operators

The above diagram, and all of the fuzzy rule examples that follow, make use of certain operators. Any fuzzy logic program that implements rules according to the examples should include the following operator declarations:

```
:-  op( 1150,  fy, ( if )   ),
    op( 1150, xfy, ( then )),
    op( 1150, xfx, ( else )),
    op( 1100, xfy, ( or )   ),
    op( 1000, xfy, ( and )  ),
    op(  700, xfx, ( is )   ),
    op(  600,  fy, ( not )  ).
```

The Fuzzy Rule Name

The name of a fuzzy rule is an atom to be used whenever referring to the rule. An example of a fuzzy variable rule could be 'pressure_rule1'.

The Fuzzy Rule Conditions

The conditions of a fuzzy rule refer to fuzzy variables and their qualifiers. Multiple conditions are joined together using the connectives 'and', 'or' and 'not'. Depending on the type of connective applied to the conditions, the conclusion of the rule is calculated in different ways. The various methods of handling conjunctions, disjunctions and negations are dealt with in more detail in the description of the *fuzzy_propagate/4* predicate.

The Fuzzy Rule 'Then' Conclusion

The value of the qualifier for the fuzzy variable mentioned in the 'then' part of a fuzzy rule is calculated using the current degrees of membership values for the fuzzy variable qualifiers mentioned in the conditions.

The Fuzzy Rule 'Else' Conclusion

The value of the qualifier for the fuzzy variable mentioned in the 'else' part of a fuzzy rule is found by taking 1 minus the value calculated for the fuzzy variable qualifier mentioned in the 'then' part of the fuzzy rule. This section of a fuzzy rule is optional.

A4.4 Defining Fuzzy Matrices

The three rules shown in the previous section have the same structure (i.e. they all have conditions that refer only to qualifiers of the 'temperature' and 'pressure' fuzzy variables and draw a conclusion for a qualifier of the 'throttle' fuzzy variable). Also the conditions are all joined by conjunctions. When this is the case they and all other rules of the same form in the program, can be combined together for convenience using a fuzzy rule matrix. This creates what is also known as a fuzzy associative memory.

The Anatomy of a Fuzzy Rule Matrix

To define a fuzzy rule matrix you specify the fuzzy variables to be used in all the lines of the matrix. Then you specify the qualifiers you are going to use with those variables. The anatomy of a Prolog fuzzy matrix definition is shown in the following diagram. The parts in bold indicate keywords and syntax recognised by the fuzzy logic interpreter. The plain text indicates the names and parameters that are chosen by the programmer. The variables of the matrix are highlighted with a dotted border.

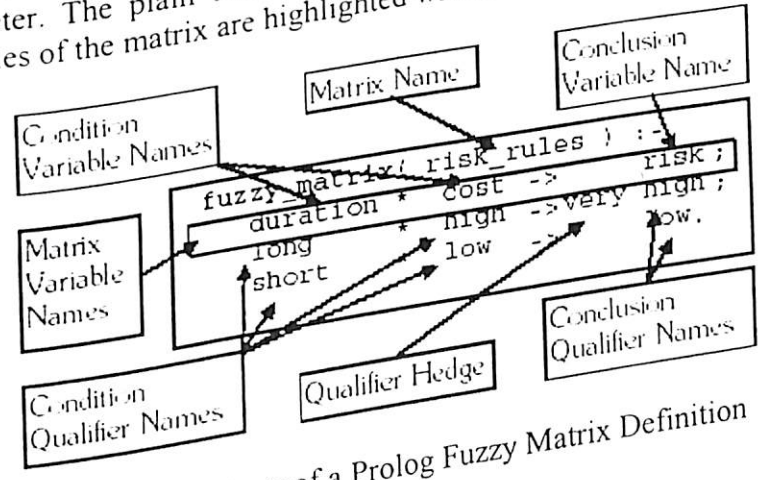


Fig. A8 - The Parts of a Prolog Fuzzy Matrix Definition

The Fuzzy Rule Matrix Name

The name of a fuzzy rule matrix is an atom to be used whenever referring to the matrix. An example of a fuzzy matrix could be 'throttle_rules'. A fuzzy matrix program that started would define this:

```
fuzzy_matrix( throttle_rules ) :-
```

The Fuzzy Rule Matrix Variable Names

The first line in the body of the definition defines the fuzzy variables that will be used in all the rules of the matrix. The following example shows a fuzzy rule matrix that will apply to the fuzzy variables 'temperature', 'pressure' and ' throttle':

```
temperature * pressure -> throttle ;
```

This represents rules of the form:

```
if temperature is _
and pressure is _
then throttle is _
```

The underscores show that the qualifiers are not specified at this stage.

The Fuzzy Rule Matrix Qualifiers

All the remaining lines, following the first fuzzy variable declaration line in the body of the matrix definition, should contain qualifiers for each of the fuzzy variables in the same relative positions. Continuing on from the previous example, the following example shows a rule that applies to the fuzzy variables 'temperature', 'pressure' and ' throttle':

```
cold * weak -> positive_large;
```

This represents the rule:

```
if temperature is cold
and pressure is weak
then throttle is positive_large.
```

A5. Fuzzy Predicates

uncertainty_dynamics/0

initialise the fuzzy sub-system by removing all fuzzy variables and rules

Comments: The *uncertainty_dynamics/0* predicate is used to reset an initial state by removing all the fuzzy variables and rules that have been created.

Examples: The following goal removes all the fuzzy rules and variables:

```
?- uncertainty_dynamics.
Yes
```

uncertainty_listing/0

lists all dynamic predicates representing fuzzy variables and rules

Comments: The *uncertainty_listing/0* predicate is used to show the code for all the currently defined fuzzy variables and rules.

Examples: This example assumes that the following fuzzy variable definition has been compiled:

```
fuzzy_variable( temperature ) :-[
0 , 500 ] ;
cold, \ , linear, [ 110 , 165 ] ;
cool, /\, linear, [ 110 , 165 , 220 ] ;
normal, /\, linear, [ 165 , 220 , 275 ] ;
warm, /\, linear, [ 220 , 275 , 330 ] ;
hot, /, linear, [ 275 , 330 ] .
```

A call to *uncertainty_listing/0* shows the current state of the fuzzy sub-system, showing the fuzzy facts that support the definition:

```
?- uncertainty_listing.

/* fuzzy_variable/1 */
fuzzy_variable(temperature) .
/* fuzzy_variable_range/3 */
fuzzy_variable_range(temperature,0,500) .
/* fuzzy_variable_defuzzifier/2 */
fuzzy_variable_defuzzifier(temperature,A) :-
fuzzy_variable_centroid(temperature,A) .
/* fuzzy_variable_qualifier/5 */.

fuzzy_variable_qualifier(temperature,cold,\,linear,[110,165]) .
fuzzy_variable_qualifier(temperature,cool,/\,linear,[110,165,220]) .
fuzzy_variable_qualifier(temperature,normal,/\,linear,[165,220,275]) .
fuzzy_variable_qualifier(temperature,warm,/\,linear,[220,275,330]) .
fuzzy_variable_qualifier(temperature,hot,/,linear,[275,330]) .
yes
```

fuzzy_reset_membership/0

reset the degrees of membership of all fuzzy variable qualifiers to zero

Comments: The *fuzzy_reset_membership/0* predicate is used to reset an initial state by setting the degrees of membership for all the currently
defined qualifiers attached to all the currently defined fuzzy variables to zero.

Examples: Regardless of the number of fuzzy variables currently defined, the following call will reset all of the degrees of membership for all of their qualifiers to zero:

```
?- fuzzy_reset_membership.
```

fuzzy_reset_membership/1

set the degrees of membership for a fuzzy variable's qualifiers to zero

Syntax: fuzzy_reset_membership(*Variable*) +*Variable* <atom>

Comments: The *fuzzy_reset_membership/1* predicate is used to reset the initial state for a named fuzzy variable by setting the degrees of membership for all its qualifiers to zero.

Examples: The following call resets all of the degrees of membership for all of the qualifiers of the 'temperature' fuzzy variable to zero:

```
?- fuzzy_reset_membership( temperature).
```

fuzzy_variable_value/2

get or set the value for a fuzzy variable

Syntax: fuzzy_variable_value(*Variable*, *Value*) +*Variable* <atom> ?*Value* <number>

Comments: The *fuzzy_variable_value/2* predicate is used to assign or retrieve a value for a named fuzzy variable. If the *Value* argument is 'crisp' then it is converted into degrees of membership for all the fuzzy variable's. qualifiers using each qualifier's membership function, specified when the qualifier was defined (*Defining Fuzzy Variable Qualifiers*).

If the *Value* argument is an unbound variable, the de-fuzzification expression for the fuzzy variable (see *Defining Fuzzy Variable De-Fuzzifiers*) is used to produce a single 'crisp' value which is returned in the *Value* argument. Note that because of the difference between the fuzzification and de-fuzzification processes, you are not guaranteed to get back the same 'crisp' value you put in.

Examples: The following example sets the value for the fuzzy 'temperature' variable to 15.5. This results in various degrees of membership being assigned to its qualifiers:

```
?- fuzzy_variable_value( temperature, 15.5 ).
```

The following example returns a value for the fuzzy 'temperature' variable. This is done using the 'temperature' variable's de-fuzzification expression:

```
?- fuzzy_variable_value( temperature, TempValue ).
```

fuzzy_propagate/1

propagate the degrees of membership values of fuzzy variable qualifiers using the specified rule agenda

Syntax: *fuzzy_propagate(Agenda)* +*Agenda* list of <atom>

Comments: The predicate *fuzzy_propagate/1* is used to apply a list of fuzzy rules. whose names are given in the *Agenda* argument, to the current fuzzy variable state by propagating changes to the membership values for the fuzzy variable qualifiers. This is done using the default methods for handling conjunctions, disjunctions and negations in the fuzzy rules. The *Agenda* argument may also be the name of a fuzzy rule matrix.

This predicate is defined by the program:

```
fuzzy_propagate(Agenda) :-
fuzzy_propagate(minimum, maximum, complement, Agenda).
```

Examples: The following call to *fuzzy_propagate/1* takes the fuzzy rules 't1', 't2' and 't3' and applies them to the current fuzzy variable state. The propagation is done using the 'minimum' method for handling conjunctions, the 'maximum' method for handling disjunctions and the 'complement' method for handling negations (see *fuzzy_propagate*/4):

```
?- fuzzy_propagate([t1,t2,t3]).
```

fuzzy_propagate/4

propagate the degrees of membership values of fuzzy variable qualifiers using the specified combinators and rule agenda

Syntax: *fuzzy_propagate(Conjunction, Disjunction, Negation, Agenda)*
+*Conjunction* <atom> in the domain {minimum,product,truncate}
+*Disjunction* <atom> in the domain {maximum,addition,strengthen}
+*Negation* <atom> in the domain {complement}
+*Agenda* list of <atom>

Comments: The predicate *fuzzy_propagate/4* is used to apply a list of fuzzy rules, whose names are given in the *Agenda* argument, to the current fuzzy variable state by propagating changes to the membership values for the fuzzy variable qualifiers. This is done using the specified methods for handling conjunctions, disjunctions and negations in the fuzzy rules. The *Agenda* argument may also be the name of a fuzzy rule matrix.

The following table shows all the different methods available for calculating membership values, where P and Q are expressions and XP is the degree of membership of P and XQ is the degree of membership of Q.

Table A2 - Methods for Calculating Membership Values

Expression	Method	Description
P and Q	minimum	min(X P ,X Q)
	product	X P * X Q
	truncate	max((X P + X Q -1), 0)
P or Q	maximum	max(X P ,X Q)
	strengthen	X P + X Q * (1-X P)
	addition	min(X P +X Q ,1)
not P	complement	1-X P

For example, given the fuzzy rule:

```
uncertainty_rule( throttle1 ) :-if
temperature is cold
and pressure is weak
then throttle is positive_large .
```

Let's assume the degree of membership for the 'cold' qualifier of the 'temperature' fuzzy variable is 0.5 and the degree of membership for the 'weak' qualifier of the 'pressure' fuzzy variable is 0.25.

The resultant degree of membership for 'positive_large' of 'throttle' using the 'minimum' method is:

```
min(0.5,0.25)
```

which gives the value 0.25. In contrast, the resultant degree of membership for 'positive_large' of 'throttle' using the 'product' method is:

```
0.5 * 0.25
```

which this time gives the value 0.125. Finally the resultant degree of membership for 'positive_large' of 'throttle' using the 'truncate' method is:

```
max((0.5 + 0.25 - 1),0)
```

which gives the value 0.

Examples: The following call to *fuzzy_propagate/4* takes the fuzzy rules 't1', 't2' and 't3' and applies them to the current fuzzy variable state. The propagation is done using the 'minimum' method for handling conjunctions, the 'strengthen' method for handling disjunctions and the 'complement' method for handling negations:

```
?- fuzzy_propagate( minimum, strengthen, complement,
[t1,t2,t3] ).
```

The following call to *fuzzy_propagate/4* uses the fuzzy rule matrix 'throttle_rules' and applies them to the current fuzzy variable state. The propagation is done using the 'product' method for handling conjunctions, the 'sum' method for handling disjunctions and the 'complement' method for handling negations:

```
?- fuzzy_propagate( product, sum, complement,
[throttle_rules] ).
```

uncertainty_propagate/2

Propagate uncertainty values via the specified rules.

Syntax: *uncertainty_propagate(Measure, Rules)*

+*Measure* An uncertainty measure <atom>
being one of {probability, odds, certainty_factor, fuzzy}
+*Rules* <list> of named rules or rule matrix

Examples:

```
?- uncertainty_propagate( fuzzy, [ train1, train2, train3, train4, train5 ] ).
<return>
yes
?- uncertainty_propagate( odds, boiler_matrix ). <return>
yes
```

APPENDIX B *Statistics of the Samples of Four Cases*

For B-dimensional data the mean vector (μ) is given by a B-dimensional vector whose element μ_i represents the mean of i^{th} component of all the B-dimensional samples. Hence, format of mean vector is as follows:

$$\mu = [\mu_1, \mu_2, \mu_3, \ldots, \mu_{i-1}, \mu_i, \mu_{i+1}, \ldots, \mu_B]$$

For B-dimensional data the covariance matrix (Σ) of size $B \times B$ has its general element $\Sigma_{i,j}$, which shows covariance between i^{th} and j^{th} componets of all the B-dimensional samples whenever $i \neq j$, and otherwise it shows variance in the i^{th} components of all the B-dimensional samaples. Hence, format of covariance (and corelation) matrix is as follows:

	$Band_1$	$Band_2$	.....	$Band_B$
$Band_1$	$\Sigma_{1,1}$	$\Sigma_{1,2}$	.....	$\Sigma_{1,B}$
$Band_2$	$\Sigma_{2,1}$	$\Sigma_{2,2}$	.....	$\Sigma_{2,B}$
.	.	.	.....	.
.	.	.	.....	.
$Band_B$	$\Sigma_{B,1}$	$\Sigma_{B,2}$	.....	$\Sigma_{B,B}$

B1. Statistics of Case 1

B1.1 Separability Measures of Training Samples

Divergence Matrix

	Crop	Urban	Tree	Sand
Crop	0	3.44	11.77	6.65
Urban	3.44	0	3.07	2.91
Tree	11.77	3.07	0	5.04
Sand	6.65	2.91	5.04	0

Transformed Divergence Matrix

	Crop	Urban	Tree	Sand
Crop	0	34.96	77.04	56.42
Urban	34.96	0	31.85	30.54
Tree	77.04	31.85	0	46.72
Sand	56.42	30.54	46.72	0

JM Distance Matrix

	Crop	Urban	Tree	Sand
Crop	0	0.39	1.15	0.74
Urban	0.39	0	0.28	0.28
Tree	1.15	0.28	0	0.45
Sand	0.74	0.28	0.45	0

B1.2 Separability Measures of Test Samples

Divergence Matrix

	Crop	Urban	Tree	Sand
Crop	0	4.09	13.83	6.68
Urban	4.09	0	3.93	3.08
Tree	13.83	3.93	0	4.32
Sand	6.68	3.08	4.32	0

Transformed Divergence Matrix

	Crop	Urban	Tree	Sand
Crop	0	40.06	82.25	56.60
Urban	40.06	0	38.81	31.98
Tree	82.25	38.81	0	41.70
Sand	56.60	31.98	41.70	0

JM Distance Matrix

	Crop	Urban	Tree	Sand
Crop	0	0.44	1.21	0.74
Urban	0.44	0	0.32	0.31
Tree	1.21	0.32	0	0.42
Sand	0.74	0.31	0.42	0

B1.3 Mean Vector, Covariance, and Corelation of Samples of Class No. 1 (Crop)

All Pixels

Mean Vector

120.60	121.46	118.04	208.20

Covariance Matrix

159.59	226.75	-0.02	244.41
226.75	339.00	-6.34	366.46
-0.02	-6.34	75.03	1.20
244.41	366.46	1.20	518.52

Corelation Matrix

1	0.97	-0.00	0.85
0.97	1	-0.04	0.87
-0.00	-0.04	1	0.01
0.85	0.87	0.01	1

Training Pixels

Mean Vector

120.50	121.31	118.10	208.16

Covariance Matrix

158.30	227.23	-0.80	242.57
227.23	343.66	-7.24	368.46
-0.80	-7.24	74.15	1.44
242.57	368.46	1.44	517.40

Corelation Matrix

1	0.97	-0.01	0.85
0.97	1	-0.05	0.87
-0.01	-0.05	1	0.01
0.85	0.87	0.01	1

Test Pixels

Mean Vector

120.79	121.75	117.92	208.26

Covariance Matrix

162.63	226.42	1.58	248.90
226.42	330.60	-4.51	363.60
1.58	-4.51	77.01	0.74
248.90	363.60	0.74	522.44

Corelation Matrix

1	0.98	0.01	0.85
0.98	1	-0.03	0.87
0.01	-0.03	1	0.00
0.85	0.87	0.00	1

B1.4 Mean Vector, Covariance, and Corelation of Samples of Class No. 2 (Urban)

All Pixels

Mean Vector

106.86	98.79	109.55	175.79

Covariance Matrix

285.54	375.38	62.57	394.38
375.38	532.61	72.13	577.38
62.57	72.13	131.45	97.76
394.38	577.38	97.76	851.89

Corelation Matrix

1	0.96	0.32	0.80
0.96	1	0.27	0.86
0.32	0.27	1	0.29
0.80	0.86	0.29	1

Training Pixels

Mean Vector

106.90	98.79	109.57	175.83

Covariance Matrix

274.53	366.06	57.62	390.96
366.06	524.25	68.48	574.70
57.62	68.48	127.14	96.46
390.96	574.70	96.46	851.01

Corelation Matrix

1	0.96	0.31	0.81
0.96	1	0.27	0.86
0.31	0.27	1	0.29
0.81	0.86	0.29	1

Test Pixels

Mean Vector

106.78	98.79	109.51	175.71

Covariance Matrix

308.00	394.60	72.58	401.81
394.60	550.17	79.55	583.61
72.58	79.55	140.27	100.51
401.81	583.61	100.51	854.94

Corelation Matrix

1	0.96	0.35	0.78
0.96	1	0.29	0.85
0.35	0.29	1	0.29
0.78	0.85	0.29	1

B1.5 Mean Vector, Covariance, and Corelation of Samples of Class No. 3 (Tree)

All Pixels

Mean Vector

104.08	95.15	102.07	168.01

Covariance Matrix

91.71	112.69	-8.52	70.78
112.69	162.19	-18.82	103.72
-8.52	-18.82	105.64	-11.73
70.78	103.72	-11.73	143.59

Corelation Matrix

1	0.92	-0.09	0.62
0.92	1	-0.14	0.68
-0.09	-0.14	1	-0.10
0.62	0.68	-0.10	1

Training Pixels

Mean Vector

104.04	95.06	101.91	167.81

Covariance Matrix

93.66	116.80	-3.03	76.38
116.80	169.70	-10.66	112.50
-3.03	-10.66	98.89	-5.28
76.38	112.50	-5.28	150.14

Corelation Matrix

1	0.93	-0.03	0.64
0.93	1	-0.08	0.70
-0.03	-0.08	1	-0.04
0.64	0.70	-0.04	1

Test Pixels

Mean Vector

104.18	95.34	102.39	168.42

Covariance Matrix

88.11	104.79	-19.62	59.72
104.79	147.61	-35.35	86.33
-19.62	-35.35	119.38	-24.90
59.72	86.33	-24.90	130.67

Corelation Matrix

1	0.92	-0.19	0.56
0.92	1	-0.27	0.62
-0.19	-0.27	1	-0.20
0.56	0.62	-0.20	1

B1.6 Mean Vector, Covariance, and Corelation of Samples of Class No. 4 (Sand)

All Pixels

Mean Vector

98.40	87.32	120.26	159.48

Covariance Matrix

247.16	356.26	-160.87	327.13
356.26	545.82	-262.87	512.10
-160.87	-262.87	310.74	-254.60
327.13	512.10	-254.60	676.41

Corelation Matrix

1	0.97	-0.58	0.80
0.97	1	-0.64	0.84
-0.58	-0.64	1	-0.56
0.80	0.84	-0.56	1

Training Pixels

Mean Vector

98.12	86.99	120.19	159.28

Covariance Matrix

250.96	363.02	-158.48	326.39
363.02	557.48	-260.53	514.99
-158.48	-260.53	314.74	-252.67
326.39	514.99	-252.67	679.62

Corelation Matrix

1	0.97	-0.56	0.79
0.97	1	-0.62	0.84
-0.56	-0.62	1	-0.55
0.79	0.84	-0.55	1

Test Pixels

Mean Vector

98.95	87.99	120.41	159.89

Covariance Matrix

239.95	343.44	-166.34	329.44
343.44	523.72	-268.62	507.71
-166.34	-268.62	303.79	-259.45
329.44	507.71	-259.45	672.15

Corelation Matrix

1	0.97	-0.62	0.82
0.97	1	-0.67	0.86
-0.62	-0.67	1	-0.57
0.82	0.86	-0.57	1

B2. Statistics of Case 2

B2.1 Separability Measures of Training Samples

Divergence Matrix

	Gram	Wheat	Urban	Sand
Gram	0	3.13	6.60	12.26
Wheat	3.13	0	2.12	4.85
Urban	6.60	2.12	0	2.44
Sand	12.26	4.85	2.44	0

Transformed Divergence Matrix

	Gram	Wheat	Urban	Sand
Gram	0	32.41	56.19	78.40
Wheat	32.41	0	23.30	45.49
Urban	56.19	23.30	0	26.32
Sand	78.40	45.49	26.32	0

JM Distance Matrix

	Gram	Wheat	Urban	Sand
Gram	0	0.36	0.64	1.30
Wheat	0.36	0	0.22	0.50
Urban	0.64	0.22	0	0.26
Sand	1.30	0.50	0.26	0

B2.2 Separability Measures of Training Samples

Divergence Matrix

	Gram	Wheat	Urban	Sand
Gram	0	3.01	7.58	12.28
Wheat	3.01	0	3.71	5.88
Urban	7.58	3.71	0	2.59
Sand	12.28	5.88	2.59	0

Transformed Divergence Matrix

	Gram	Wheat	Urban	Sand
Gram	0	31.34	61.22	78.46
Wheat	31.34	0	37.10	52.08
Urban	61.22	37.10	0	27.66
Sand	78.46	52.08	27.66	0

JM Distance Matrix

	Gram	Wheat	Urban	Sand
Gram	0	0.35	0.70	1.30
Wheat	0.35	0	0.34	0.54
Urban	0.70	0.34	0	0.28
Sand	1.30	0.54	0.28	0

B2.3 Mean Vector, Covariance and Corelation of Samples of Class No. 1 (Gram)

All Pixels

Mean Vector

123.63	126.20	119.51	215.48

Covariance Matrix

113.57	157.59	-3.78	140.46
157.59	234.40	-15.49	210.70
-3.78	-15.49	62.02	-18.85
140.46	210.70	-18.85	269.39

Corelation Matrix

1	0.97	-0.05	0.80
0.97	1	-0.13	0.84
-0.05	-0.13	1	-0.15
0.80	0.84	-0.15	1

Training Pixels

Mean Vector

123.48	126.01	119.64	215.40

Covariance Matrix

116.05	162.33	-6.01	143.41
162.33	243.69	-18.25	218.50
-6.01	-18.25	60.18	-22.47
143.41	218.50	-22.47	272.52

Corelation Matrix

1	0.97	-0.07	0.81
0.97	1	-0.15	0.85
-0.07	-0.15	1	-0.18
0.81	0.85	-0.18	1

Test Pixels

Mean Vector

123.93	126.57	119.24	215.63

Covariance Matrix

108.90	148.51	0.80	135.04
148.51	216.46	-9.84	195.79
0.80	-9.84	65.86	-11.58
135.04	195.79	-11.58	264.18

Corelation Matrix

1	0.97	0.01	0.80
0.97	1	-0.08	0.82
0.01	-0.08	1	-0.09
0.80	0.82	-0.09	1

B2.4 Mean Vector, Covariance and Corelation of Samples of Class No. 2 (Wheat)

All Pixels

Mean Vector

116.52	113.79	113.65	196.16

Covariance Matrix

176.97	241.63	-18.01	298.51
241.63	343.19	-27.84	417.44
-18.01	-27.84	59.08	-30.64
298.51	417.44	-30.64	633.00

Corelation Matrix

1	0.98	-0.18	0.89
0.98	1	-0.20	0.90
-0.18	-0.20	1	-0.16
0.89	0.90	-0.16	1

Training Pixels

Mean Vector

116.73	113.78	113.91	196.18

Covariance Matrix

182.34	251.40	-24.42	302.35
251.40	359.54	-35.79	427.95
-24.42	-35.79	63.66	-38.26
302.35	427.95	-38.26	637.04

Corelation Matrix

1	0.98	-0.23	0.89
0.98	1	-0.24	0.89
-0.23	-0.24	1	-0.19
0.89	0.89	-0.19	1

Test Pixels

Mean Vector

116.09	113.81	113.11	196.12

Covariance Matrix

167.48	224.15	-5.63	293.42
224.15	313.38	-12.10	400.02
-5.63	-12.10	49.98	-15.65
293.42	400.02	-15.65	630.46

Corelation Matrix

1	0.98	-0.06	0.90
0.98	1	-0.10	0.90
-0.06	-0.10	1	-0.09
0.90	0.90	-0.09	1

B2.5 Mean Vector, Covariance and Corelation of Samples of Class No. 3 (Urban)

All Pixels

Mean Vector

110.97	103.78	112.65	182.89

Covariance Matrix

234.46	294.43	14.68	271.53
294.43	413.86	-0.89	414.83
14.68	-0.89	128.92	1.92
271.53	414.83	1.92	667.41

Corelation Matrix

1	0.95	0.08	0.69
0.95	1	-0.00	0.79
0.08	-0.00	1	0.01
0.69	0.79	0.01	1

Training Pixels

Mean Vector

110.92	103.67	112.62	182.94

Covariance Matrix

219.61	281.54	8.52	268.90
281.54	401.73	-5.23	412.89
8.52	-5.23	123.10	1.28
268.90	412.89	1.28	668.81

Corelation Matrix

1	0.95	0.05	0.70
0.95	1	-0.02	0.80
0.05	-0.02	1	0.00
0.70	0.80	0.00	1

Test Pixels

Mean Vector

111.07	104.02	112.71	182.79

Covariance Matrix

264.81	320.97	27.05	277.46
320.97	439.11	7.79	419.72
27.05	7.79	140.91	3.21
277.46	419.72	3.21	666.18

Corelation Matrix

1	0.94	0.14	0.66
0.94	1	0.03	0.78
0.14	0.03	1	0.01
0.66	0.78	0.01	1

B2.6 Mean Vector, Covariance and Corelation of Samples of Class No. 4 (Sand)

All Pixels

Mean Vector

98.40	87.32	120.26	159.48

Covariance Matrix

247.16	356.26	-160.87	327.13
356.26	545.82	-262.87	512.10
-160.87	-262.87	310.74	-254.60
327.13	512.10	-254.60	676.41

Corelation Matrix

1	0.97	-0.58	0.80
0.97	1	-0.64	0.84
-0.58	-0.64	1	-0.56-
0.80	0.84	-0.56	1

Training Pixels

Mean Vector

98.12	86.99	120.19	159.28

Covariance Matrix

250.96	363.02	-158.48	326.39
363.02	557.48	-260.53	514.99
-158.48	-260.53	314.74	-252.67
326.39	514.99	-252.67	679.62

Corelation Matrix

1	0.97	-0.56	0.79
0.97	1	-0.62	0.84
-0.56	-0.62	1	-0.55
0.79	0.84	-0.55	1

Test Pixels

Mean Vector

98.95	87.99	120.41	159.89

Covariance Matrix

239.95	343.44	-166.34	329.44
343.44	523.72	-268.62	507.71
-166.34	-268.62	303.79	-259.45
329.44	507.71	-259.45	672.15

Corelation Matrix

1	0.97	-0.62	0.82
0.97	1	-0.67	0.86
-0.62	-0.67	1	-0.57
0.82	0.86	-0.57	1

B3. Statistics of Case 3

B3.1 Separability Measures of Training Samples

Divergence Matrix

	Crop	Tree	Urban
Crop	0	16.21	9.65
Tree	16.21	0	1.90
Urban	9.65	1.90	0

Transformed Divergence Matrix

	Crop	Tree	Urban
Crop	0	86.81	70.06
Tree	86.81	0	21.17
Urban	70.06	21.17	0

JM Distance Matrix

	Crop	Tree	Urban
Crop	0	1.55	1.00
Tree	1.55	0	0.21
Urban	1.00	0.21	0

B3.2 Separability Measures of Test Samples

Divergence Matrix

	Crop	Tree	Urban
Crop	0	14.34	8.78
Tree	14.34	0	2.35
Urban	8.78	2.35	0

Transformed Divergence Matrix

	Crop	Tree	Urban
Crop	0	83.34	66.62
Tree	83.34	0	25.49
Urban	66.62	25.49	0

JM Distance Matrix

	Crop	Tree	Urban
Crop	0	1.39	0.93
Tree	1.39	0	0.24
Urban	0.93	0.24	0

B3.3 Mean Vector, Covariance and corelation of Samples of Class No. 1 (Crop)

All Pixels

Mean Vector

113.37	110.46	119.74	190.99

Covariance Matrix

351.61	525.31	-64.18	696.03
525.31	808.37	-110.89	1078.05
-64.18	-110.89	93.66	-178.27
696.03	1078.05	-178.27	1582.59

Corelation Matrix

1	0.99	-0.35	0.93
0.99	1	-0.40	0.95
-0.35	-0.40	1	-0.46
0.93	0.95	-0.46	1

Training Pixels

Mean Vector

113.46	110.59	119.79	191.34

Covariance Matrix

353.69	528.21	-61.95	699.98
528.21	811.75	-107.44	1083.33
-61.95	-107.44	90.54	-175.04
699.98	1083.33	-175.04	1591.38

Corelation Matrix

1	0.99	-0.35	0.93
0.99	1	-0.40	0.95
-0.35	-0.40	1	-0.46
0.93	0.95	-0.46	1

Test Pixels

Mean Vector

113.19	110.20	119.65	190.28

Covariance Matrix

348.47	521.04	-68.87	690.06
521.04	803.97	-118.18	1070.50
-68.87	-118.18	100.19	-185.38
690.06	1070.50	-185.38	1569.07

Corelation Matrix

1	0.98	-0.37	0.93
0.98	1	-0.42	0.95
-0.37	-0.42	1	-0.47
0.93	0.95	-0.47	1

B3.4 Mean Vector, Covariance and corelation of Samples of Class No. 2 (Tree)

All Pixels

Mean Vector

99.59	89.44	101.35	162.94

Covariance Matrix

82.85	101.15	20.54	99.40
101.15	143.56	25.81	135.64
20.54	25.81	31.65	27.42
99.40	135.64	27.42	199.46

Corelation Matrix

1	0.93	0.40	0.77
0.93	1	0.38	0.80
0.40	0.38	1	0.35
0.77	0.80	0.35	1

Training Pixels

Mean Vector

99.48	89.37	101.22	162.90

Covariance Matrix

83.75	100.70	21.87	95.97
100.70	142.35	27.70	131.14
21.87	27.70	30.29	27.77
95.97	131.14	27.77	197.89

Corelation Matrix

1	0.92	0.43	0.75
0.92	1	0.42	0.78
0.43	0.42	1	0.36
0.75	0.78	0.36	1

Test Pixels

Mean Vector

99.82	89.60	101.59	163.02

Covariance Matrix

81.35	102.46	17.87	106.69
102.46	146.60	22.08	145.26
17.87	22.08	34.45	26.81
106.69	145.26	26.81	203.50

Corelation Matrix

1	0.94	0.34	0.83
0.94	1	0.31	0.84
0.34	0.31	1	0.32
0.83	0.84	0.32	1

B3.5 Mean Vector, Covariance and corelation of Samples of Class No. 3 (Urban)

All Pixels

Mean Vector

105.79	96.03	102.42	167.37

Covariance Matrix

162.83	196.15	45.80	143.80
196.15	267.26	61.09	200.94
45.80	61.09	95.03	53.71
143.80	200.94	53.71	304.32

Corelation Matrix

1	0.94	0.37	0.65
0.94	1	0.38	0.70
0.37	0.38	1	0.32
0.65	0.70	0.32	1

Training Pixels

Mean Vector

105.82	96.09	102.30	167.76

Covariance Matrix

163.83	198.18	53.00	147.84
198.18	270.84	68.68	202.82
53.00	68.68	100.48	58.72
147.84	202.82	58.72	300.43

Corelation Matrix

1	0.94	0.41	0.67
0.94	1	0.42	0.71
0.41	0.42	1	0.34
0.67	0.71	0.34	1

Test Pixels

Mean Vector

105.73	95.90	102.68	166.57

Covariance Matrix

161.34	192.70	31.52	136.07
192.70	260.92	46.09	197.66
31.52	46.09	84.28	44.12
136.07	197.66	44.12	312.16

Corelation Matrix

1	0.94	0.27	0.61
0.94	1	0.31	0.69
0.27	0.31	1	0.27
0.61	0.69	0.27	1

B4. Statistics of Case 4

B4.1 Separability Measures of Training Samples

Divergence Matrix

	Crop	Tree
Crop	0	22.28
Tree	22.28	0

Transformed Divergence Matrix

	Crop	Tree
Crop	0	93.83
Tree	93.83	0

JM Distance Matrix

	Crop	Tree
Crop	0	1.57
Tree	1.57	0

B4.2 Separability Measures of Test Samples

Divergence Matrix

	Crop	Tree
Crop	0	17.45
Tree	17.45	0

Transformed Divergence Matrix

	Crop	Tree
Crop	0	88.71
Tree	88.71	0

JM Distance Matrix

	Crop	Tree
Crop	0	1.44
Tree	1.44	0

B4.3 Mean Vector, Covariance and corelation of Samples of Class No. 1 (Crop)

All Pixels

Mean Vector

113.68	110.93	119.85	191.85

Covariance Matrix

353.40	527.38	-62.56	701.02
527.38	810.18	-108.84	1083.07
-62.56	-108.84	91.52	-177.08
701.02	1083.07	-177.08	1590.57

Corelation Matrix

1	0.99	-0.35	0.94
0.99	1	-0.40	0.95
-0.35	-0.40	1	-0.46
0.94	0.95	-0.46	1

Training Pixels

Mean Vector

113.72	110.99	119.78	192.06

Covariance Matrix

354.65	529.08	-61.44	702.90
529.08	812.08	-106.68	1085.77
-61.44	-106.68	89.70	-175.31
702.90	1085.77	-175.31	1596.13

Corelation Matrix

1	0.99	-0.34	0.93
0.99	1	-0.40	0.95
-0.34	-0.40	1	-0.46
0.93	0.95	-0.46	1

Test Pixels

Mean Vector

113.58	110.80	119.99	191.43

Covariance Matrix

351.96	525.54	-64.97	699.31
525.54	808.80	-113.46	1080.87
-64.97	-113.46	95.42	-181.06
699.31	1080.87	-181.06	1583.98

Corelation Matrix

1	0.99	-0.35	0.94
0.99	1	-0.41	0.95
-0.35	-0.41	1	-0.47
0.94	0.95	-0.47	1

B4.4 Mean Vector, Covariance and corelation of Samples of Class No. 2 (Tree)

All Pixels

Mean Vector

103.32	94.18	101.92	169.63

Covariance Matrix

56.13	63.76	18.95	41.44
63.76	93.11	21.74	58.80
18.95	21.74	39.13	22.06
41.44	58.80	22.06	92.30

Corelation Matrix

1	0.88	0.40	0.58
0.88	1	0.36	0.63
0.40	0.36	1	0.37
0.58	0.63	0.37	1

Training Pixels

Mean Vector

103.20	94.08	101.80	169.62

Covariance Matrix

57.52	64.01	22.07	36.24
64.01	92.85	25.94	52.29
22.07	25.94	38.29	24.36
36.24	52.29	24.36	86.55

Corelation Matrix

1	0.88	0.47	0.51
0.88	1	0.44	0.58
0.47	0.44	1	0.42
0.51	0.58	0.42	1

Test Pixels

Mean Vector

103.54	94.37	102.16	169.63

Covariance Matrix

53.63	63.61	12.73	52.12
63.61	94.20	13.39	72.24
12.73	13.39	40.98	17.60
52.12	72.24	17.60	104.45

Corelation Matrix

1	0.89	0.27	0.70
0.89	1	0.22	0.73
0.27	0.22	1	0.27
0.70	0.73	0.27	1

APPENDIX C

Tabulation of the Results of Execution of 20 Algorithms on the Test Samples (represented in combinations of 1, 2, 3, and 4 bands) of Four Cases

Table C1
Case 1 (Overlapping Classes)

Code of Algorithm	Combinations of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)	Alpha	Beta
ML	[1]	37.94	0.231	1.29	0.58	1.87	na	na
ML	[2]	40	0.257	1.28	0.59	1.87	na	na
ML	[3]	40.96	0.243	1.32	0.55	1.87	na	na
ML	[4]	42.7	0.283	1.32	0.55	1.87	na	na
ML	[1,2]	42.77	0.276	1.82	1.02	2.84	na	na
ML	[1,3]	46.56	0.319	1.84	1.05	2.89	na	na
ML	[1,4]	43.6	0.291	1.84	1.06	2.9	na	na
ML	[2,3]	47.97	0.334	1.79	1.06	2.85	na	na
ML	[2,4]	43.22	0.289	1.79	1.06	2.85	na	na
ML	[3,4]	50.42	0.359	1.83	1.01	2.84	na	na
ML	[1,2,3]	49.26	0.347	2.47	1.64	4.12	na	na
ML	[1,2,4]	43.99	0.291	2.48	1.64	4.13	na	na
ML	[1,3,4]	52.15	0.376	2.47	1.64	4.12	na	na
ML	[2,3,4]	52.03	0.378	2.46	1.66	4.12	na	na
ML	[1,2,3,4]	52.03	0.375	3.24	2.43	5.68	na	na
FM	[1]	39.74	0.24	1.63	2.18	3.82	na	na
FM	[2]	41.93	0.266	1.62	2.31	3.94	na	na
FM	[3]	43.28	0.255	1.63	2.13	3.76	na	na
FM	[4]	45.14	0.294	1.69	2.12	3.81	na	na
FM	[1,2]	43.73	0.277	2.34	3.53	5.88	na	na
FM	[1,3]	51.38	0.354	2.28	3.57	5.86	na	na
FM	[1,4]	44.95	0.288	2.35	3.55	5.91	na	na
FM	[2,3]	50.48	0.342	2.35	3.5	5.86	na	na
FM	[2,4]	45.14	0.291	2.27	3.57	5.85	na	na
FM	[3,4]	53.63	0.375	2.33	3.55	5.89	na	na
FM	[1,2,3]	52.54	0.363	3.23	5.48	8.71	na	na
FM	[1,2,4]	48.42	0.325	3.23	5.48	8.71	na	na
FM	[1,3,4]	54.66	0.385	3.22	5.49	8.71	na	na
FM	[2,3,4]	54.41	0.383	3.16	5.54	8.7	na	na
FM	[1,2,3,4]	56.59	0.411	4.29	8.12	12.41	na	na
FD	[1]	37.94	0.231	2.08	2.13	4.22	na	na
FD	[2]	39.68	0.253	2.07	2.11	4.19	na	na
FD	[3]	43.02	0.257	2.08	2.06	4.15	na	na
FD	[4]	43.28	0.285	2.08	2.05	4.14	na	na
FD	[1,2]	42.96	0.277	3.08	3.66	6.73	na	na
FD	[1,3]	47.91	0.331	3.01	3.68	6.7	na	na
FD	[1,4]	43.99	0.292	3.01	3.74	6.76	na	na
FD	[2,3]	48.36	0.336	3.01	3.7	6.71	na	na
FD	[2,4]	43.73	0.291	2.99	3.75	6.77	na	na
FD	[3,4]	52.09	0.375	4.31	3.67	6.66	na	na
FD	[1,2,3]	49.84	0.352	4.22	5.99	10.29	na	na
FD	[1,2,4]	44.82	0.3	4.32	5.98	10.19	na	na
FD	[1,3,4]	53.05	0.384	4.23	5.98	10.29	na	na
FD	[2,3,4]	52.73	0.383	5.91	5.99	10.21	na	na
FD	[1,2,3,4]	53.12	0.387	8.99	14.9	23.89	na	na
SM	[1]	36.85	0.134	0.19	0.58	0.77	na	na
SM	[2]	39.04	0.152	0.19	0.59	0.78	na	na
SM	[3]	44.12	0.24	0.17	0.59	0.76	na	na
SM	[4]	41.8	0.164	0.17	0.59	0.76	na	na

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)	Alpha	Beta
SM	[1,2]	38.84	0.148	0.26	1.05	1.31	na	na
SM	[1,3]	49.65	0.303	0.48	1.01	1.49	na	na
SM	[1,4]	41.86	0.172	0.35	1.01	1.36	na	na
SM	[2,3]	50.55	0.314	0.4	1.02	1.42	na	na
SM	[2,4]	41.35	0.167	0.34	1.02	1.36	na	na
SM	[3,4]	51.83	0.324	0.49	1.01	1.5	na	na
SM	[1,2,3]	50.35	0.311	0.65	1.65	2.3	na	na
SM	[1,2,4]	41.8	0.173	0.51	1.64	2.15	na	na
SM	[1,3,4]	51.9	0.324	0.76	1.64	2.4	na	na
SM	[2,3,4]	52.15	0.329	0.73	1.63	2.36	na	na
SM	[1,2,3,4]	51.77	0.323	1.04	2.43	3.47	na	na
SF	[1]	39.61	0.147	0.34	2.13	2.47	na	na
SF	[2]	42.12	0.169	0.26	2.18	2.44	na	na
SF	[3]	45.02	0.251	0.34	2.13	2.47	na	na
SF	[4]	42.32	0.159	0.33	2.13	2.46	na	na
SF	[1,2]	41.8	0.162	0.62	3.53	4.16	na	na
SF	[1,3]	52.09	0.332	1	3.51	4.51	na	na
SF	[1,4]	40.77	0.155	0.84	3.55	4.4	na	na
SF	[2,3]	52.41	0.337	0.96	3.55	4.52	na	na
SF	[2,4]	41.22	0.161	0.8	3.55	4.36	na	na
SF	[3,4]	53.05	0.34	1.12	3.53	4.66	na	na
SF	[1,2,3]	53.31	0.348	1.51	5.48	6.99	na	na
SF	[1,2,4]	41.54	0.165	1.26	5.48	6.74	na	na
SF	[1,3,4]	54.41	0.359	1.82	5.47	7.29	na	na
SF	[2,3,4]	54.28	0.357	1.76	5.47	7.23	na	na
SF	[1,2,3,4]	55.56	0.375	2.51	8.06	10.58	na	na
IE	[1]	36.98	0.157	0.15	0.05	0.2	na	1.25
IE	[2]	39.16	0.181	0.15	0.09	0.24	na	1.25
IE	[3]	41.22	0.209	0.16	0.04	0.19	na	1.25
IE	[4]	41.09	0.206	0.26	0.04	0.2	na	1.25
IE	[1,2]	38.84	0.166	0.26	0.07	0.33	na	1.25
IE	[1,3]	54.02	0.359	0.21	0.07	0.33	na	1.25
IE	[1,4]	41.67	0.184	0.26	0.12	0.33	na	1.25
IE	[2,3]	52.73	0.348	0.21	0.07	0.33	na	1.25
IE	[2,4]	40.13	0.164	0.26	0.12	0.33	na	1.25
IE	[3,4]	54.41	0.368	0.32	0.07	0.33	na	1.25
IE	[1,2,3]	53.95	0.358	0.31	0.1	0.42	na	1.25
IE	[1,2,4]	40.77	0.166	0.31	0.16	0.47	na	1.25
IE	[1,3,4]	55.88	0.374	0.31	0.15	0.46	na	1.25
IE	[2,3,4]	54.41	0.358	0.37	0.15	0.46	na	1.25
IE	[1,2,3,4]	55.37	0.366	0.31	0.18	0.55	na	1.25
BF	[1]	18.71	0.005	0.35	0.05	0.4	na	na
BF	[2]	35.5	0.2	0.35	0.06	0.41	na	na
BF	[3]	40.39	0.167	0.31	0.1	0.41	na	na
BF	[4]	29.58	0.136	0.31	0.05	0.36	na	na
BF	[1,2]	33.89	0.18	0.59	0.15	0.74	na	na
BF	[1,3]	24.12	0.016	0.59	0.14	0.73	na	na
BF	[1,4]	21.74	0.008	0.64	0.1	0.74	na	na
BF	[2,3]	40.71	0.264	0.58	0.15	0.73	na	na
BF	[2,4]	39.29	0.246	0.63	0.15	0.78	na	na
BF	[3,4]	33.25	0.169	0.64	0.14	0.78	na	na
BF	[1,2,3]	39.16	0.238	0.91	0.2	1.11	na	na
BF	[1,2,4]	41.16	0.254	0.92	0.19	1.11	na	na
BF	[1,3,4]	22.83	0.01	0.92	0.2	1.12	na	na
BF	[2,3,4]	41.29	0.271	0.92	0.19	1.11	na	na
BF	[1,2,3,4]	45.08	0.298	1.24	0.25	1.49	na	na
MB	[1]	18.71	0.005	0.84	0.1	0.94	1	0.1
MB	[2]	35.24	0.197	0.92	0.06	0.98	1	0.1
MB	[3]	40.19	0.157	0.93	0.06	0.99	1	0.1
MB	[4]	30.42	0.146	0.9	0.05	0.95	1	0.1
MB	[1,2]	33.76	0.178	1.47	0.1	1.57	1	0.1
MB	[1,3]	24.12	0.01	1.43	0.15	1.58	1	0.1
MB	[1,4]	21.74	0.008	1.43	0.15	1.58	1	0.1
MB	[2,3]	40.84	0.261	1.42	0.16	1.58	1	0.1

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)	Alpha	Beta
MB	[2,4]	39.29	0.246	1.44	0.15	1.59	1	0.1
MB	[3,4]	33.5	0.167	1.44	0.15	1.59	1	0.1
MB	[1,2,3]	38.91	0.234	2.12	0.2	2.32	1	0.1
MB	[1,2,4]	38.01	0.224	2.12	0.19	2.31	1	0.1
MB	[1,3,4]	23.47	0.012	2.11	0.19	2.3	1	0.1
MB	[2,3,4]	41.03	0.263	2.08	0.2	2.28	1	0.1
MB	[1,2,3,4]	41.93	0.266	2.9	0.29	3.19	1	0.1
MC	[1]	39.42	0.231	0.85	0.04	0.89	na	na
MC	[2]	41.54	0.248	0.74	0.04	0.78	na	na
MC	[3]	43.47	0.257	0.85	0.04	0.89	na	na
MC	[4]	44.89	0.289	0.74	0.04	0.78	na	na
MC	[1,2]	39.42	0.231	0.82	0.08	0.9	na	na
MC	[1,3]	39.42	0.231	0.83	0.07	0.9	na	na
MC	[1,4]	39.42	0.231	0.82	0.08	0.9	na	na
MC	[2,3]	41.54	0.248	0.83	0.07	0.9	na	na
MC	[2,4]	41.54	0.248	0.88	0.12	1	na	na
MC	[3,4]	43.47	0.257	0.83	0.08	0.91	na	na
MC	[1,2,3]	39.42	0.231	0.87	0.16	1.03	na	na
MC	[1,2,4]	39.42	0.231	0.87	0.15	1.02	na	na
MC	[1,3,4]	39.42	0.231	0.87	0.16	1.03	na	na
MC	[2,3,4]	41.54	0.248	0.87	0.16	1.03	na	na
MC	[1,2,3,4]	39.42	0.231	1.08	0.19	1.27	na	na
MM	[1]	39.42	0.231	0.4	0.04	0.44	na	na
MM	[2]	41.54	0.248	0.41	0.04	0.45	na	na
MM	[3]	43.47	0.257	0.41	0.05	0.46	na	na
MM	[4]	44.89	0.289	0.39	0.04	0.43	na	na
MM	[1,2]	39.42	0.231	0.39	0.07	0.46	na	na
MM	[1,3]	39.42	0.231	0.38	0.08	0.46	na	na
MM	[1,4]	39.42	0.231	0.38	0.08	0.46	na	na
MM	[2,3]	41.54	0.248	0.39	0.07	0.46	na	na
MM	[2,4]	41.54	0.248	0.38	0.08	0.45	na	na
MM	[3,4]	43.47	0.257	0.38	0.07	0.57	na	na
MM	[1,2,3]	39.42	0.231	0.42	0.15	0.57	na	na
MM	[1,2,4]	39.42	0.231	0.42	0.15	0.58	na	na
MM	[1,3,4]	39.42	0.231	0.43	0.15	0.57	na	na
MM	[2,3,4]	41.54	0.248	0.41	0.16	0.69	na	na
MM	[1,2,3,4]	39.42	0.231	0.5	0.19	0.88	na	na
AA	[1]	37.49	0.138	0.26	0.62	0.86	na	na
AA	[2]	39.61	0.157	0.23	0.63	0.83	na	na
AA	[3]	44.57	0.245	0.21	0.62	0.77	na	na
AA	[4]	42.44	0.174	0.1	0.67	1.52	na	na
AA	[1,2]	39.68	0.156	0.4	1.12	1.77	na	na
AA	[1,3]	50.8	0.32	0.64	1.13	1.36	na	na
AA	[1,4]	42.7	0.181	0.23	1.13	1.68	na	na
AA	[2,3]	51.19	0.324	0.55	1.13	1.35	na	na
AA	[2,4]	41.86	0.172	0.22	1.13	1.48	na	na
AA	[3,4]	49.26	0.282	0.36	1.12	2.65	na	na
AA	[1,2,3]	51.64	0.33	0.86	1.79	2.1	na	na
AA	[1,2,4]	42.38	0.178	0.36	1.74	2.34	na	na
AA	[1,3,4]	49.2	0.282	0.55	1.79	2.27	na	na
AA	[2,3,4]	49.32	0.283	0.52	1.75	3.34	na	na
AA	[1,2,3,4]	48.87	0.277	0.73	2.61	0.88	na	na
AI	[1]	37.49	0.138	0.26	0.62	0.86	na	na
AI	[2]	39.61	0.157	0.23	0.63	0.88	na	na
AI	[3]	44.57	0.245	0.21	0.67	0.77	na	na
AI	[4]	42.44	0.174	0.14	0.63	1.58	na	na
AI	[1,2]	40.06	0.16	0.46	1.12	1.88	na	na
AI	[1,3]	50.35	0.324	0.75	1.13	1.69	na	na
AI	[1,4]	41.22	0.172	0.56	1.13	1.72	na	na
AI	[2,3]	51.38	0.334	0.59	1.13	1.58	na	na
AI	[2,4]	40.71	0.165	0.45	1.13	1.72	na	na
AI	[3,4]	51.51	0.324	0.59	1.78	2.88	na	na
AI	[1,2,3]	51.19	0.336	1.1	1.75	2.58	na	na
AI	[1,2,4]	41.22	0.174	0.83				

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
AI	[1,3,4]	52.28	0.344	1.34	1.74	3.08
AI	[2,3,4]	52.99	0.352	1.07	1.78	2.85
AI	[1,2,3,4]	52.09	0.344	1.76	2.62	4.39
IA	[1]	36.27	0.196	1.15	0.63	1.78
IA	[2]	38.2	0.216	1.11	0.63	1.74
IA	[3]	38.46	0.213	1.11	0.67	1.78
IA	[4]	40.84	0.229	0.85	0.63	1.48
IA	[1,2]	41.16	0.25	1.77	1.12	2.89
IA	[1,3]	47.52	0.32	1.52	1.13	2.65
IA	[1,4]	41.22	0.251	1.74	1.13	2.87
IA	[2,3]	47.85	0.322	1.52	1.12	2.64
IA	[2,4]	41.61	0.258	1.7	1.13	2.83
IA	[3,4]	51.13	0.357	1.54	1.12	2.66
IA	[1,2,3]	49.84	0.346	2.17	1.79	3.96
IA	[1,2,4]	42.57	0.268	2.4	1.79	4.2
IA	[1,3,4]	52.48	0.374	2.36	1.78	4.15
IA	[2,3,4]	51.7	0.367	2.34	1.79	4.14
IA	[1,2,3,4]	52.28	0.374	3.13	2.62	5.76
II	[1]	34.34	0.149	0.83	0.62	1.45
II	[2]	36.33	0.172	0.84	0.67	1.51
II	[3]	41.29	0.224	0.42	0.68	1.1
II	[4]	40.19	0.213	0.81	0.62	1.43
II	[1,2]	39.68	0.215	1.46	1.13	2.59
II	[1,3]	46.75	0.315	1.8	1.13	2.93
II	[1,4]	40.77	0.243	1.68	1.13	2.81
II	[2,3]	47.91	0.33	1.77	1.13	2.9
II	[2,4]	40.84	0.244	1.66	1.13	2.79
II	[3,4]	50.55	0.355	1.8	1.74	2.93
II	[1,2,3]	49	0.341	2.48	1.79	4.23
II	[1,2,4]	41.74	0.254	2.36	1.88	4.16
II	[1,3,4]	52.03	0.372	2.53	1.82	4.42
II	[2,3,4]	51.77	0.373	2.53	2.62	4.36
II	[1,2,3,4]	51.77	0.371	3.31		5.94
XX	[1]	40.39	0.152	0.47	2.17	2.64
XX	[2]	41.48	0.168	0.39	2.22	2.61
XX	[3]	44.31	0.243	0.35	2.16	2.51
XX	[4]	43.79	0.17	0.27	2.17	2.44
XX	[1,2]	40.96	0.162	0.83	3.68	4.51
XX	[1,3]	51.96	0.338	1.21	3.64	4.85
XX	[1,4]	41.86	0.16	0.65	3.65	4.3
XX	[2,3]	51.45	0.33	1.16	3.63	4.8
XX	[2,4]	42.44	0.165	0.54	3.7	4.24
XX	[3,4]	53.38	0.338	0.87	3.63	4.5
XX	[1,2,3]	52.6	0.346	1.84	5.64	7.48
XX	[1,2,4]	43.09	0.175	0.97	5.63	6.6
XX	[1,3,4]	54.66	0.354	1.48	5.63	7.11
XX	[2,3,4]	54.47	0.352	1.44	5.64	7.08
XX	[1,2,3,4]	55.24	0.363	2.05	8.28	10.34
XN	[1]	40.39	0.152	0.48	2.16	2.64
XN	[2]	41.48	0.168	0.39	2.21	2.6
XN	[3]	44.31	0.243	0.36	2.16	2.52
XN	[4]	43.79	0.17	0.21	2.22	2.43
XN	[1,2]	41.03	0.168	0.86	3.61	4.48
XN	[1,3]	51.9	0.343	1.49	3.62	5.12
XN	[1,4]	38.84	0.145	1.15	3.64	4.8
XN	[2,3]	51.32	0.334	1.24	3.63	4.88
XN	[2,4]	40.45	0.165	1	3.64	4.64
XN	[3,4]	52.99	0.346	1.3	3.62	4.93
XN	[1,2,3]	52.54	0.35	2.17	5.65	7.82
XN	[1,2,4]	41.03	0.175	1.62	5.65	7.27
XN	[1,3,4]	53.89	0.362	2.46	5.64	8.1
XN	[2,3,4]	54.08	0.361	2.21	5.63	7.84
XN	[1,2,3,4]	55.63	0.385	3.44	8.28	11.73

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
NX	[1]	38.2	0.212	1.56	2.16	3.72
NX	[2]	39.87	0.226	1.43	2.21	3.64
NX	[3]	43.22	0.255	1.55	2.18	3.73
NX	[4]	42.83	0.237	1.24	2.21	3.45
NX	[1,2]	41.86	0.248	2.37	3.61	5.99
NX	[1,3]	51.51	0.353	2.21	3.63	5.85
NX	[1,4]	43.54	0.264	2.37	3.65	6.02
NX	[2,3]	50.68	0.342	2.18	3.63	5.82
NX	[2,4]	43.6	0.265	2.28	3.7	5.98
NX	[3,4]	53.38	0.369	2.22	3.63	5.85
NX	[1,2,3]	52.67	0.363	3.11	5.64	8.75
NX	[1,2,4]	47.07	0.306	3.29	5.64	8.93
NX	[1,3,4]	54.79	0.386	3.27	5.65	8.92
NX	[2,3,4]	54.34	0.381	3.26	5.64	8.9
NX	[1,2,3,4]	56.72	0.412	4.5	8.26	12.76
NN	[1]	36.85	0.172	1.22	2.17	3.39
NN	[2]	38.65	0.195	1.23	2.2	3.43
NN	[3]	38.71	0.199	0.88	2.17	3.05
NN	[4]	42.25	0.224	1.13	2.22	3.35
NN	[1,2]	40.51	0.222	2.15	3.62	5.78
NN	[1,3]	51.45	0.354	2.39	3.64	6.04
NN	[1,4]	42.96	0.255	2.34	3.64	5.98
NN	[2,3]	50.68	0.344	2.38	3.63	6.02
NN	[2,4]	43.34	0.26	2.25	3.63	5.89
NN	[3,4]	53.44	0.372	2.37	3.61	5.99
NN	[1,2,3]	52.6	0.364	3.29	5.63	8.92
NN	[1,2,4]	46.75	0.301	3.26	5.65	8.91
NN	[1,3,4]	54.66	0.385	3.41	5.64	9.05
NN	[2,3,4]	54.41	0.383	3.39	5.64	9.03
NN	[1,2,3,4]	56.59	0.411	4.55	8.25	12.8
RF	[1]	18.71	0.005	0.17	0.02	0.19
RF	[2]	18.78	0.003	0.16	0.03	0.19
RF	[3]	18.07	-0.001	0.16	0.03	0.19
RF	[4]	20.71	0.002	0.17	0.02	0.19
RF	[1,2]	18.84	0.003	0.23	0.04	0.27
RF	[1,3]	18.65	0.005	0.23	0.04	0.27
RF	[1,4]	21.22	0.007	0.27	0.04	0.31
RF	[2,3]	18.71	0.002	0.28	0.04	0.32
RF	[2,4]	21.16	0.006	0.27	0.08	0.31
RF	[3,4]	20.64	0.002	0.23	0.05	0.31
RF	[1,2,3]	18.78	0.002	0.33	0.06	0.38
RF	[1,2,4]	21.22	0.007	0.33	0.05	0.39
RF	[1,3,4]	21.16	0.006	0.34	0.06	0.39
RF	[2,3,4]	21.09	0.005	0.33	0.05	0.39
RF	[1,2,3,4]	21.16	0.006	0.43	0.06	0.5
MR	[1]	18.14	0	0.18	0.07	0.26
MR	[2]	18.14	0	0.18	0.08	0.22
MR	[3]	18.14	0	0.18	0.04	0.22
MR	[4]	18.14	0	0.29	0.04	0.22
MR	[1,2]	18.14	0	0.3	0.04	0.36
MR	[1,3]	18.14	0	0.25	0.07	0.41
MR	[1,4]	18.14	0	0.29	0.11	0.36
MR	[2,3]	18.14	0	0.3	0.12	0.37
MR	[2,4]	18.14	0	0.24	0.11	0.42
MR	[3,4]	18.14	0	0.36	0.15	0.36
MR	[1,2,3]	18.14	0	0.41	0.1	0.51
MR	[1,2,4]	18.14	0	0.42	0.15	0.51
MR	[1,3,4]	18.14	0	0.36	0.15	0.57
MR	[2,3,4]	18.14	0	0.48	0.18	0.51
MR	[1,2,3,4]	18.14	0	0.48	0.18	0.66

Table C2
Case 2 (Overlapping Classes)

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
ML	[1]	45.25	0.276	0.9	0.4	1.3
ML	[2]	48.64	0.311	0.93	0.39	1.32
ML	[3]	41.11	0.242	0.9	0.4	1.3
ML	[4]	49.67	0.323	0.9	0.4	1.3
ML	[1,2]	46.38	0.304	1.18	0.71	1.89
ML	[1,3]	48.17	0.314	1.19	0.71	1.9
ML	[1,4]	51.36	0.355	1.23	0.66	1.89
ML	[2,3]	50.71	0.344	1.23	0.67	1.9
ML	[2,4]	49.29	0.329	1.23	0.67	1.9
ML	[3,4]	52.4	0.363	1.18	0.71	1.89
ML	[1,2,3]	51.36	0.356	1.62	1.08	2.7
ML	[1,2,4]	50.14	0.347	1.62	1.09	2.71
ML	[1,3,4]	52.4	0.362	1.62	1.09	2.71
ML	[2,3,4]	53.15	0.372	1.62	1.09	2.71
ML	[1,2,3,4]	54.28	0.394	2.09	1.6	3.7
FM	[1]	47.41	0.297	1.04	1.38	2.42
FM	[2]	48.92	0.319	0.98	1.43	2.41
FM	[3]	42.33	0.247	0.99	1.38	2.37
FM	[4]	49.11	0.325	0.98	1.38	2.36
FM	[1,2]	52.3	0.363	1.4	2.24	3.64
FM	[1,3]	51.36	0.351	1.4	2.24	3.64
FM	[1,4]	54.56	0.381	1.41	2.24	3.65
FM	[2,3]	51.55	0.348	1.41	2.24	3.65
FM	[2,4]	52.96	0.364	1.41	2.24	3.66
FM	[3,4]	53.9	0.371	1.4	2.24	3.64
FM	[1,2,3]	54.37	0.383	1.88	3.46	5.35
FM	[1,2,4]	56.26	0.406	1.89	3.46	5.36
FM	[1,3,4]	56.73	0.404	1.89	3.41	5.31
FM	[2,3,4]	56.07	0.394	1.89	3.4	5.3
FM	[1,2,3,4]	58.8	0.437	2.48	5.01	7.49
FD	[1]	45.06	0.276	1.13	1.29	2.42
FD	[2]	48.45	0.31	1.13	1.3	2.43
FD	[3]	41.2	0.242	1.13	1.3	2.43
FD	[4]	50.33	0.336	1.14	1.29	2.43
FD	[1,2]	47.22	0.313	1.59	2.17	3.77
FD	[1,3]	48.82	0.324	1.67	2.18	3.86
FD	[1,4]	51.65	0.358	1.6	2.19	3.8
FD	[2,3]	50.71	0.343	1.67	2.19	3.87
FD	[2,4]	51.74	0.358	1.6	2.19	3.79
FD	[3,4]	52.59	0.365	1.67	2.19	3.83
FD	[1,2,3]	51.46	0.357	1.6	2.17	3.65
FD	[1,2,4]	51.36	0.361	1.66	3.37	5.65
FD	[1,3,4]	52.96	0.368	2.27	3.44	5.65
FD	[2,3,4]	53.25	0.374	2.2	3.44	5.65
FD	[1,2,3,4]	54.47	0.396	2.2	3.44	5.65
SM	[1]	40.92	0.229	2.95	5.03	7.98
SM	[2]	41.49	0.244	0.1	0.4	0.5
SM	[3]	41.86	0.227	0.14	0.39	0.53
SM	[4]	45.34	0.293	0.2	0.4	0.6
SM	[1,2]	42.24	0.25	0.08	0.39	0.47
SM	[1,3]	48.82	0.312	0.18	0.66	0.84
SM	[1,4]	48.73	0.325	0.29	0.71	1
SM	[2,3]	50.52	0.334	0.22	0.71	0.93
SM	[2,4]	47.79	0.317	0.33	0.66	0.99
SM	[3,4]	50.8	0.343	0.2	0.67	0.87
SM	[1,2,3]	50.42	0.332	0.29	0.71	1
SM	[1,2,4]	49.11	0.331	0.41	1.09	1.5
SM	[1,3,4]	51.08	0.341	0.33	1.09	1.42
SM	[2,3,4]	51.74	0.351	0.46	1.09	1.55
				0.45	1.09	1.54

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
SM	[1,2,3,4]	52.96	0.365	0.66	1.59	2.25
SF	[1]	46.85	0.294	0.2	1.43	1.63
SF	[2]	48.64	0.319	0.2	1.43	1.63
SF	[3]	41.96	0.23	0.28	1.42	1.7
SF	[4]	51.55	0.362	0.19	1.38	1.57
SF	[1,2]	49.2	0.32	0.36	2.23	2.59
SF	[1,3]	51.18	0.333	0.67	2.24	2.91
SF	[1,4]	52.87	0.356	0.52	2.25	2.77
SF	[2,3]	51.36	0.334	0.64	2.24	2.88
SF	[2,4]	52.49	0.356	0.5	2.25	2.75
SF	[3,4]	53.34	0.358	0.72	2.25	2.97
SF	[1,2,3]	51.83	0.34	0.95	3.43	4.38
SF	[1,2,4]	53.72	0.37	0.74	3.47	4.21
SF	[1,3,4]	55.5	0.379	1.12	3.46	4.59
SF	[2,3,4]	53.9	0.359	1.15	3.46	4.61
SF	[1,2,3,4]	56.44	0.396	1.62	4.96	6.58
IE	[1]	48.64	0.306	0.07	0.07	0.14
IE	[2]	49.39	0.318	0.12	0.03	0.15
IE	[3]	38.76	0.113	0.08	0.07	0.15
IE	[4]	50.61	0.333	0.08	0.03	0.11
IE	[1,2]	49.86	0.323	0.16	0.05	0.21
IE	[1,3]	54.28	0.356	0.16	0.05	0.21
IE	[1,4]	54.37	0.371	0.15	0.06	0.21
IE	[2,3]	54.75	0.367	0.16	0.05	0.21
IE	[2,4]	53.34	0.363	0.11	0.1	0.2
IE	[3,4]	57.01	0.395	0.11	0.09	0.31
IE	[1,2,3]	54.84	0.367	0.19	0.12	0.31
IE	[1,2,4]	54.66	0.376	0.19	0.12	0.27
IE	[1,3,4]	56.73	0.39	0.2	0.07	0.27
IE	[2,3,4]	56.54	0.391	0.2	0.07	0.41
IE	[1,2,3,4]	56.73	0.392	0.27	0.14	0.3
BF	[1]	27.28	0.008	0.22	0.08	0.31
BF	[2]	15.52	0.043	0.22	0.09	0.32
BF	[3]	30.1	0.083	0.23	0.09	0.32
BF	[4]	39.04	0.176	0.27	0.05	0.55
BF	[1,2]	42.24	0.232	0.42	0.13	0.57
BF	[1,3]	27.19	0.008	0.45	0.12	0.55
BF	[1,4]	39.7	0.183	0.43	0.12	0.51
BF	[2,3]	26.25	0.13	0.42	0.09	0.5
BF	[2,4]	44.12	0.29	0.42	0.08	0.51
BF	[3,4]	38.76	0.173	0.43	0.08	0.8
BF	[1,2,3]	42.14	0.231	0.63	0.17	0.79
BF	[1,2,4]	45.91	0.277	0.63	0.16	0.75
BF	[1,3,4]	39.42	0.181	0.59	0.16	0.79
BF	[2,3,4]	44.31	0.29	0.63	0.16	1
BF	[1,2,3,4]	45.81	0.276	0.79	0.21	0.57
MB	[1]	27.56	0.011	0.53	0.04	0.6
MB	[2]	14.77	0.036	0.55	0.05	0.52
MB	[3]	29.07	0.071	0.48	0.04	0.52
MB	[4]	36.78	0.138	0.48	0.04	0.96
MB	[1,2]	42.33	0.234	0.84	0.12	0.88
MB	[1,3]	27	0.006	0.8	0.08	0.96
MB	[1,4]	37.44	0.145	0.84	0.12	0.89
MB	[2,3]	27.56	0.145	0.8	0.09	0.96
MB	[2,4]	37.91	0.223	0.84	0.12	0.88
MB	[3,4]	38.85	0.174	0.8	0.08	1.36
MB	[1,2,3]	42.24	0.233	1.18	0.18	1.27
MB	[1,2,4]	45.81	0.275	1.11	0.16	1.35
MB	[1,3,4]	39.32	0.18	1.19	0.16	1.28
MB	[2,3,4]	38.48	0.227	1.11	0.17	1.66
MB	[1,2,3,4]	45.63	0.273	1.45	0.21	0.41
MC	[1]	47.22	0.293	0.38	0.03	

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
MC	[2]	47.79	0.304	0.39	0.03	0.42
MC	[3]	44.87	0.263	0.39	0.03	0.42
MC	[4]	49.39	0.325	0.39	0.03	0.42
MC	[1,2]	47.22	0.293	0.39	0.05	0.44
MC	[1,3]	47.22	0.293	0.4	0.05	0.45
MC	[1,4]	47.22	0.293	0.33	0.11	0.44
MC	[2,3]	47.79	0.304	0.42	0.1	0.52
MC	[2,4]	47.79	0.304	0.4	0.05	0.45
MC	[3,4]	44.87	0.263	0.39	0.05	0.44
MC	[1,2,3]	47.22	0.293	0.43	0.12	0.55
MC	[1,2,4]	47.22	0.293	0.42	0.13	0.55
MC	[1,3,4]	47.22	0.293	0.42	0.13	0.55
MC	[2,3,4]	47.79	0.304	0.42	0.12	0.54
MC	[1,2,3,4]	47.22	0.293	0.42	0.14	0.56
MM	[1]	47.22	0.293	0.22	0.04	0.26
MM	[2]	47.79	0.304	0.24	0.03	0.27
MM	[3]	44.87	0.263	0.23	0.03	0.26
MM	[4]	49.39	0.325	0.24	0.04	0.28
MM	[1,2]	47.22	0.293	0.23	0.05	0.28
MM	[1,3]	47.22	0.293	0.22	0.05	0.27
MM	[1,4]	47.22	0.293	0.25	0.1	0.35
MM	[2,3]	47.79	0.304	0.22	0.06	0.28
MM	[2,4]	47.79	0.304	0.21	0.06	0.27
MM	[3,4]	44.87	0.263	0.25	0.1	0.35
MM	[1,2,3]	47.22	0.293	0.22	0.08	0.3
MM	[1,2,4]	47.22	0.293	0.24	0.12	0.36
MM	[1,3,4]	47.22	0.293	0.25	0.11	0.36
MM	[2,3,4]	47.79	0.304	0.23	0.07	0.3
MM	[1,2,3,4]	47.22	0.293	0.23	0.15	0.38
AA	[1]	40.92	0.232	0.15	0.43	0.58
AA	[2]	41.49	0.244	0.1	0.46	0.56
AA	[3]	42.24	0.237	0.41	0.42	0.83
AA	[4]	37.91	0.208	0.06	0.43	0.49
AA	[1,2]	42.24	0.25	0.18	0.75	0.93
AA	[1,3]	49.95	0.327	0.38	0.75	1.13
AA	[1,4]	40.73	0.238	0.16	0.75	0.91
AA	[2,3]	50.52	0.334	0.28	0.76	1.04
AA	[2,4]	40.26	0.233	0.11	0.76	0.87
AA	[3,4]	45.34	0.282	0.19	0.76	0.95
AA	[1,2,3]	50.42	0.332	0.46	0.76	1.62
AA	[1,2,4]	41.2	0.244	0.2	1.16	1.36
AA	[1,3,4]	46.75	0.296	0.29	1.16	1.45
AA	[2,3,4]	47.04	0.3	0.24	1.16	1.4
AA	[1,2,3,4]	47.98	0.31	0.35	1.74	2.09
AI	[1]	40.92	0.232	0.15	0.42	0.57
AI	[2]	41.49	0.244	0.13	0.42	0.55
AI	[3]	42.24	0.237	0.41	0.42	0.83
AI	[4]	37.91	0.208	0.06	0.42	0.49
AI	[1,2]	42.99	0.261	0.21	0.43	0.97
AI	[1,3]	50.24	0.329	0.94	0.76	1.7
AI	[1,4]	50.89	0.351	0.29	0.76	1.04
AI	[2,3]	51.27	0.343	0.94	0.75	1.7
AI	[2,4]	48.73	0.326	0.21	0.76	0.97
AI	[3,4]	52.87	0.363	1.01	0.76	1.76
AI	[1,2,3]	52.4	0.361	1.33	0.75	2.54
AI	[1,2,4]	51.08	0.353	0.35	1.21	1.51
AI	[1,3,4]	52.96	0.364	1.48	1.16	2.7
AI	[2,3,4]	53.53	0.374	1.48	1.22	2.64
AI	[1,2,3,4]	55.03	0.401	1.97	1.16	3.7
IA	[1]	47.88	0.299	0.54	1.73	0.97
IA	[2]	48.82	0.312	0.44	0.43	0.86
IA	[3]	38.95	0.212	0.87	0.42	1.29
IA	[4]	49.67	0.324	0.53	0.42	0.96
IA	[1,2]				0.43	

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
IA	[1,2]	46.75	0.297	0.83	0.76	1.59
IA	[1,3]	48.45	0.313	1.17	0.76	1.93
IA	[1,4]	51.65	0.351	1.11	0.76	1.87
IA	[2,3]	51.08	0.343	1.11	0.75	1.86
IA	[2,4]	49.86	0.329	1.03	0.76	1.79
IA	[3,4]	52.87	0.362	1.14	0.75	1.89
IA	[1,2,3]	52.12	0.362	1.57	1.16	2.73
IA	[1,2,4]	50.05	0.341	1.54	1.16	2.7
IA	[1,3,4]	52.4	0.36	1.63	1.21	2.84
IA	[2,3,4]	53.15	0.371	1.63	1.16	2.79
IA	[1,2,3,4]	54.47	0.396	2.11	1.74	3.85
II	[1]	46.1	0.287	0.34	0.42	0.76
II	[2]	46.85	0.302	0.32	0.42	0.74
II	[3]	38.57	0.182	0.45	0.43	0.88
II	[4]	48.17	0.322	0.25	0.43	0.68
II	[1,2]	47.13	0.301	0.6	0.76	1.36
II	[1,3]	49.01	0.318	1.11	0.75	1.86
II	[1,4]	53.15	0.365	0.85	0.76	1.61
II	[2,3]	51.08	0.344	1.12	0.71	1.83
II	[2,4]	50.89	0.338	0.88	0.76	1.64
II	[3,4]	52.87	0.362	1.13	0.75	1.88
II	[1,2,3]	51.93	0.359	1.53	1.2	2.73
II	[1,2,4]	49.95	0.336	1.27	1.21	2.48
II	[1,3,4]	52.4	0.36	1.63	1.16	2.79
II	[2,3,4]	53.15	0.371	1.63	1.16	2.79
II	[1,2,3,4]	54.47	0.396	2.11	1.74	3.86
XX	[1]	46.75	0.289	0.27	1.41	1.68
XX	[2]	48.64	0.319	0.2	1.45	1.65
XX	[3]	41.02	0.191	0.55	1.41	1.96
XX	[4]	49.76	0.343	0.16	1.41	1.57
XX	[1,2]	49.2	0.32	0.35	2.34	2.69
XX	[1,3]	51.83	0.339	0.69	2.35	3.04
XX	[1,4]	53.06	0.368	0.34	2.34	2.68
XX	[2,3]	51.36	0.333	0.58	2.35	2.93
XX	[2,4]	52.02	0.359	0.32	2.3	2.62
XX	[3,4]	53.62	0.369	0.44	2.34	2.78
XX	[1,2,3]	51.83	0.34	0.96	3.52	4.49
XX	[1,2,4]	53.15	0.367	0.5	3.55	4.05
XX	[1,3,4]	54.19	0.37	0.73	3.53	4.27
XX	[2,3,4]	53.81	0.366	0.78	3.52	4.31
XX	[1,2,3,4]	54.84	0.378	1.02	5.11	6.13
XN	[1]	46.75	0.289	0.22	1.4	1.62
XN	[2]	48.64	0.319	0.2	1.4	1.6
XN	[3]	41.02	0.191	0.55	1.4	1.95
XN	[4]	49.76	0.343	0.15	1.41	1.56
XN	[1,2]	49.2	0.32	0.36	2.29	2.65
XN	[1,3]	52.3	0.351	1.26	2.34	3.6
XN	[1,4]	53.15	0.36	0.55	2.35	2.9
XN	[2,3]	51.93	0.342	1.23	2.3	3.53
XN	[2,4]	52.49	0.356	0.5	2.35	2.85
XN	[3,4]	54.19	0.37	1.37	2.33	3.7
XN	[1,2,3]	53.9	0.371	1.76	3.53	5.3
XN	[1,2,4]	53.72	0.369	0.83	3.53	4.37
XN	[1,3,4]	56.44	0.398	1.97	3.53	5.5
XN	[2,3,4]	55.69	0.386	1.89	3.54	5.44
XN	[1,2,3,4]	58.42	0.43	2.57	5.11	7.68
NX	[1]	47.41	0.297	0.77	1.45	2.22
NX	[2]	48.92	0.319	0.74	1.41	2.15
NX	[3]	42.33	0.247	1.08	1.4	2.48
NX	[4]	49.11	0.325	0.76	1.41	2.17
NX	[1,2]	51.93	0.352	1.25	2.34	3.59
NX	[1,3]	51.65	0.351	1.41	2.3	3.71
NX	[1,4]	54.47	0.375	1.41	2.35	3.76

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
NX	[2,3]	51.83	0.346	1.38	2.3	3.68
NX	[2,4]	52.78	0.358	1.39	2.3	3.69
NX	[3,4]	53.81	0.367	1.42	2.34	3.76
NX	[1,2,3]	54.28	0.379	1.9	3.54	5.44
NX	[1,2,4]	55.88	0.399	1.9	3.53	5.44
NX	[1,3,4]	56.44	0.4	2	3.52	5.53
NX	[2,3,4]	55.88	0.39	1.99	3.52	5.52
NX	[1,2,3,4]	58.61	0.434	2.62	5.09	7.71
NN	[1]	47.88	0.296	0.45	1.45	1.9
NN	[2]	49.58	0.321	0.57	1.41	1.98
NN	[3]	38.1	0.175	0.71	1.4	2.11
NN	[4]	49.95	0.328	0.51	1.41	1.92
NN	[1,2]	52.12	0.352	1.03	2.33	3.37
NN	[1,3]	51.65	0.351	1.34	2.36	3.7
NN	[1,4]	54.19	0.369	1.23	2.34	3.57
NN	[2,3]	51.83	0.346	1.39	2.3	3.7
NN	[2,4]	52.78	0.355	1.25	2.36	3.61
NN	[3,4]	53.72	0.366	1.41	2.28	3.69
NN	[1,2,3]	54.28	0.379	1.95	3.52	5.48
NN	[1,2,4]	55.13	0.386	1.77	3.53	5.31
NN	[1,3,4]	56.44	0.4	2.01	3.53	5.54
NN	[2,3,4]	55.88	0.39	1.92	3.54	5.47
NN	[1,2,3,4]	58.61	0.434	2.6	5.1	7.7
RF	[1]	27.38	0.01	0.13	0.01	0.14
RF	[2]	27.47	0.008	0.12	0.02	0.14
RF	[3]	26.53	0	0.08	0.02	0.1
RF	[4]	29.44	0.013	0.12	0.02	0.14
RF	[1,2]	27.56	0.009	0.12	0.07	0.19
RF	[1,3]	27.38	0.01	0.17	0.02	0.19
RF	[1,4]	30.2	0.022	0.17	0.03	0.2
RF	[2,3]	27.47	0.008	0.17	0.02	0.19
RF	[2,4]	30.1	0.021	0.17	0.02	0.19
RF	[3,4]	29.44	0.013	0.16	0.07	0.23
RF	[1,2,3]	27.56	0.009	0.21	0.08	0.29
RF	[1,2,4]	30.2	0.022	0.21	0.03	0.24
RF	[1,3,4]	30.2	0.022	0.25	0.06	0.31
RF	[2,3,4]	30.1	0.021	0.2	0.08	0.28
RF	[1,2,3,4]	30.2	0.022	0.29	0.05	0.34
MR	[1]	26.53	0	0.1	0.02	0.12
MR	[2]	26.53	0	0.13	0.03	0.16
MR	[3]	26.53	0	0.09	0.03	0.12
MR	[4]	26.53	0	0.09	0.03	0.12
MR	[1,2]	26.53	0	0.23	0.05	0.28
MR	[1,3]	26.53	0	0.18	0.09	0.27
MR	[1,4]	26.53	0	0.18	0.09	0.27
MR	[2,3]	26.53	0	0.18	0.05	0.23
MR	[2,4]	26.53	0	0.18	0.05	0.23
MR	[3,4]	26.53	0	0.18	0.05	0.23
MR	[1,2,3]	26.53	0	0.18	0.05	0.23
MR	[1,2,4]	26.53	0	0.18	0.11	0.34
MR	[1,3,4]	26.53	0	0.23	0.12	0.35
MR	[2,3,4]	26.53	0	0.23	0.08	0.35
MR	[1,2,3,4]	26.53	0	0.27	0.07	0.34
				0.27	0.1	0.41
				0.31		

Table C3
Case 3 (Less Separable Classes)

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
ML	[1]	57.82	0.383	0.5	0.34	0.84
ML	[2]	61.3	0.431	0.55	0.29	0.84
ML	[3]	62.22	0.444	0.54	0.3	0.84
ML	[4]	55.85	0.352	0.54	0.34	0.88
ML	[1,2]	63.73	0.463	0.73	0.56	1.29
ML	[1,3]	67.9	0.523	0.74	0.56	1.3
ML	[1,4]	66.4	0.502	0.74	0.56	1.3
ML	[2,3]	68.02	0.525	0.73	0.56	1.29
ML	[2,4]	66.16	0.498	0.74	0.55	1.29
ML	[3,4]	67.9	0.525	0.73	0.56	1.29
ML	[1,2,3]	70.45	0.56	0.95	0.88	1.83
ML	[1,2,4]	67.21	0.512	0.95	0.88	1.83
ML	[1,3,4]	73.23	0.601	0.94	0.88	1.82
ML	[2,3,4]	71.84	0.581	1	0.83	1.83
ML	[1,2,3,4]	73.81	0.609	1.24	1.26	2.5
FM	[1]	58.86	0.39	0.58	1.07	1.65
FM	[2]	60.72	0.414	0.58	1.06	1.64
FM	[3]	65.47	0.481	0.57	1.08	1.65
FM	[4]	55.04	0.332	0.58	1.11	1.69
FM	[1,2]	65.7	0.485	0.78	1.79	2.57
FM	[1,3]	70.68	0.556	0.84	1.76	2.6
FM	[1,4]	67.44	0.508	0.78	1.8	2.58
FM	[2,3]	71.26	0.565	0.84	1.76	2.6
FM	[2,4]	66.28	0.491	0.79	1.8	2.59
FM	[3,4]	69.87	0.545	0.84	1.75	2.59
FM	[1,2,3]	72.42	0.582	1.13	2.69	3.83
FM	[1,2,4]	69.87	0.542	1.12	2.69	3.82
FM	[1,3,4]	76.36	0.641	1.12	2.68	3.81
FM	[2,3,4]	74.86	0.618	1.07	2.73	3.81
FM	[1,2,3,4]	76.71	0.645	1.45	3.89	5.34
FD	[1]	58.63	0.39	0.6	0.93	1.53
FD	[2]	60.49	0.416	0.6	0.92	1.52
FD	[3]	61.88	0.438	0.66	0.88	1.54
FD	[4]	55.62	0.347	0.6	0.93	1.53
FD	[1,2]	64.19	0.469	0.87	1.59	2.46
FD	[1,3]	68.83	0.535	0.87	1.61	2.48
FD	[1,4]	67.32	0.515	0.89	1.59	2.48
FD	[2,3]	69.76	0.549	0.82	1.6	2.42
FD	[2,4]	66.28	0.499	0.88	1.59	2.47
FD	[3,4]	68.13	0.528	0.87	1.59	2.46
FD	[1,2,3]	72.31	0.587	1.18	2.45	3.63
FD	[1,2,4]	67.79	0.52	1.18	2.44	3.62
FD	[1,3,4]	73.58	0.605	1.18	2.44	3.63
FD	[2,3,4]	73.58	0.605	1.18	2.44	3.62
FD	[1,2,3,4]	74.51	0.619	1.52	3.55	5.07
SM	[1]	39.17	0.055	0.13	0.3	0.43
SM	[2]	40.32	0.07	0.1	0.34	0.44
SM	[3]	64.08	0.466	0.07	0.33	0.4
SM	[4]	40.56	0.063	0.07	0.33	0.42
SM	[1,2]	41.37	0.087	0.09	0.56	0.75
SM	[1,3]	70.34	0.553	0.19	0.57	0.81
SM	[1,4]	43.11	0.11	0.24	0.56	0.76
SM	[2,3]	69.99	0.548	0.2	0.56	0.81
SM	[2,4]	43.92	0.123	0.29	0.52	0.76
SM	[3,4]	67.56	0.514	0.2	0.56	0.85
SM	[1,2,3]	71.84	0.575	0.29	0.56	1.28
SM	[1,2,4]	43.8	0.122	0.39	0.89	1.14
SM	[1,3,4]	73.35	0.597	0.26	0.88	1.28
SM	[2,3,4]	72.65	0.587	0.44	0.84	1.29
				0.41	0.88	

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
SM	[1,2,3,4]	73.58	0.6	0.55	1.27	1.82
SF	[1]	48.2	0.228	0.18	1.11	1.29
SF	[2]	51.33	0.278	0.18	1.13	1.31
SF	[3]	66.28	0.49	0.21	1.12	1.33
SF	[4]	51.22	0.28	0.17	1.12	1.29
SF	[1,2]	56.32	0.346	0.24	1.81	2.05
SF	[1,3]	69.87	0.542	0.48	1.8	2.28
SF	[1,4]	65.93	0.486	0.36	1.81	2.17
SF	[2,3]	69.99	0.543	0.44	1.81	2.25
SF	[2,4]	65.12	0.474	0.36	1.8	2.16
SF	[3,4]	69.99	0.545	0.49	1.8	2.29
SF	[1,2,3]	70.8	0.555	0.68	2.69	3.37
SF	[1,2,4]	66.51	0.492	0.54	2.69	3.23
SF	[1,3,4]	74.16	0.605	0.73	2.74	3.47
SF	[2,3,4]	74.04	0.603	0.72	2.74	3.46
SF	[1,2,3,4]	74.62	0.611	0.99	3.91	4.9
IE	[1]	54.35	0.313	0.09	0.02	0.11
IE	[2]	53.53	0.292	0.05	0.02	0.07
IE	[3]	63.04	0.422	0.05	0.02	0.07
IE	[4]	53.65	0.294	0.05	0.06	0.11
IE	[1,2]	54.35	0.305	0.11	0.04	0.15
IE	[1,3]	65.7	0.469	0.11	0.04	0.15
IE	[1,4]	56.55	0.335	0.07	0.08	0.15
IE	[2,3]	65.59	0.463	0.07	0.09	0.16
IE	[2,4]	56.2	0.326	0.07	0.09	0.16
IE	[3,4]	64.77	0.448	0.07	0.04	0.11
IE	[1,2,3]	66.51	0.48	0.07	0.06	0.19
IE	[1,2,4]	56.2	0.326	0.13	0.11	0.2
IE	[1,3,4]	66.16	0.473	0.09	0.06	0.19
IE	[2,3,4]	64.89	0.45	0.13	0.1	0.23
IE	[1,2,3,4]	66.16	0.472	0.13	0.12	0.27
BF	[1]	41.02	0.112	0.15	0.04	0.19
BF	[2]	53.07	0.315	0.16	0.03	0.19
BF	[3]	58.52	0.389	0.16	0.03	0.19
BF	[4]	60.02	0.369	0.15	0.03	0.18
BF	[1,2]	53.19	0.316	0.3	0.06	0.36
BF	[1,3]	56.89	0.377	0.25	0.06	0.31
BF	[1,4]	59.91	0.372	0.25	0.07	0.32
BF	[2,3]	56.32	0.371	0.27	0.11	0.38
BF	[2,4]	60.25	0.416	0.25	0.1	0.35
BF	[3,4]	59.44	0.413	0.25	0.1	0.35
BF	[1,2,3]	56.66	0.375	0.35	0.13	0.48
BF	[1,2,4]	60.37	0.418	0.39	0.1	0.49
BF	[1,3,4]	59.91	0.418	0.39	0.14	0.53
BF	[2,3,4]	58.98	0.406	0.34	0.14	0.48
BF	[1,2,3,4]	59.33	0.411	0.48	0.17	0.65
MB	[1]	42.99	0.145	0.33	0.04	0.37
MB	[2]	49.02	0.251	0.32	0.04	0.36
MB	[3]	57.82	0.378	0.33	0.04	0.37
MB	[4]	59.68	0.363	0.27	0.08	0.35
MB	[1,2]	49.13	0.252	0.45	0.11	0.56
MB	[1,3]	56.32	0.37	0.46	0.1	0.56
MB	[1,4]	59.56	0.367	0.46	0.11	0.57
MB	[2,3]	55.62	0.36	0.45	0.11	0.56
MB	[2,4]	57.24	0.367	0.45	0.11	0.56
MB	[3,4]	59.33	0.411	0.46	0.11	0.56
MB	[1,2,3]	55.97	0.365	0.63	0.1	0.77
MB	[1,2,4]	57.24	0.367	0.63	0.14	0.76
MB	[1,3,4]	59.79	0.416	0.64	0.13	0.77
MB	[2,3,4]	59.56	0.414	0.63	0.13	0.76
MB	[1,2,3,4]	59.79	0.417	0.77	0.16	0.93
MC	[1]	55.74	0.344	0.23	0.03	0.26
MC	[2]	53.3	0.307	0.21	0.02	0.23

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
MC	[3]	62.69	0.448	0.23	0.02	0.25
MC	[4]	54.92	0.331	0.22	0.02	0.24
MC	[1,2]	55.74	0.344	0.23	0.04	0.27
MC	[1,3]	55.74	0.344	0.23	0.08	0.31
MC	[1,4]	55.74	0.344	0.23	0.09	0.32
MC	[2,3]	53.3	0.307	0.21	0.04	0.25
MC	[2,4]	53.3	0.307	0.24	0.08	0.32
MC	[3,4]	62.69	0.448	0.24	0.08	0.32
MC	[1,2,3]	55.74	0.344	0.24	0.1	0.34
MC	[1,2,4]	55.74	0.344	0.23	0.1	0.33
MC	[1,3,4]	55.74	0.344	0.24	0.1	0.34
MC	[2,3,4]	53.3	0.307	0.23	0.11	0.34
MC	[1,2,3,4]	55.74	0.344	0.22	0.12	0.34
MM	[1]	55.74	0.344	0.1	0.02	0.12
MM	[2]	53.3	0.307	0.1	0.03	0.13
MM	[3]	62.69	0.448	0.11	0.02	0.13
MM	[4]	54.92	0.331	0.11	0.06	0.17
MM	[1,2]	55.74	0.344	0.11	0.08	0.19
MM	[1,3]	55.74	0.344	0.1	0.04	0.14
MM	[1,4]	55.74	0.344	0.16	0.05	0.21
MM	[2,3]	53.3	0.307	0.16	0.04	0.2
MM	[2,4]	53.3	0.307	0.11	0.08	0.19
MM	[3,4]	62.69	0.448	0.1	0.09	0.19
MM	[1,2,3]	55.74	0.344	0.16	0.07	0.23
MM	[1,2,4]	55.74	0.344	0.11	0.1	0.21
MM	[1,3,4]	55.74	0.344	0.16	0.07	0.23
MM	[2,3,4]	53.3	0.307	0.17	0.11	0.28
MM	[1,2,3,4]	55.74	0.344	0.11	0.12	0.23
AA	[1]	48.2	0.228	0.29	0.32	0.61
AA	[2]	49.02	0.235	0.25	0.37	0.62
AA	[3]	62.57	0.447	0.05	0.36	0.41
AA	[4]	49.94	0.258	0.26	0.35	0.61
AA	[1,2]	52.84	0.287	0.4	0.6	1
AA	[1,3]	68.37	0.527	0.18	0.6	0.78
AA	[1,4]	60.95	0.41	0.46	0.6	1.06
AA	[2,3]	67.44	0.513	0.18	0.61	0.79
AA	[2,4]	61.41	0.416	0.45	0.64	1.09
AA	[3,4]	65.59	0.488	0.14	0.64	0.78
AA	[1,2,3]	69.18	0.538	0.25	0.94	1.19
AA	[1,2,4]	62.8	0.435	0.68	0.95	1.63
AA	[1,3,4]	71.15	0.567	0.31	0.94	1.25
AA	[2,3,4]	70.34	0.555	0.31	0.94	1.25
AA	[1,2,3,4]	71.73	0.575	0.39	1.39	1.78
AI	[1]	48.2	0.228	0.24	0.36	0.6
AI	[2]	49.02	0.235	0.29	0.32	0.61
AI	[3]	62.57	0.447	0.05	0.36	0.41
AI	[4]	49.94	0.258	0.26	0.35	0.61
AI	[1,2]	54.11	0.307	0.41	0.6	1.01
AI	[1,3]	68.6	0.523	0.65	0.6	1.25
AI	[1,4]	61.18	0.413	0.52	0.6	1.12
AI	[2,3]	66.16	0.487	0.6	0.6	1.2
AI	[2,4]	61.41	0.416	0.46	0.61	1.06
AI	[3,4]	68.48	0.525	0.58	0.94	1.19
AI	[1,2,3]	68.83	0.528	0.87	0.98	1.81
AI	[1,2,4]	63.04	0.439	0.66	0.94	1.64
AI	[1,3,4]	70.57	0.554	0.88	0.99	1.82
AI	[2,3,4]	68.83	0.529	0.82	1.34	1.81
AI	[1,2,3,4]	70.8	0.558	1.17	1.39	2.51
IA	[1]	52.49	0.288	0.36	0.36	0.72
IA	[2]	55.39	0.326	0.33	0.36	0.69
IA	[3]	62.22	0.441	0.54	0.31	0.85
IA	[4]	56.66	0.352	0.42	0.36	0.78
IA	[1,2]	60.02	0.399	0.58	0.64	1.22
IA	[1,3]	67.32	0.509	0.72	0.6	1.32

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
IA	[1,4]	64.54	0.468	0.68	0.61	1.29
IA	[2,3]	66.86	0.505	0.75	0.6	1.35
IA	[2,4]	63.96	0.46	0.66	0.64	1.3
IA	[3,4]	68.13	0.527	0.75	0.61	1.36
IA	[1,2,3]	69.87	0.549	1.01	0.94	1.95
IA	[1,2,4]	65.7	0.486	0.92	0.94	1.86
IA	[1,3,4]	72.31	0.586	0.97	0.94	1.91
IA	[2,3,4]	71.03	0.568	0.97	0.94	1.91
IA	[1,2,3,4]	73.23	0.599	1.31	1.34	2.65
II	[1]	52.49	0.288	0.3	0.36	0.66
II	[2]	55.39	0.326	0.37	0.31	0.68
II	[3]	62.22	0.441	0.54	0.36	0.9
II	[4]	56.66	0.352	0.42	0.36	0.78
II	[1,2]	61.41	0.421	0.61	0.6	1.21
II	[1,3]	67.67	0.519	0.81	0.6	1.41
II	[1,4]	66.28	0.498	0.75	0.6	1.35
II	[2,3]	68.13	0.527	0.82	0.59	1.41
II	[2,4]	65.7	0.489	0.74	0.61	1.35
II	[3,4]	68.02	0.527	0.76	0.6	1.36
II	[1,2,3]	70.45	0.56	1.04	0.94	1.98
II	[1,2,4]	67.21	0.511	1.02	0.94	1.96
II	[1,3,4]	73.23	0.601	1.04	0.94	1.98
II	[2,3,4]	71.96	0.583	0.99	0.98	1.97
II	[1,2,3,4]	73.81	0.609	1.33	1.34	2.67
XX	[1]	57.13	0.358	0.32	1.14	1.46
XX	[2]	58.98	0.385	0.28	1.17	1.45
XX	[3]	65.93	0.488	0.17	1.09	1.26
XX	[4]	55.04	0.332	0.38	1.09	1.47
XX	[1,2]	63.04	0.44	0.55	1.84	2.39
XX	[1,3]	69.29	0.535	0.37	1.84	2.21
XX	[1,4]	65.82	0.481	0.67	1.84	2.51
XX	[2,3]	71.03	0.563	0.38	1.84	2.22
XX	[2,4]	65.24	0.472	0.66	1.83	2.5
XX	[3,4]	68.83	0.531	0.38	2.75	2.21
XX	[1,2,3]	71.38	0.566	0.53	2.75	3.28
XX	[1,2,4]	67.79	0.508	0.95	2.79	3.71
XX	[1,3,4]	73.7	0.6	0.58	2.74	3.37
XX	[2,3,4]	72.77	0.586	0.63	3.97	3.37
XX	[1,2,3,4]	74.51	0.611	0.84	4.03	4.81
XN	[1]	57.13	0.358	0.32	1.09	1.41
XN	[2]	58.98	0.385	0.34	1.14	1.48
XN	[3]	65.93	0.488	0.17	1.14	1.31
XN	[4]	55.04	0.332	0.38	1.14	1.52
XN	[1,2]	62.57	0.523	0.54	1.84	2.38
XN	[1,3]	68.71	0.483	0.77	2.07	2.84
XN	[1,4]	65.93	0.51	0.77	2.26	3.03
XN	[2,3]	68.02	0.472	0.72	1.86	2.58
XN	[2,4]	65.24	0.526	0.66	1.88	2.54
XN	[3,4]	68.83	0.547	0.74	1.84	2.58
XN	[1,2,3]	70.34	0.508	1.05	2.8	3.86
XN	[1,2,4]	67.79	0.605	0.9	2.84	3.75
XN	[1,3,4]	74.16	0.576	1.05	2.75	3.8
XN	[2,3,4]	72.31	0.614	1.02	2.8	3.83
XN	[1,2,3,4]	74.86	0.614	1.37	4.03	5.4
NX	[1]	59.1	0.391	0.43	1.15	1.58
NX	[2]	60.37	0.405	0.51	1.09	1.6
NX	[3]	65.47	0.481	0.62	1.15	1.77
NX	[4]	55.04	0.325	0.55	1.09	1.64
NX	[1,2]	64.77	0.468	0.78	1.8	2.58
NX	[1,3]	70.68	0.554	0.79	1.85	2.64
NX	[1,4]	66.51	0.493	0.78	1.86	2.64
NX	[2,3]	71.15	0.562	0.87	1.84	2.71
NX	[2,4]	65.24	0.473	0.84	1.85	2.69
NX	[3,4]	69.64	0.541	0.87	1.83	2.7

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
NX	[1,2,3]	71.96	0.574	1.09	2.79	3.88
NX	[1,2,4]	69.06	0.529	1.13	2.75	3.88
NX	[1,3,4]	76.25	0.638	1.11	2.8	3.91
NX	[2,3,4]	74.97	0.619	1.16	2.76	3.92
NX	[1,2,3,4]	76.48	0.641	1.43	3.99	5.42
NN	[1]	59.1	0.391	0.41	1.14	1.55
NN	[2]	60.37	0.405	0.45	1.16	1.61
NN	[3]	65.47	0.481	0.67	1.09	1.76
NN	[4]	55.04	0.325	0.54	1.14	1.68
NN	[1,2]	65.35	0.478	0.81	1.85	2.66
NN	[1,3]	70.57	0.554	0.88	1.85	2.73
NN	[1,4]	67.21	0.504	0.87	1.84	2.71
NN	[2,3]	70.92	0.559	0.88	1.84	2.72
NN	[2,4]	66.05	0.486	0.87	1.84	2.71
NN	[3,4]	69.76	0.543	0.88	1.84	2.72
NN	[1,2,3]	72.19	0.579	1.17	2.8	3.98
NN	[1,2,4]	69.87	0.542	1.11	2.79	3.91
NN	[1,3,4]	76.36	0.641	1.18	2.75	3.94
NN	[2,3,4]	74.86	0.618	1.18	2.79	3.98
NN	[1,2,3,4]	76.71	0.645	1.5	3.98	5.48
RF	[1]	36.15	0	0.06	0.01	0.07
RF	[2]	36.15	0	0.09	0.01	0.1
RF	[3]	36.15	0	0.05	0.01	0.06
RF	[4]	47.51	0.175	0.05	0.02	0.07
RF	[1,2]	36.15	0	0.11	0.02	0.13
RF	[1,3]	36.15	0	0.12	0.02	0.14
RF	[1,4]	47.51	0.175	0.11	0.02	0.14
RF	[2,3]	36.15	0	0.12	0.03	0.14
RF	[2,4]	47.51	0.175	0.07	0.02	0.14
RF	[3,4]	47.39	0.173	0.08	0.07	0.14
RF	[1,2,3]	36.15	0	0.1	0.06	0.17
RF	[1,2,4]	47.51	0.175	0.14	0.07	0.17
RF	[1,3,4]	47.39	0.173	0.15	0.03	0.18
RF	[2,3,4]	47.39	0.173	0.14	0.03	0.17
RF	[1,2,3,4]	47.39	0.173	0.16	0.03	0.2
MR	[1]	36.15	0	0.1	0.04	0.12
MR	[2]	36.15	0	0.06	0.02	0.08
MR	[3]	36.15	0	0.06	0.02	0.08
MR	[4]	38.24	0.032	0.05	0.02	0.12
MR	[1,2]	36.15	0	0.13	0.07	0.17
MR	[1,3]	36.15	0	0.09	0.04	0.17
MR	[1,4]	38.24	0.032	0.08	0.08	0.12
MR	[2,3]	36.15	0	0.13	0.04	0.17
MR	[2,4]	38.24	0.032	0.13	0.04	0.17
MR	[3,4]	38.24	0.032	0.13	0.04	0.17
MR	[1,2,3]	36.15	0.032	0.16	0.04	0.26
MR	[1,2,4]	38.24	0.032	0.16	0.1	0.22
MR	[1,3,4]	38.24	0	0.16	0.06	0.21
MR	[2,3,4]	38.24	0.032	0.16	0.05	0.25
MR	[1,2,3,4]	38.24	0.032	0.15	0.1	0.3
				0.18	0.12	

Table C4
Case 4 (Separable Classes)

Code of Algorithm	Combinations of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
ML	[1]	85.36	0.682	0.2	0.14	0.34
ML	[2]	89.69	0.768	0.19	0.18	0.37
ML	[3]	89.69	0.769	0.19	0.18	0.37
ML	[4]	90.93	0.795	0.2	0.18	0.38
ML	[1,2]	89.69	0.77	0.25	0.32	0.57
ML	[1,3]	90.93	0.796	0.25	0.32	0.57
ML	[1,4]	90.72	0.791	0.29	0.28	0.57
ML	[2,3]	92.99	0.84	0.24	0.33	0.57
ML	[2,4]	92.16	0.822	0.25	0.32	0.57
ML	[3,4]	94.85	0.881	0.29	0.28	0.57
ML	[1,2,3]	92.78	0.837	0.36	0.49	0.85
ML	[1,2,4]	91.55	0.808	0.36	0.45	0.81
ML	[1,3,4]	94.02	0.862	0.35	0.45	0.8
ML	[2,3,4]	94.23	0.867	0.32	0.49	0.81
ML	[1,2,3,4]	94.64	0.878	0.48	0.7	1.18
FM	[1]	85.57	0.675	0.21	0.6	0.81
FM	[2]	89.28	0.753	0.2	0.6	0.8
FM	[3]	90.93	0.792	0.21	0.59	0.8
FM	[4]	89.07	0.745	0.2	0.6	0.8
FM	[1,2]	89.48	0.756	0.26	1.44	1.7
FM	[1,3]	90.93	0.787	0.33	0.99	1.32
FM	[1,4]	89.69	0.756	0.31	1.23	1.54
FM	[2,3]	92.78	0.831	0.27	1.4	1.67
FM	[2,4]	90.52	0.776	0.32	0.98	1.3
FM	[3,4]	95.05	0.884	0.3	0.99	1.29
FM	[1,2,3]	94.02	0.86	0.37	1.48	1.85
FM	[1,2,4]	90.93	0.786	0.42	1.5	1.92
FM	[1,3,4]	94.02	0.859	0.37	1.48	1.85
FM	[2,3,4]	93.61	0.85	0.37	1.46	1.83
FM	[1,2,3,4]	94.02	0.859	0.45	2.1	2.55
FD	[1]	85.15	0.671	0.21	0.46	0.67
FD	[2]	88.45	0.738	0.21	0.46	0.67
FD	[3]	91.13	0.797	0.22	0.45	0.67
FD	[4]	89.07	0.748	0.2	0.45	0.65
FD	[1,2]	88.66	0.745	0.32	0.77	1.09
FD	[1,3]	91.96	0.815	0.26	0.83	1.09
FD	[1,4]	89.48	0.757	0.26	0.79	1.05
FD	[2,3]	93.61	0.853	0.31	0.78	1.09
FD	[2,4]	92.16	0.82	0.33	0.76	1.09
FD	[3,4]	95.05	0.885	0.26	0.83	1.09
FD	[1,2,3]	95.05	0.886	0.44	1.18	1.62
FD	[1,2,4]	91.96	0.815	0.37	1.22	1.59
FD	[1,3,4]	94.64	0.875	0.42	1.17	1.59
FD	[2,3,4]	94.23	0.865	0.37	1.22	1.59
FD	[1,2,3,4]	94.85	0.881	0.5	1.74	2.24
SM	[1]	67.01	0.014	0.08	0.18	0.26
SM	[2]	68.45	0.055	0.05	0.18	0.23
SM	[3]	89.69	0.769	0.03	0.19	0.22
SM	[4]	68.45	0.005	0.04	0.18	0.22
SM	[1,2]	68.04	0.051	0.1	0.29	0.39
SM	[1,3]	91.34	0.802	0.13	0.32	0.45
SM	[1,4]	69.69	0.075	0.06	0.32	0.38
SM	[2,3]	92.58	0.828	0.13	0.29	0.42
SM	[2,4]	69.48	0.079	0.1	0.29	0.39
SM	[3,4]	95.05	0.883	0.13	0.32	0.45
SM	[1,2,3]	92.78	0.833	0.17	0.5	0.67
SM	[1,2,4]	69.48	0.079	0.13	0.49	0.62
SM	[1,3,4]	94.85	0.878	0.17	0.49	0.66
SM	[2,3,4]	94.85	0.878	0.13	0.49	0.62

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
SM	[1,2,3,4]	95.05	0.883	0.2	0.7	0.9
SF	[1]	80.62	0.537	0.06	0.6	0.66
SF	[2]	84.12	0.61	0.05	0.6	0.65
SF	[3]	90.93	0.792	0.07	0.59	0.66
SF	[4]	85.36	0.641	0.05	0.59	0.64
SF	[1,2]	87.84	0.713	0.13	0.95	1.08
SF	[1,3]	90.93	0.787	0.18	0.95	1.13
SF	[1,4]	89.69	0.756	0.15	0.95	1.1
SF	[2,3]	92.78	0.831	0.18	1	1.18
SF	[2,4]	90.52	0.776	0.15	0.95	1.1
SF	[3,4]	95.05	0.884	0.12	0.97	1.09
SF	[1,2,3]	94.02	0.86	0.22	1.45	1.67
SF	[1,2,4]	90.93	0.786	0.18	1.44	1.62
SF	[1,3,4]	94.02	0.859	0.23	1.45	1.68
SF	[2,3,4]	93.61	0.85	0.23	1.44	1.67
SF	[1,2,3,4]	94.02	0.859	0.34	2.06	2.4
IE	[1]	83.92	0.604	0.06	0.01	0.07
IE	[2]	83.92	0.59	0.01	0.02	0.03
IE	[3]	91.96	0.813	0.02	0.01	0.03
IE	[4]	86.8	0.675	0.01	0.02	0.03
IE	[1,2]	83.92	0.59	0.03	0.02	0.05
IE	[1,3]	90.72	0.779	0.03	0.02	0.05
IE	[1,4]	84.74	0.611	0.03	0.02	0.05
IE	[2,3]	92.16	0.811	0.03	0.02	0.05
IE	[2,4]	84.12	0.59	0.03	0.02	0.05
IE	[3,4]	92.78	0.826	0.03	0.02	0.07
IE	[1,2,3]	91.34	0.79	0.04	0.03	0.11
IE	[1,2,4]	84.12	0.588	0.07	0.04	0.11
IE	[1,3,4]	91.34	0.788	0.04	0.07	0.11
IE	[2,3,4]	91.96	0.805	0.08	0.03	0.12
IE	[1,2,3,4]	91.13	0.783	0.04	0.08	
BF	[1]	84.12	0.662	0.04	0.02	0.06
BF	[2]	76.08	0.536	0.04	0.02	0.06
BF	[3]	87.01	0.671	0.04	0.02	0.06
BF	[4]	88.87	0.734	0.08	0.03	0.07
BF	[1,2]	77.11	0.553	0.12	0.04	0.12
BF	[1,3]	91.34	0.802	0.08	0.04	0.16
BF	[1,4]	89.28	0.741	0.08	0.08	0.12
BF	[2,3]	82.89	0.651	0.12	0.04	0.16
BF	[2,4]	90.52	0.791	0.08	0.04	0.16
BF	[3,4]	90.72	0.774	0.15	0.08	0.21
BF	[1,2,3]	83.71	0.666	0.12	0.06	0.21
BF	[1,2,4]	90.72	0.795	0.12	0.09	0.21
BF	[1,3,4]	90.72	0.773	0.16	0.09	0.22
BF	[2,3,4]	93.2	0.847	0.19	0.06	0.27
BF	[1,2,3,4]	93.4	0.851		0.08	
MB	[1]	84.12	0.662	0.08	0.02	0.1
MB	[2]	76.08	0.536	0.14	0.02	0.16
MB	[3]	87.01	0.671	0.12	0.02	0.14
MB	[4]	88.87	0.734	0.13	0.03	0.16
MB	[1,2]	77.11	0.553	0.13	0.08	0.21
MB	[1,3]	91.34	0.802	0.17	0.04	0.21
MB	[1,4]	89.28	0.741	0.18	0.04	0.22
MB	[2,3]	82.89	0.651	0.18	0.08	0.26
MB	[2,4]	90.52	0.791	0.13	0.08	0.21
MB	[3,4]	90.72	0.774	0.17	0.04	0.21
MB	[1,2,3]	83.71	0.666	0.22	0.05	0.27
MB	[1,2,4]	90.72	0.795	0.22	0.05	0.27
MB	[1,3,4]	90.72	0.773	0.22	0.05	0.27
MB	[2,3,4]	93.2	0.847	0.22	0.05	0.27
MB	[1,2,3,4]	93.4	0.851	0.27	0.11	0.38
MC	[1]	80.82	0.59	0.07	0.01	0.08
MC	[2]	84.54	0.657	0.07	0.01	0.08

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
MC	[3]	89.48	0.764	0.07	0.01	0.08
MC	[4]	86.8	0.703	0.07	0.02	0.09
MC	[1,2]	80.82	0.59	0.11	0.03	0.14
MC	[1,3]	80.82	0.59	0.07	0.06	0.13
MC	[1,4]	80.82	0.59	0.07	0.02	0.09
MC	[2,3]	84.54	0.657	0.07	0.03	0.1
MC	[2,4]	84.54	0.657	0.12	0.02	0.14
MC	[3,4]	89.48	0.764	0.07	0.06	0.13
MC	[1,2,3]	80.82	0.59	0.07	0.03	0.1
MC	[1,2,4]	80.82	0.59	0.12	0.03	0.15
MC	[1,3,4]	80.82	0.59	0.07	0.08	0.15
MC	[2,3,4]	84.54	0.657	0.07	0.03	0.1
MC	[1,2,3,4]	80.82	0.59	0.04	0.08	0.15
MM	[1]	80.82	0.59	0.04	0.01	0.05
MM	[2]	84.54	0.657	0.04	0.05	0.09
MM	[3]	89.48	0.764	0.04	0.05	0.09
MM	[4]	86.8	0.703	0.04	0.02	0.06
MM	[1,2]	80.82	0.59	0.04	0.03	0.07
MM	[1,3]	80.82	0.59	0.04	0.02	0.06
MM	[1,4]	80.82	0.59	0.06	0.02	0.08
MM	[2,3]	84.54	0.657	0.04	0.02	0.06
MM	[2,4]	84.54	0.657	0.04	0.02	0.06
MM	[3,4]	89.48	0.764	0.09	0.02	0.11
MM	[1,2,3]	80.82	0.59	0.04	0.07	0.11
MM	[1,2,4]	80.82	0.59	0.04	0.08	0.12
MM	[1,3,4]	80.82	0.59	0.04	0.03	0.07
MM	[2,3,4]	84.54	0.657	0.04	0.03	0.07
MM	[1,2,3,4]	80.82	0.59	0.04	0.09	0.13
AA	[1]	82.27	0.589	0.15	0.2	0.35
AA	[2]	83.92	0.608	0.11	0.19	0.3
AA	[3]	91.96	0.813	0.03	0.19	0.22
AA	[4]	84.12	0.611	0.12	0.19	0.31
AA	[1,2]	84.33	0.628	0.2	0.34	0.54
AA	[1,3]	91.75	0.808	0.11	0.3	0.41
AA	[1,4]	86.6	0.685	0.23	0.34	0.57
AA	[2,3]	92.16	0.817	0.07	0.35	0.42
AA	[2,4]	88.04	0.715	0.23	0.34	0.57
AA	[3,4]	94.23	0.862	0.11	0.31	0.42
AA	[1,2,3]	92.58	0.826	0.14	0.49	0.63
AA	[1,2,4]	88.45	0.728	0.33	0.49	0.82
AA	[1,3,4]	94.02	0.857	0.1	0.53	0.63
AA	[2,3,4]	94.23	0.862	0.14	0.53	0.67
AA	[1,2,3,4]	94.43	0.868	0.13	0.79	0.92
AI	[1]	82.27	0.589	0.15	0.2	0.35
AI	[2]	83.92	0.608	0.15	0.19	0.34
AI	[3]	91.96	0.813	0.03	0.2	0.23
AI	[4]	84.12	0.611	0.16	0.19	0.35
AI	[1,2]	86.8	0.693	0.21	0.35	0.56
AI	[1,3]	90.93	0.796	0.25	0.35	0.6
AI	[1,4]	88.25	0.727	0.28	0.31	0.59
AI	[2,3]	92.99	0.84	0.31	0.34	0.65
AI	[2,4]	89.07	0.742	0.23	0.35	0.58
AI	[3,4]	94.85	0.881	0.25	0.34	0.59
AI	[1,2,3]	92.58	0.832	0.37	0.52	0.89
AI	[1,2,4]	90.31	0.775	0.35	0.52	0.87
AI	[1,3,4]	94.02	0.862	0.36	0.49	0.85
AI	[2,3,4]	94.23	0.867	0.37	0.53	0.9
AI	[1,2,3,4]	94.43	0.873	0.44	0.74	1.18
IA	[1]	82.27	0.589	0.15	0.2	0.35
IA	[2]	83.92	0.608	0.15	0.19	0.34
IA	[3]	91.96	0.813	0.03	0.19	0.22
IA	[4]	84.12	0.611	0.16	0.19	0.35
IA	[1,2]	84.33	0.628	0.2	0.34	0.54
IA	[1,3]	91.75	0.808	0.07	0.34	0.41

Code of Algorithm	Combinations of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
IA	[1,4]	86.6	0.685	0.23	0.35	0.58
IA	[2,3]	92.16	0.817	0.07	0.34	0.41
IA	[2,4]	88.04	0.715	0.22	0.35	0.57
IA	[3,4]	94.23	0.862	0.11	0.31	0.42
IA	[1,2,3]	92.58	0.826	0.14	0.49	0.63
IA	[1,2,4]	88.45	0.728	0.33	0.49	0.82
IA	[1,3,4]	94.02	0.857	0.1	0.53	0.63
IA	[2,3,4]	94.23	0.862	0.14	0.53	0.67
IA	[1,2,3,4]	94.43	0.868	0.13	0.79	0.92
II	[1]	82.27	0.589	0.15	0.2	0.35
II	[2]	83.92	0.608	0.15	0.19	0.34
II	[3]	91.96	0.813	0.03	0.19	0.22
II	[4]	84.12	0.611	0.16	0.2	0.36
II	[1,2]	86.8	0.693	0.21	0.35	0.56
II	[1,3]	90.93	0.796	0.25	0.36	0.61
II	[1,4]	88.25	0.727	0.27	0.31	0.58
II	[2,3]	92.99	0.84	0.3	0.34	0.64
II	[2,4]	89.07	0.742	0.23	0.34	0.57
II	[3,4]	94.85	0.881	0.25	0.34	0.59
II	[1,2,3]	92.58	0.832	0.37	0.53	0.9
II	[1,2,4]	90.31	0.775	0.35	0.53	0.88
II	[1,3,4]	94.02	0.862	0.36	0.49	0.85
II	[2,3,4]	94.23	0.867	0.37	0.53	0.9
II	[1,2,3,4]	94.43	0.873	0.45	0.74	1.19
XX	[1]	85.57	0.675	0.17	0.61	0.78
XX	[2]	89.28	0.753	0.17	0.61	0.78
XX	[3]	90.93	0.792	0.1	0.61	0.71
XX	[4]	89.07	0.745	0.12	0.65	0.77
XX	[1,2]	89.48	0.756	0.27	0.98	1.25
XX	[1,3]	90.93	0.787	0.12	1.02	1.14
XX	[1,4]	89.69	0.756	0.28	0.97	1.25
XX	[2,3]	92.78	0.831	0.11	1.03	1.14
XX	[2,4]	90.52	0.776	0.28	0.98	1.26
XX	[3,4]	95.05	0.884	0.11	1.02	1.13
XX	[1,2,3]	94.02	0.86	0.26	1.47	1.73
XX	[1,2,4]	90.93	0.786	0.34	1.48	1.82
XX	[1,3,4]	94.02	0.859	0.21	1.47	1.68
XX	[2,3,4]	93.61	0.85	0.26	1.48	1.74
XX	[1,2,3,4]	94.02	0.859	0.31	2.1	2.41
XN	[1]	85.57	0.675	0.18	0.6	0.78
XN	[2]	89.28	0.753	0.16	0.61	0.77
XN	[3]	90.93	0.792	0.06	0.61	0.67
XN	[4]	89.07	0.745	0.17	0.61	0.78
XN	[1,2]	89.48	0.756	0.22	0.98	1.2
XN	[1,3]	90.93	0.787	0.3	0.97	1.27
XN	[1,4]	89.69	0.756	0.25	1.02	1.27
XN	[2,3]	92.78	0.831	0.24	0.98	1.22
XN	[2,4]	90.52	0.776	0.28	0.98	1.26
XN	[3,4]	95.05	0.884	0.25	1.02	1.27
XN	[1,2,3]	94.02	0.86	0.37	1.47	1.84
XN	[1,2,4]	90.93	0.786	0.35	1.47	1.82
XN	[1,3,4]	94.02	0.859	0.37	1.47	1.84
XN	[2,3,4]	93.61	0.85	0.37	1.47	1.84
XN	[1,2,3,4]	94.02	0.859	0.5	2.12	2.62
NX	[1]	85.57	0.675	0.17	0.61	0.78
NX	[2]	89.28	0.753	0.16	0.61	0.77
NX	[3]	90.93	0.792	0.07	0.6	0.67
NX	[4]	89.07	0.745	0.18	0.61	0.79
NX	[1,2]	89.48	0.756	0.21	0.99	1.2
NX	[1,3]	90.93	0.787	0.16	0.97	1.13
NX	[1,4]	89.69	0.756	0.23	0.98	1.21
NX	[2,3]	92.78	0.831	0.16	0.97	1.13
NX	[2,4]	90.52	0.776	0.28	0.98	1.26
NX	[3,4]	95.05	0.884	0.12	1.02	1.14

Code of Algorithm	Combina-tions of Bands	Overall Accuracy (OA %)	KHAT	Class Assignment Time(CAT)	Statistics Computing Time(SCT)	Total Classification Time(CT)
NX	[1,2,3]	94.02	0.86	0.25	1.47	1.72
NX	[1,2,4]	90.93	0.786	0.34	1.48	1.82
NX	[1,3,4]	94.02	0.859	0.21	1.48	1.69
NX	[2,3,4]	93.61	0.85	0.25	1.48	1.73
NX	[1,2,3,4]	94.02	0.859	0.31	2.1	2.41
NN	[1]	85.57	0.675	0.18	0.6	0.78
NN	[2]	89.28	0.753	0.16	0.6	0.76
NN	[3]	90.93	0.792	0.06	0.61	0.67
NN	[4]	89.07	0.745	0.17	0.61	0.78
NN	[1,2]	89.48	0.756	0.22	0.98	1.2
NN	[1,3]	90.93	0.787	0.3	0.97	1.27
NN	[1,4]	89.69	0.756	0.25	1.02	1.27
NN	[2,3]	92.78	0.831	0.24	0.98	1.22
NN	[2,4]	90.52	0.776	0.27	0.99	1.26
NN	[3,4]	95.05	0.884	0.25	1.02	1.27
NN	[1,2,3]	94.02	0.86	0.37	1.47	1.84
NN	[1,2,4]	90.93	0.786	0.35	1.48	1.83
NN	[1,3,4]	94.02	0.859	0.37	1.47	1.84
NN	[2,3,4]	93.61	0.85	0.37	1.47	1.84
NN	[1,2,3,4]	94.02	0.859	0.51	2.11	2.62
RF	[1]	54.64	0.243	0.02	0.01	0.03
RF	[2]	54.43	0.239	0.06	0.01	0.07
RF	[3]	42.06	0.101	0.02	0	0.02
RF	[4]	82.68	0.65	0.02	0.01	0.03
RF	[1,2]	60	0.308	0.03	0.01	0.04
RF	[1,3]	60.41	0.313	0.02	0.02	0.04
RF	[1,4]	84.12	0.675	0.03	0.01	0.04
RF	[2,3]	58.56	0.29	0.03	0.01	0.04
RF	[2,4]	84.74	0.686	0.03	0.01	0.04
RF	[3,4]	86.39	0.717	0.02	0.01	0.03
RF	[1,2,3]	63.92	0.36	0.03	0.02	0.05
RF	[1,2,4]	85.15	0.694	0.04	0.02	0.05
RF	[1,3,4]	87.42	0.736	0.03	0.02	0.05
RF	[2,3,4]	87.84	0.744	0.08	0.01	0.1
RF	[1,2,3,4]	88.04	0.748	0.09	0.02	0.11
MR	[1]	36.7	0.049	0.02	0.05	0.07
MR	[2]	39.59	0.075	0.06	0.02	0.08
MR	[3]	32.16	0.006	0.02	0.01	0.03
MR	[4]	41.65	0.096	0.03	0.01	0.03
MR	[1,2]	39.79	0.077	0.04	0.02	0.05
MR	[1,3]	37.11	0.053	0.03	0.02	0.05
MR	[1,4]	42.27	0.103	0.03	0.02	0.05
MR	[2,3]	40.21	0.082	0.03	0.03	0.06
MR	[2,4]	42.68	0.107	0.08	0.02	0.1
MR	[3,4]	42.27	0.103	0.08	0.04	0.12
MR	[1,2,3]	40.21	0.082	0.04	0.07	0.11
MR	[1,2,4]	42.89	0.109	0.04	0.03	0.07
MR	[1,3,4]	42.68	0.107	0.08	0.03	0.11
MR	[2,3,4]	43.3	0.114	0.05	0.09	0.14
MR	[1,2,3,4]	43.3	0.114			

APPENDIX D

Confusion Matrices of Case 1 in Combination of Four Bands & K-Matrices for Four Cases

/* Confusion Matrix by ML on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	251	24	15	20	310	80.97
Urban	194	172	210	88	664	25.90
Tree	9	30	244	16	299	81.61
Sand	44	37	59	142	282	50.35
Total	498	263	528	266	809	59.71
UA	50.40	65.40	46.21	53.38	53.85	52.03

Time (sec) for 1. Classification : 5.678
2. Class Assignment : 3.244
3. Statistics : 2.434 Khat : 0.375

/* Confusion Matrix by FM on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	226	51	2	31	310	72.90
Urban	129	284	109	142	664	42.77
Tree	1	69	197	32	299	65.89
Sand	12	73	24	173	282	61.35
Total	368	477	332	378	880	60.73
UA	61.41	59.54	59.34	45.77	56.51	56.59

Time (sec) for 1. Classification : 12.408
2. Class Assignment : 4.287
3. Statistics : 8.121 Khat : 0.411

/* Confusion Matrix by FD on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	253	22	15	20	310	81.61
Urban	184	185	200	95	664	27.86
Tree	7	36	242	14	299	80.94
Sand	39	42	55	146	282	51.77
Total	483	285	512	275	826	60.55
UA	52.38	64.91	47.27	53.09	54.41	53.12

Time (sec) for 1. Classification : 14.901
2. Class Assignment : 5.908
3. Statistics : 8.993 Khat : 0.387

/* Confusion Matrix by 1 SM on 1555 TEST samples of 4669 [Bands - [1,2,3,4] pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	245	44	1	20	310	79.03
Urban	176	316	76	96	664	47.59
Tree	7	152	121	19	299	40.47
Sand	30	115	14	123	282	43.62
Total	458	627	212	258	805	52.68
UA	53.49	50.40	57.08	47.67	52.16	51.77

Time (sec) for 1. Classification : 3.475
2. Class Assignment : 1.041
3. Statistics : 2.434 Khat : 0.323

/* Confusion Matrix by 1 SF on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	211	76	0	23	310	68.06
Urban	120	350	65	129	664	52.71
Tree	1	124	144	30	299	48.16
Sand	11	97	15	159	282	56.38
Total	343	647	224	341	864	56.33
UA	61.52	54.10	64.29	46.63	56.63	55.56

Time (sec) for 1. Classification : 10.576
2. Class Assignment : 2.514
3. Statistics : 8.062 Khat : 0.375

/* Confusion Matrix by EF on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	228	64	0	18	310	73.55
Urban	152	363	53	96	664	54.67
Tree	3	139	143	14	299	47.83
Sand	24	120	11	127	282	45.04
Total	407	686	207	255	861	55.27
UA	56.02	52.92	69.08	49.80	56.96	55.37

Time (sec) for 1. Classification : 0.551
2. Class Assignment : 0.37
3. Statistics : 0.181 Khat : 0.366

/* Confusion Matrix by BF on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	209	35	52	14	310	67.42
Urban	130	95	329	110	664	14.31
Tree	4	2	283	10	299	94.65
Sand	17	39	112	114	282	40.43
Total	360	171	776	248	701	54.20
UA	58.06	55.56	36.47	45.97	49.01	45.08

Time (sec) for 1. Classification : 1.493
2. Class Assignment : 1.242
3. Statistics : 0.251 Khat : 0.298

/* Confusion Matrix by MB on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	201	19	51	39	310	64.84
Urban	109	54	326	175	664	8.13
Tree	4	2	280	13	299	93.65
Sand	16	38	111	117	282	41.49
Total	330	113	768	344	652	52.03
UA	60.91	47.79	36.46	34.01	44.79	41.93

Time (sec) for 1. Classification : 3.195
2. Class Assignment : 2.904
3. Statistics : 0.291 Khat : 0.266

/* Confusion Matrix by MC on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	255	20	12	23	310	82.26
Urban	250	65	119	230	664	9.79
Tree	43	34	150	72	299	50.17
Sand	59	16	64	143	282	50.71
Total	607	135	345	468	613	48.23
UA	42.01	48.15	43.48	30.56	41.05	39.42

Time (sec) for 1. Classification : 1.272
2. Class Assignment : 1.082
3. Statistics : 0.19 Khat : 0.231

/* Confusion Matrix by MM on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	255	20	12	23	310	82.26
Urban	250	65	119	230	664	9.79
Tree	43	34	150	72	299	50.17
Sand	59	16	64	143	282	50.71
Total	607	135	345	468	613	48.23
UA	42.01	48.15	43.48	30.56	41.05	39.42

Time (sec) for 1. Classification : 0.691
2. Class Assignment : 0.501
3. Statistics : 0.19 Khat : 0.231

/* Confusion Matrix by AA on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	256	37	0	17	310	82.58
Urban	189	308	59	108	664	46.39
Tree	9	198	80	12	299	26.76
Sand	40	119	7	116	282	41.13
Total	494	662	146	253	760	49.21
UA	51.82	46.53	54.79	45.85	49.75	48.87

Time (sec) for 1. Classification : 3.344
2. Class Assignment : 0.731
3. Statistics : 2.613 Khat : 0.277

/* Confusion Matrix by AI on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	240	47	3	20	310	77.42
Urban	182	266	120	96	664	40.06
Tree	5	111	164	19	299	54.85
Sand	33	85	24	140	282	49.65
Total	460	509	311	275	810	55.29
UA	52.17	52.26	52.73	50.91	52.02	52.09

Time (sec) for 1. Classification : 4.386
2. Class Assignment : 1.763
3. Statistics : 2.623 Khat : 0.344

/* Confusion Matrix by IA on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	251	26	13	20	310	80.97
Urban	193	189	192	90	664	28.46
Tree	9	43	230	17	299	76.92
Sand	43	43	53	143	282	50.71
Total	496	301	488	270	813	59.27
UA	50.60	62.79	47.13	52.96	53.37	52.28

Time (sec) for 1. Classification : 5.758
2. Class Assignment : 3.135
3. Statistics : 2.623 Khat : 0.374

/* Confusion Matrix by II on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	251	24	15	20	310	80.97
Urban	194	174	208	88	664	26.20
Tree	9	36	238	16	299	79.60
Sand	44	38	58	142	282	50.35
Total	498	272	519	266	805	59.28
UA	50.40	63.97	45.86	53.38	53.40	51.77

Time (sec) for 1. Classification : 5.939
2. Class Assignment : 3.315
3. Statistics : 2.624 Khat : 0.371

/* Confusion Matrix by XX on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	211	75	0	24	310	68.06
Urban	113	371	54	126	664	55.87
Tree	1	145	128	25	299	42.81
Sand	10	113	10	149	282	52.84
Total	335	704	192	324	859	54.90
UA	62.99	52.70	66.67	45.99	57.08	55.24

Time (sec) for 1. Classification : 10.335
2. Class Assignment : 2.053
3. Statistics : 8.282 Khat : 0.363

/* Confusion Matrix by XN on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	220	63	1	26	310	70.97
Urban	128	318	83	135	664	47.89
Tree	1	106	162	30	299	54.18
Sand	12	87	18	165	282	58.51
Total	361	574	264	356	865	57.89
UA	60.94	55.40	61.36	46.35	56.01	55.63

Time (sec) for 1. Classification : 11.726
2. Class Assignment : 3.445
3. Statistics : 8.281 Khat : 0.385

/* Confusion Matrix by NX on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	226	51	2	31	310	72.90
Urban	129	288	107	140	664	43.37
Tree	1	71	195	32	299	65.22
Sand	12	73	24	173	282	61.35
Total	368	483	328	376	882	60.71
UA	61.41	59.63	59.45	46.01	56.6	56.72

Time (sec) for 1. Classification : 12.758
2. Class Assignment : 4.496
3. Statistics : 8.262 Khat : 0.412

/* Confusion Matrix by NN on 1555 TEST samples of 4669. [Bands - [1,2,3,4] / pln_l3.rsd / pln_4_4669.smp / 2 : 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	226	51	2	31	310	72.90
Urban	129	284	109	142	664	42.77
Tree	1	69	197	32	299	65.89
Sand	12	73	24	173	282	61.35
Total	368	477	332	378	880	60.73
UA	61.41	59.54	59.34	45.77	56.51	56.59

Time (sec) for 1. Classification : 12.798
2. Class Assignment : 4.547
3. Statistics : 8.251 Khat : 0.411

/* Confusion Matrix by RF on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	0	113	1	196	310	0
Urban	0	48	0	616	664	7.23
Tree	0	0	0	299	299	0
Sand	0	0	1	281	282	99.65
Total	0	161	2	1392	329	26.72
UA	0	29.81	0	20.19	12.50	21.16

Time (sec) for 1. Classification : 0.501
2. Class Assignment : 0.431
3. Statistics : 0.07 Khat : 0.006

/* Confusion Matrix by MR on 1555 TEST samples of 4669 [Bands - [1,2,3,4] / pln_13.rsd / pln_4_4669.smp / 2 1] */

	Crop	Urban	Tree	Sand	Total	PA
Crop	0	0	0	310	310	0
Urban	0	0	0	664	664	0
Tree	0	0	0	299	299	0
Sand	0	0	0	282	282	100
Total	0	0	0	1555	282	25
UA	0	0	0	18.14	4.53	18.14

Time (sec) for 1. Classification : 0.661
2. Class Assignment : 0.481
3. Statistics : 0.18 Khat = 0.0

K-Matrices

K-Matrix (Case 1)

Class Pair	Band1	Band2	Band3	Band4
Crop / Urban	0.47	0.54	0.43	0.62
Crop / Tree	0.74	0.83	0.87	1.15
Crop / Sand	0.79	0.81	0.08	1.00
Urban / Tree	0.11	0.10	0.36	0.19
Urban / Sand	0.27	0.25	0.37	0.30
Tree / Sand	0.23	0.22	0.66	0.22

K-Matrix (Case 2)

Class Pair	Band1	Band2	Band3	Band4
Gram / Wheat	0.28	0.35	0.36	0.46
Gram / Urban	0.49	0.63	0.37	0.77
Gram / Sand	0.95	1.00	0.02	1.32
Wheat / Urban	0.21	0.26	0.07	0.26
Wheat / Sand	0.64	0.63	0.24	0.72
Urban / Sand	0.42	0.38	0.26	0.46

K-Matrix (Case 3)

Class Pair	Band1	Band2	Band3	Band4
Crop / Tree	0.50	0.53	1.24	0.53
Crop / Urban	0.24	0.32	0.90	0.41
Tree / Urban	0.29	0.24	0.07	0.15

K Matrix (Case 4)

Class Pair	Band1	Band2	Band3	Band4
Crop / Tree	0.40	0.44	1.15	0.46

APPENDIX E — Source Code (Programs in LPA WinProlog)

E1. Program to convert remote sensing data, given in each band in different external files, into Prolog facts

```
/*
  This program takes all band data files and
generate Prolog facts containing
  raw Remote Sensing Data. Here the output file is
OutFile = 'pln_l3.pl',
  Change name of Outfile in the clause pil/0,
compile and then simply execute

 :- pil.
*/

pil :-
  BandFilesList = ['band1.img', 'band2.img',
'band3.img', 'band4.img'],
  OutFile = 'pln_l3.pl',
  middle_image(BandFilesList,891,1001,300,OutFile).

% Pilani MaxCols (x-coord) = 891, MaxRows (y-coord)
= 1001
middle_image(BandFilesList,MaxCols,MaxRows,Size,
OutFile) :-
  StartCol is 100 + (MaxCols - Size)//2,
  StartRow is 100 + (MaxRows - Size)//2,
  tell(OutFile),

read_data(BandFilesList,MaxCols,MaxRows,StartCol,
StartRow,Size,Size),
  told.

read_image(BandFilesList,MaxCols,MaxRows,Cols,Ro
ws,OutFile) :-
  StartCol is (MaxCols - Cols)//2,
  StartRow is (MaxRows - Rows)//2,
  tell(OutFile),

read_data(BandFilesList,MaxCols,MaxRows,StartCol,
StartRow,Cols,Rows),
  told.

% read_data(BandDataFileList, TotalCols, TotalRows,
StartCol, StartRow, Cols, Rows).
% Col means X-Coord, Row means Y_Coord

read_data(FileList,TotCols,TotRows,StartCol,StartRo
w,HowManyCols,HowManyRows) :-
  ToSkip is (StartRow - 1)*TotCols + 1,
  skip_pixels(FileList,ToSkip),
  read_rows(FileList, TotCols, StartCol, 0,
HowManyRows, HowManyCols),
  close_files(FileList).

read_rows(_, _, _, RowsReadSoFar, HowManyRows, _)
:-
  RowsReadSoFar >= HowManyRows,!.

read_rows(FileList, TotCols, StartCol,
RowsReadSoFar, HowManyRows, HowManyCols) :-
  read_row(FileList, TotCols, StartCol, HowManyCols,
RowPixels),
  flatten(RowPixels, FlatList),
  string_chars(RowWord, FlatList),
  Term =.. [p,RowsReadSoFar,RowWord],
  nl,writeq(Term),write('.'),
  RowsReadSoFar1 is RowsReadSoFar + 1,
  read_rows(FileList, TotCols, StartCol, RowsReadSoFar1,
HowManyRows, HowManyCols).
read_row(FileList, TotalCols,StartCol, HowManyCols,
RowPixels) :-
  skip_pixels(FileList, StartCol),
  read_pixels(FileList, 0, HowManyCols, RowPixels),
  ToSkip is TotalCols - StartCol - HowManyCols + 2,
  skip_pixels(FileList,ToSkip).

% skip_pixels(FileList, HowMany)
skip_pixels([],_) :- !.
skip_pixels(_,0) :- !.
skip_pixels([FH|FT],HowMany) :-
  see(FH),
  \+ eof,!,
  inpos(PrevPos),
  NextPos is PrevPos + HowMany - 1,
  inpos(NextPos),
  skip_pixels(FT,HowMany).

skip_pixels([FH|FT],_) :-
  see(FH),
  seen,
  skip_pixels(FT,_).

read_pixels(_,HowMany, HowMany, []) :- !.
read_pixels(FileList, ReadSoFar, HowMany,
[Pixel|PixelsTail]) :-
  ReadSoFar < HowMany,
  read_pixel(FileList, Pixel),
  ReadSoFar1 is ReadSoFar + 1,
  read_pixels(FileList, ReadSoFar1, HowMany, PixelsTail).

% read_pixel(FileList, Pixel)
read_pixel([],[]) :- !.
read_pixel([FH|FT],[BH|BT]) :-
  see(FH),
  \+ eof,!,
  getx(1,BH),
  read_pixel(FT,BT).

read_pixel([FH|FT],[]) :-
  see(FH),
  seen,
  read_pixel(FT,[]).

close_files([]) :- !.
close_files([H|T]) :-
  see(H),
  seen,
  close_files(T).

% between generates a list of integers between Lower &
Upper (inclusive)
% if Upper & Lower sequence is given, then the list is in
descending order

between(H, H, [H]) :- !.
between(Lower, Upper, [Lower|Tail]) :-
  Lower < Upper,!,
  LowerNext is Lower + 1,
  between(LowerNext,Upper,Tail).
```

```
between(Upper, Lower, [Upper | Tail])
 Upper > Lower,!,
 UpperNext is Upper - 1,
 between(UpperNext,Lower,Tail).

flatten([Head | Tail],FlatList) :-
 flatten(Head,FlatHead),
 flatten(Tail, FlatTail),!,
 append(FlatHead, FlatTail, FlatList).
flatten([],[]).
flatten(X,[X]).
```

E2. Program to generate and save an image (color or grey) from the data stored in Prolog facts

```
/*
  A Simple Liss-III Image Display
==========================================
=================
  The liss3_eg/0 predicate runs a display of Liss-III
Image
  which displays a dialog containing a "Grafix"
window.

  For Liss-III Pixel =  [G,R,NIR,SWIR]
  Once the image is display take print-screen and
paste it in Paint brush to
  store image as BMP file
*/

:- dynamic(fcc_mapping/1), dynamic(image_size/3).

% create the dialog, set its handler and then show
the dialog
liss3_eg :-
  display_dialog( 'pln_l3.pl' ).

run :-
  liss3_eg('pln_l3.pl', [2,1,0]).

liss3_eg(DataFile, FCC_MapList) :-
  retractall(fcc_mapping(_)),
  asserta(fcc_mapping(FCC_MapList)),
  display_dialog(DataFile).

display_dialog(DataFile) :-
  asserta(datafile(DataFile)),
  see(DataFile),
  read(Term),
  (
   Term =.. [image_size,_, _, _]
       -> asserta(Term)
  )
  ; true
  ),
  create_gfx_eg(DataFile),
  window_handler( gfx_eg, gfx_eg_handler ),
  show_dialog( gfx_eg ),
  gfx_begin((gfx_eg,900)),
  ms(draw_image(DataFile),Ms), Time is Ms/1000.
nl,write(time/Time),nl,
  gfx_end((gfx_eg,900)).

% create the grafix example dialog windows

create_gfx_eg(DataFile) :-
  cat(['Image Display - ',DataFile], WinLabel,_),
  atom_string(WinLabel, WinLabelStr),
  wdcreate( gfx_eg, WinLabelStr,
                    20, 20, 450, 400, [ws_sysmenu,
ws_caption]),
  image_size(MaxRows, MaxCols, _),
  wccreate( (gfx_eg,900),
        grafix, 'Grafix',
                    16,  16, MaxCols, MaxRows, [ws_child,

ws_border,

ws_visible] ),

  wccreate( (gfx_eg,1),
        button, 'Ok',
                    32, 350,  50,  20, [ws_child,

ws_visible,

ws_tabstop,

bs_pushbutton] ).

% handle the close message by returning the atom "close"

gfx_eg_handler( _, msg_close, _, close ).

% handle the "ok" button by returning the atom "ok"

gfx_eg_handler( (gfx_eg,1), msg_button, _, ok ).


*
% handle paint messages by starting, drawing some
graphics, and ending

gfx_eg_handler( Win, msg_paint, grafix, _ ) :-
  gfx_paint( Win ),
  datafile(DataFile),
  see(DataFile),
  seen,
  see(DataFile),
  read(Term),

  Term =.. [image_size,MaxRows, MaxCols, Bands]
      -> asserta(Term) ; true

  draw_image(DataFile),
  seen,
  gfx_end( Win ).
*

image_size(300,300,4).
fcc_mapping([2,1,0]).

draw_image(DataFile) :-
  image_size(MaxRows, MaxCols, BandCount),
  fcc_mapping(FCC_MapList),
  StartRow is 0,
  Times is MaxCols - 1,
  append_times(Times,BandCount,Joins),
  draw_rows(DataFile, StartRow,MaxRows,MaxCols, Joins,
FCC_MapList),
  seen.

draw_rows(_, MaxRows,MaxRows,_,_,_) :- !.
draw_rows(DataFile,
Row,MaxRows,MaxCols,Joins,FCC_MapList) :-
  see(DataFile),
  read(Term),
  arg(2, Term, PixelStr),
  cat(List, PixelStr, Joins),
  ms(draw_this_row(Row,MaxCols,0,List, FCC_MapList),Ms),
  %write('/tot'/Ms),
  ToDraw is Row + 1,
```

```
draw_rows(DataFile,
ToDraw,MaxRows,MaxCols,Joins,FCC_MapList).

draw_this_row(_,Max,Max,_,_) :- !.
draw_this_row(ThisRow,MaxCols,ColNow,[Pixel|T],F
CC_MapList) :-
 string_chars(Pixel, PixelValues),
 fcc_mapping(FCC_MapList, PixelValues, [Red,
Green, Blue]),

 ms(gfx_pen_create(my_pen, Red, Green,
Blue,1),Ms1),
 ms(gfx( ( pen = my_pen -> polyline(ColNow,
ThisRow,ColNow, ThisRow) )), Ms2),
 %ms(gfx_pen_close(my_pen),Ms3),
 ColNext is ColNow + 1,
 %nl,write(pc/Ms1/gfx/Ms2).

draw_this_row(ThisRow,MaxCols,ColNext,T,FCC_Ma
pList).

append_times(0,_,[]) :- !.
append_times(Times, Ele, [Ele|Tail]) :-
 Times1 is Times - 1,
 append_times(Times1, Ele, Tail).

% fcc_mapping(Bands, Pixel, [R,G,B])
fcc_mapping([],_,[]) :- !.
fcc_mapping([BandNo/Weight|OtherTwoBands],
Pixel, [ColourValue|OtherTwoColours]) :-
 (
  (
   (
    member(Value, Pixel, BandNo),
    ColourValue is ip(Value*Weight)
   );
   (
    BandNo =< 0,
    ColourValue is abs(BandNo)
   )
  ),!,

fcc_mapping(OtherTwoBands, Pixel,
OtherTwoColours)

fcc_mapping([BandNo|OtherTwoBands], Pixel,
[ColourValue|OtherTwoColours]) :-
 (
  (
   member(ColourValue, Pixel, BandNo);
   (
    BandNo =< 0,
    ColourValue is abs(BandNo)
   )
  ),!,
  fcc_mapping(OtherTwoBands, Pixel,
OtherTwoColours).
```

E3. Program to display two different color composites and then permit selection and histogram generation of rectangular or square sample sites and then store these sites

```
/* To run this example, compile this program and
then run the goal:
    ?- train(ImageFile,ColorComposite1,
ColorComposite2, SampleFile).
```

For Example:

```
    ?- train('pln_l3.pl', 'pln_icc.bmp', 'pilani_210.bmp').
    ?- train('pln_l3.pl', 'pln_icc.bmp', 'pln_class.smp').
*/

test :-
  train('pln_l3.pl', 'pln_icc.bmp', 'pilani_210.bmp',
'pln_train.smp').

%%%%%%%%%%%%%%%%%%%%
% USER INTERFACE %
%%%%%%%%%%%%%%%%%%%%

% initialise data, prepare graphics objects, and create the
dialog

train(DataFile, BitMap1, BitMap2) :-
  tidy_train,
  init_train(DataFile,BitMap1,BitMap2),
  create_train_window,
  window_handler( train, train_handler ),
  call_dialog( train, _ ),
  !,
  tidy_train.

% initialise data, prepare graphics objects, and create the
dialog

train(DataFile, BitMap1, BitMap2, SampleFile) :-
  tidy_train,
  init_train(DataFile,BitMap1,BitMap2),
  create_train_window,
  (
   consult(SampleFile); true
  ),
  (
   (
    forall( sample_details(SampleNo,SiteName,Range),
         assertz(sample(SampleNo,SiteName,Range) )
        ),
    abolish_files([SampleFile])
   )
   ; true
  ),
  window_handler( train, train_handler ),
  inform_samples(train,1),
  call_dialog( train, _ ),
  !,
  tidy_train.

create_train_window :-
   Dstyle = [ws_caption,ws_sysmenu,
ws_maximizebox,ws_minimizebox, ws_thickframe],
   Bstyle = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
   Sstyle = [ws_child,ws_visible,ss_left],
   Gstyle = [ws_child,ws_visible],
   Cstyle =
[ws_child,ws_visible,ws_border,ws_tabstop,cbs_dropdown,w
s_vscroll],
   Estyle =
[ws_child,ws_visible,ws_border,ws_tabstop,es_left],

   wdcreate( train, 'Training Samples Dialog', 0, 0, 640,
455, Dstyle ),

   wccreate( (train,3), button, '&Add', 10, 320, 60, 20,
Bstyle ),
   wccreate( (train,4), button, '&Delete', 90, 320, 60, 20,
Bstyle ),
   wccreate( (train,5), button, '&Unselect',170, 320, 60, 20,
Bstyle ),
   wccreate( (train,6), button, '&Classify', 10, 360, 60, 20,
Bstyle ),
```

```
  wccreate( (train,7), button, &Histo ,   90, 360,
60, 20, Bstyle ),
  wccreate( (train,8), button, &Exit ,   170, 360,
60, 20, Bstyle ),
  wccreate( (train,9), grafix, ,   10, 10, 300,
300, Gstyle ),
  wccreate( (train,10), grafix, ,   320, 10, 300,
300, Gstyle ),

  wccreate( (train,15), static, 'SNo ',   260, 320,
40, 16, Sstyle ),
  wccreate( (train,16), static, 'Site Name', 320, 320,
100, 16, Sstyle),
  wccreate( (train,17), static, 'Region ',   440, 320,
100, 16, Sstyle ),
  wccreate( (train,18), static, '#Pxls ',   560, 320,
50, 16, Sstyle ),

  wccreate( (train,11), combobox, 'SNo',   260, 340,
40, 64, Cstyle ),
  wccreate( (train,12), combobox, 'Site Name',   320,
340, 100, 64, Cstyle),
  wccreate( (train,13), combobox, 'Region', 440, 340,
100, 64, Cstyle ),
  wccreate( (train,14), combobox, '#Pxls', 560, 340,
50, 64, Cstyle ),

  wccreate( (train,19), edit, '',320, 405, 100, 16,
Estyle ),
  wccreate( (train,20), button, 'Change &Name',440,
405, 100, 16, Bstyle ),
  wenable((train,20),0),
  !,
  set_buttons( 0, 0, 0, 0, 0, 1 )

% tidy up dynamic data and graphics objects

tidy_train :-
  abolish([temp_gfx/3,
temp_select/4,temp_mouse/4,sample/3,temp_sampl
e/3,class_clicked/1,last_mouse_pos/3]),
  catch( _, gfx_bitmap_close(paper1) ),
  catch( _, gfx_bitmap_close(paper2) ),
  catch( _, gfx_brush_close(red) ),
  catch( _, gfx_brush_close(yellow) ),
  catch( _, gfx_pen_close(white) ),
  catch( _, wclose(train) ),
  !.

% initialise dynamic data and graphics objects

init_train(DataFile, BitMap1, BitMap2) :-
  dynamic( [temp_gfx/3,
temp_select/4,temp_mouse/4,sample/3,temp_sampl
e/3, class_clicked/1, last_mouse_pos/3 ]),
  asserta(last_mouse_pos(dummy, dummy, 0)),
  (ensure_loaded(DataFile);true),
  absolute_file_name(BitMap1, BMPfile1 ),
  absolute_file_name(BitMap2, BMPfile2 ),
  gfx_bitmap_load( paper1, BMPfile1 ),
  gfx_bitmap_load( paper2, BMPfile2 ),
  gfx_brush_create( red, 255, 0, 0, solid ),
  gfx_brush_create( yellow, 255, 255, 0, solid ),
  gfx_pen_create( white, 255, 255, 255, 3 ),
  !.

%%%%%%%%%%%%%%%%%%%%
% WINDOW HANDLER %
%%%%%%%%%%%%%%%%%%%%

% perform an addition of class
train_handler( (train,3), msg_button, _, _ ) :-
  train(add), set_buttons(0,0,0,1,0,1).

% perform a removal of class
train_handler( (train,4), msg_button, _, _ ) :-
  train(delete), set_buttons(0,0,0,1,0,1).

% performs unselection of current sample selection
train_handler( (train,5), msg_button, _, _ ) :-
  (retractall(class_clicked(_)) ; true),
  train(unselect), set_buttons(0,0,0,1,0,1).

% stores classes
train_handler( (train,6), msg_button, _, _ ) :-
  train(classify), set_buttons(0,0,0,0,0,1).

% Tries to draw histogram
train_handler( (train,7), msg_button, _, _ ) :-
  %train(unselect),
  %draw_histo(1,2,[1,2,2,3,1,2,3,4,3,5]),
  set_buttons(0,0,0,1,0,1).

train_handler( (train,8), msg_button, _, done ) :- !.
train_handler( train, msg_close, _, done ) :- !.

/*
% adjust controls if the dialog has been resized
train_handler( train, msg_size, _, _ ) :-
  wsize( train, _, _, W, H ),
  BL is W - 100,
  GR is W - 120,
  GB is H - 60,
  SR is W - 40,
  ST is H - 45,
  wsize( (train,3), BL,  8, 80, 22 ),
  wsize( (train,4), BL, 38, 80, 22 ),
  wsize( (train,5), BL, 68, 80, 22 ),
  wsize( (train,6), BL, 98, 80, 22 ),
  wsize( (train,8), 10, ST, SR, 25 ),
  wsize( (train,9), 10, 10, GR, GB ),
  gfx_window_redraw( (train,9), 0, 0, GR, GB ).
*/

% repaint the window when necessary
train_handler( (train,ID), msg_paint, grafix, _ ) :-
  findall(Span, sample(_,_,Span), Spans),
  gfx_paint( (train,ID) ),
  % set_scale,
  ((ID = 9 -> draw_paper1); draw_paper2),
  draw_samples((train,ID), Spans, notxorpen_rop),
  gfx_end( (train,ID) ).

% handle a left down message in the grafix window
train_handler( (train,ID), msg_leftdown, (X,Y), _ ) :-
  ID > 8, ID < 11,
  train(unselect),
  gfx_begin( (train,ID) ),
  % set_scale,
  gfx_transform( X0, Y0, X, Y ),
  gfx_end( (train,ID) ),
  assert( temp_mouse(X0,Y0,X0,Y0) ).

% handle a mouse move message in the grafix window

train_handler( (train,ID), msg_mousemove, (X,Y), _ ) :-
  ID > 8, ID < 11, ((ID = 9, OtherID = 10); OtherID = 9),
  ( retract( temp_mouse(X0,Y0,X1,Y1) )
  -> gfx_begin( (train,ID) ),
    % set_scale,
    gfx_transform( X2, Y2, X, Y ),
    gfx( (  rop = stock(notxorpen_rop)
         -> polyline(X0,Y0,X1,Y0,X1,Y1,X0,Y1,X0,Y0),
            polyline(X0,Y0,X2,Y0,X2,Y2,X0,Y2,X0,Y0)
         )
       ),
```

```
  gfx_end( (train,ID) ).

  gfx_begin( (train,OtherID) ),
  gfx( ( rop = stock(notxorpen_rop)
     -> polyline(X0,Y0,X1,Y0,X1,Y1,X0,Y1,X0,Y0),
       polyline(X0,Y0,X2,Y0,X2,Y2,X0,Y2,X0,Y0)
     )
    ),
  gfx_end( (train,OtherID) ),
  assert( temp_mouse(X0,Y0,X2,Y2) )
)
;
(
  show_if_inside(X,Y,SNo),
  SNo > 0  -> gfx_window_cursor( (train,ID),
stock(cross_cursor) )
  ;
  gfx_window_cursor( (train,ID), stock(arrow_cursor)
)
  ).

% handle a left up message in the grafix window

train_handler( (train,ID), msg_leftup, _, _ ) :-
  ID > 8, ID < 11,
  ( retract( temp_mouse(X0,Y0,X1,Y1) )
  -> keys( Keys ),
    ( 0 is Keys /\ 3
    -> dynamic( temp_select/4 )
    ; true
    ),
    (
    (X0,Y0) \= (X1,Y1) ->
      draw_sample( (train,ID), [X0,Y0,X1,Y1],
xorpen_rop )
    ),
    (retract( temp_select(_,_,_,_) ) ; true),
    asserta( temp_select(X0,Y0,X1,Y1) ),
    set_buttons(1,0,1,0,0,1)
  ).

% handle a mouse-right click for the inserted classes
% first finds the region where right_clicked and then
allow delete

train_handler( (train,ID), msg_rightdown, (X,Y), _ ) :-
  ID > 8, ID < 11,
  (
  (
  sample(ClassNo,_,[X0, Y0, X1, Y1]),
  X >= X0, X =< X1, Y >= Y0, Y =< Y1,
  ( retractall(class_clicked(_)) ; true ),
  asserta(class_clicked(ClassNo)),
  Pos is ClassNo - 1,
  List = [(train,11), (train,12), (train,13), (train,14)],
  sel_in_at(List,Pos),
  set_buttons(0,1,0,0,0,1)
  ) ;
  true
  ).

% ignore all other grafix window messages during
classification
train_handler( (train,ID), _, _, _ ) :-
  ID > 8, ID < 11,
  wenable( (train,ID), 1).

% handle a button for the inserted classes
train_handler( (train,ID), msg_select, _, _ ) :-
  ID > 10,
  ID < 15,
  List = [(train,11), (train,12), (train,13), (train,14)],
  remove((train,ID),List, List1),
  get_lbx_selection((train,ID), [Pos]),
  sel_in_at(List1,Pos),
  ClassNo is Pos + 1,
  ( retractall(class_clicked(_)) ; true),
  asserta(class_clicked(ClassNo)),
  set_buttons(0,1,0,1,0,1).

% handle a button for changing name of the sites
train_handler( (train,20), msg_button, _, _ ) :-
  wtext((train,19), NewName),
  NewName \== '',
  get_lbx_selection((train,12), [Pos]),
  wlbxdel((train,12),Pos),
  wlbxadd((train,12),Pos,NewName),
  wlbxsel((train,12),Pos,1),
  SiteNo is Pos + 1,
  retract(sample(SiteNo,_,Span)),
  assert(sample(SiteNo,NewName,Span), SiteNo),
  wtext((train,19),''),
  wenable((train,20),0),
  set_buttons(0,1,0,1,0,1).

% handle edit box for changing name of the sites
train_handler( (train,19), msg_focus, _, _ ) :-
  wenable((train,20),1).

% handles Add button to add a site
train(add) :-
  next_class(SNo),
  retract(temp_select(X1,Y1,X2,Y2)),
  number_string(SNo, SNoStr),
  cat(['Sample',SNoStr],SampleName,_),
  assertz(sample(SNo,SampleName,[X1,Y1,X2,Y2])),

  ( (X1,Y1) \= (X2,Y2)
    ->
    (
    draw_sample((train, 9), [X1,Y1,X2,Y2],xorpen_rop ),
    draw_sample((train,10), [X1,Y1,X2,Y2],xorpen_rop )
    )
  ),

  inform_samples(train,SNo).

train(unselect) :-
( retract( temp_select(X0,Y0,X1,Y1) )
  -> keys( Keys ),
    ( 0 is Keys /\ 3
    -> dynamic( temp_select/4 )
    ; true
    ),
    ( (X0,Y0) \= (X1,Y1)
      ->
      (
      draw_sample((train, 9), [X0,Y0,X1,Y1],notxorpen_rop
),      draw_sample((train,10), [X0,Y0,X1,Y1],notxorpen_rop
)
      )
    )
).

  train(unselect).

  train(delete) :-
    retract(class_clicked(CNo)),
    Pos is CNo - 1,
    forall( ( sample(CNo1,_,_), CNo1 >= CNo),
          (
          wlbxdel((train,11),Pos),
          wlbxdel((train,12),Pos),
          wlbxdel((train,13),Pos),
          wlbxdel((train,14),Pos)
```

```
    )
  ),

  retract(sample(CNo,_,[X2,Y2,X3,Y3])),
  draw_sample((train, 9),
[X2,Y2,X3,Y3],notxorpen_rop ),
  draw_sample((train,10),
[X2,Y2,X3,Y3],notxorpen_rop ),

  (retractall(temp_sample(_,_,_)), true),
  forall( (sample(CNo2,_,_), CNo2 > CNo ),
      (
       NewCNo is CNo2 - 1,
       retract(sample(CNo2,ClassName,Span)),

assertz(temp_sample(NewCNo,ClassName,Span))
      )
    ),

  forall( temp_sample(SampleNo,ClassName1,Range),
      (
assertz(sample(SampleNo,ClassName1,Range) ),

retract(temp_sample(SampleNo,ClassName1,Range))
      )
    ),
  inform_samples(train,CNo)

train(classify) :-
  findall(SiteName, sample(_, SiteName, _),
SiteNames),
  findall(Span, sample(_, _, Span), Spans),
  sort(SiteNames, ClassNames),
  len(ClassNames, Classes),
  number_atom(Classes, ClassesAtom),
  sum_pixels(Spans, Pixels),
  number_atom(Pixels, PixelsAtom),

  cat(['pln_',ClassesAtom,'_',PixelsAtom,'.smp'],SampleFileName,_),
  tell(SampleFileName),
  Term1 = attached_to('pln_13.rsd'),
  write_canonical(Term1), write(.),nl,
  forall(sample(SNo, Name, Region),
      (
       Term =.. [sample_details, SNo, Name, Region],
       write_canonical(Term),write(.),nl
      )
    ),
  close(SampleFileName)

train(classify).

% inform the samples starting from CNo, by creating
required windows
inform_samples(MainWindow, CNo) :-
  forall( (sample(ClassNo, ClassName,
[X1,Y1,X2,Y2]), ClassNo >= CNo),
      (
       number_string(ClassNo, Class),
       (

write('('),write(X1),write(','),write(Y1),write(')'),write('-
'),

        write('('),write(X2),write(','),write(Y2),write(')')
       ) ~> RegionString,
       Count is (X2 - X1 + 1)*(Y2 - Y1 + 1),
       number_string(Count, CountStr),
       Pos is ClassNo - 1,
       wlbxadd((MainWindow,11),Pos,Class),
       wlbxadd((MainWindow,12),Pos,ClassName),
       wlbxadd((MainWindow,13),Pos,RegionString),
       wlbxadd((MainWindow,14),Pos,CountStr)
      )
    ),

  (
   (next_class(NextSNo),
    NextSNo > 1,
    EndPos is NextSNo - 2,
    wlbxsel((MainWindow,11),EndPos,1),
    List = [(MainWindow,12), (MainWindow,13),
(MainWindow,14)],
    sel_in_at(List,EndPos)
   )
   ;
   true
  ).

sum_pixels([],0).
sum_pixels([[X0,Y0,X1,Y1]|T], Sum) :-
  sum_pixels(T, PixelsInTail),
  Sum is PixelsInTail+(X1 - X0 + 1)*(Y1 - Y0 + 1).

draw_samples(_,[],_).
draw_samples(Win, [[X1,Y1,X2,Y2]|T],PenRop) :-
  % set_scale,
  gfx(
     (rop = stock(PenRop)
      -> polyline(X1,Y1,X2,Y1,X2,Y2,X1,Y2,X1,Y1)
     )
    ),
  draw_samples(Win, T, PenRop).

draw_sample(Win, [X1,Y1,X2,Y2],PenRop) :-
  gfx_begin(Win),
  % set_scale,
  gfx( ( rop = stock(PenRop)
         -> polyline(X1,Y1,X2,Y1,X2,Y2,X1,Y2,X1,Y1)
       )
    ),
  gfx_end(Win).

next_class(SNo) :-
  (
   sample(_,_,_) ->
   findall(X,sample(X,_,_), ClassList),
   sort(ClassList,Sorted),
   reverse(Sorted,[LastSNo|_]),
   SNo is LastSNo + 1
  )
  ;
  SNo is 1.

sel_in_at([],_).
sel_in_at([LbxH|LbxT], Pos) :-
  wlbxsel(LbxH, Pos, 1), !,
  sel_in_at(LbxT, Pos).

get_lbx_selection( Lbx, Selections ) :-
  wlbxfnd( Lbx, 0, _, NextItem ),
  get_lbx_selection( Lbx, 0, NextItem, [], Selections ).

get_lbx_selection( Lbx, Item, 0, SoFar, Sels ) :-
  wlbxsel( Lbx, Item, Sel ),
  get_selected( Sel, Lbx, Item, SoFar, Sels ).

get_lbx_selection( Lbx, Item, NextItem, SoFar, Sels ) :-
  wlbxsel( Lbx, Item, Sel ),
  get_selected( Sel, Lbx, Item, SoFar, NS ),
  wlbxfnd( Lbx, NextItem, _, AfterNextItem ),
  get_lbx_selection( Lbx, NextItem, AfterNextItem, NS, Sels ).

get_selected( 0, _, _, SoFar, SoFar ).
get_selected( 1, _, Item, SoFar, [Item|SoFar] ).
```

```
%%%%%%%%%%%%%%%%%%%%
% GUI UTILITIES %
%%%%%%%%%%%%%%%%%%%%

% set button states according to area selected

set_buttons :-
 (
  temp_select(X0,Y0,X0,Y0)
 -> set_buttons( 0, 0, 0, 0, 0, 1 )
 )
 ;
 set_buttons( 1, 0, 1, 0, 0, 1 ).

% set button states as given, and set focus to the
first enabled one

set_buttons( A, D, U, C, S, E ) :-
 wenable( (train,3), A ),
 wenable( (train,4), D ),
 wenable( (train,5), U ),
 wenable( (train,6), C ),
 wenable( (train,7), S ),
 wenable( (train,8), E ),

 forall( member( ID, [3,4,5,6,7,8] ),
      ( wstyle( (train,ID), Style ),
        Normal is Style /\ 16'fffffffe,
        wstyle( (train,ID), Normal )
      )
    ),
 ( member( ID, [3,4,5,6,7,8] ),
   wenable( (train,ID), State ),
   State = 1
 -> wfocus( (train,ID) ),
   wstyle( (train,ID), Style ),
   Hilite is Style \/ 16'00000001,
   wstyle( (train,ID), Hilite )
 ; true
 ).

% draw bitmap files

draw_paper1 :-
  gfx( bitmap(0,0,300,300,0,0,paper1) ).

draw_paper2 :-
  gfx( bitmap(0,0,300,300,0,0,paper2) ).

% display a list of message items

show_message( List ) :-
  forall( member( Item, List ),
       write( Item )
     ) -> String,
  wtext( (train,8), String ).

% set the origin and mapping for the grafix window

set_scale :-
  wsize( (train,9), _, _, Width, Height ),
  Border is 10,
  Xscale is int(Width - 2 * Border),
  Yscale is int(Height - 2 * Border),
  gfx_origin( Border, Border ),
  gfx_mapping( 400, 400, Xscale, Yscale ).

% selected area not to be overlapped

area_test(X,Y) :-
  forall(sample(_,_,[X1,Y1,X2,Y2]),
      (
       \+between(X,[X1,X2]);
       \+between(Y,[Y1,Y2])
      )
     ).

% In which sample SNo, (X,Y) is inside

area_test(X,Y,SNo) :-
  sample(SNo,_,[X1,Y1,X2,Y2]),
  between(X,[X1,X2]),
  between(Y,[Y1,Y2]).
area_test(_,_,0).

show_if_inside(X,Y,SNo) :-
  sample(SNo, Name, [X1,Y1,X2,Y2]),
  between(X,[X1,X2]),
  between(Y,[Y1,Y2]),
  last_mouse_pos(_,_,LastSNo),
  LastSNo =\= SNo,
  Sstyle = [ws_child,ws_visible,ws_border],
  number_string(SNo, SNoStr),
  Width is X2 - X1 + 1,
  Height is Y2 - Y1 + 1,
  (
   write(X1),write(','),write(Y1),write(' - '),
   write(Width),write('x'),write(Height)
  ) -> RegionStr,
  Count is (X2 - X1 + 1)*(Y2 - Y1 + 1),
  number_string(Count, CountStr),
  wccreate( (train,21), static, SNoStr,      260, 380,  40, 16,
Sstyle ),
  wccreate( (train,22), static, Name,        320, 380, 100, 16,
Sstyle),
  wccreate( (train,23), static, RegionStr,   440, 380, 100, 16,
Sstyle ),
  wccreate( (train,24), static, CountStr,    560, 380,  50, 16,
Sstyle ),
  retract(last_mouse_pos(_,_,_)),
  asserta(last_mouse_pos(X,Y,SNo)).

show_if_inside(X,Y,SNo) :-
  sample(SNo, _, [X1,Y1,X2,Y2]),
  between(X,[X1,X2]),
  between(Y,[Y1,Y2]),
  retract(last_mouse_pos(_,_,SNo)),
  asserta(last_mouse_pos(X,Y,SNo)).

show_if_inside(_,_,0) :-
  last_mouse_pos(X,_,SNo),
  integer(X),
  SNo > 0,
  retractall(last_mouse_pos(_,_,_)),
  asserta(last_mouse_pos(dummy,dummy,0)),
  wclose( (train,21) ),
  wclose( (train,22) ),
  wclose( (train,23) ),
  wclose( (train,24) ).

show_if_inside(_,_,0).

% If Num is within [Lower,Upper] inclusive, closed interval
between(Num,[Lower,Upper]) :-
  Num =< Upper,
  Num >= Lower.

% If Num is between (Lower,Upper) exclusive, open interval
between(Num,(Lower,Upper)) :-
  Num < Upper,
  Num > Lower.
```

E4. Program supporting matrix related functionality

```
/*======  =======  ========*\
|  Matrix  Related  Clauses  |
\*======  =======  ========*/

write_kmat([]) :- !.
write_kmat(MatK) :-
 MatK = [First|_],
 First = [Head|_],
 len(Head, Bands),
 band_names(Bands,BandNames),
 len(MatK, ClassesLessOne),
 Classes is ClassesLessOne + 1,
 findall(CNo, integer_bound(1,CNo,Classes),
ClassList),
 gen_pairs(ClassList, Pairs),
 write_kmat(BandNames, Pairs, MatK).

% gen_pairs(ClassList, Pairs)
gen_pairs([_|[]],[]) :- !.
gen_pairs([H|T], [PH|PT]) :-
 gen_pairs(H, T, PH),
 gen_pairs(T,PT).

gen_pairs(_,[],[]) :- !.
gen_pairs(H, [H1|T1],[H/H1|Tail]) :-
 gen_pairs(H, T1, Tail).

% band_names(Bands, List)
band_names(0,[]) :- !.
band_names(Bands, List) :-
 band_names(0, Bands, List).

band_names(Bands, Bands,[]) :- !.
band_names(OldNo, Bands, [H|T]) :-
 NewNo is OldNo + 1,
 number_string(NewNo, NumStr),
 cat(['Band',NumStr],BandStr,_),
 atom_string(H,BandStr),
 band_names(NewNo, Bands, T),
 !.

write_kmat(BandNames, Pairs, MatK) :-
 write('K-parameter Matrix'),
 nl,
 nl,
 tab(8),
 write_k_row(BandNames),
 nl,
 write_kmat(Pairs, MatK),nl.

write_kmat([],[]) :- !.
write_kmat([PH|PT],[KH|KT]) :-
 write_k_row(PH,KH),
 write_kmat(PT,KT).

write_k_row([],[]) :- !.
write_k_row([P|PT],[K|KT]) :-
 nl,
 write(P),
 tab(3),
 write_k_row(K),
 write_k_row(PT,KT).

write_k_row([]) :- !.
write_k_row([RH|RT]) :-
 float(RH),
 fwrite(f,5,-2,RH),
 tab(3),
 write_k_row(RT).

write_k_row([RH|RT]) :-
 integer(RH),
 fwrite(i,2,0,RH),
 tab(6),
 write_k_row(RT).

write_k_row([RH|RT]) :-
 atom(RH),
 write(RH),
 tab(3),
 write_k_row(RT).

% range_fuzzy(Pixel, MeanLessLower,UpperLessMean,Lower,
Upper,MinVect)
% MinVector calculated

range_fuzzy(_,[],_,_,[]) :- !.
range_fuzzy(Pixel, [LowerVector|LT],
[UpperVector|UT],[MiddleVector|MT], [MinH|MinT]) :-
 min_range_fuzzy_memship(Pixel, LowerVector,
UpperVector, MiddleVector, MinH),!,
 range_fuzzy(Pixel, LT, UT, MT, MinT).

min_range_fuzzy_memship([],_, _,_,0) :- !.
min_range_fuzzy_memship([PH|PT],[Lower|LT], [Upper|UT],
[Middle|MT], Min) :-
 min_range_fuzzy_memship(PT, LT, UT, MT, MinOfRest),!,
 Min is min(MinOfRest, (min(PH-Lower,Upper-PH))/Middle).

% mean_range_fuzzy(Pixel,
MeanLessLower,UpperLessMean,Lower, Upper,MinVect)
% MinVector calculated

mean_range_fuzzy(_,[],_,_,_,[]) :- !.
mean_range_fuzzy(Pixel, [MeanLessLowerVector|MLLT],
[UpperLessMeanVector|ULMT], [LowerVector|LT],
[UpperVector|UT],[MinH|MinT]) :-
 min_mean_range_fuzzy_memship(Pixel,
MeanLessLowerVector,UpperLessMeanVector, LowerVector,
UpperVector, MinH),!,
 mean_range_fuzzy(Pixel, MLLT, ULMT, LT, UT, MinT).

min_mean_range_fuzzy_memship([],_, _,_,_,0) :- !.
min_mean_range_fuzzy_memship([PH|PT],
[MeanLessLower|MLLT],[UpperLessMean|ULMT],[Lower|LT]
, [Upper|UT],Min) :-
 min_mean_range_fuzzy_memship(PT, MLLT, ULMT, LT, UT,
MinOfRest),!,
 Min is min(MinOfRest, min((PH-
Lower)*UpperLessMean,(Upper-PH)*MeanLessLower)).

% matrix_of_fuzzy_input(Pixel,
MeanVects,DenomMatrix,MinVect)
% MinVector calculated

matrix_of_fuzzy_input(_,[],_,[]) :- !.
matrix_of_fuzzy_input(Pixel, [MeanVectH|MeanVectT],
[NegInvDenomH|NegInvDenomT], [MinH|MinT]) :-
 min_explicit_fuzzy_memship(Pixel, MeanVectH,
NegInvDenomH, MinH),
 matrix_of_fuzzy_input(Pixel, MeanVectT, NegInvDenomT
MinT).

min_explicit_fuzzy_memship([], _,_,0) :- !.
min_explicit_fuzzy_memship([PH|PT],
[MH|MT],[NIDH|NIDT],Min) :-
 min_explicit_fuzzy_memship(PT, MT, NIDT, MinOfRest),
 Min is min(MinOfRest,(PH-MH)*(PH-MH)*NIDH).

% modulation(Beta, Bands, Lists, MeanVects,
NegInvDenomMatrix)
modulation(_,_,[],[],[]) :- !.
```

```
modulation(Beta, Bands, Lists, MeanVects,
NegInvDenomMatrix) :-
 append_times(Bands, 0, OldSumVector),
 means_freqs(Lists, MeanVects, Freqs,
NegTwiceVarMatrix,OldSumVector,SumVector),
 extend(Beta, SumVector, Freqs,
NegTwiceVarMatrix, NegInvDenomMatrix).

/*
extend(Beta,SumVector, ExtendMatrix,
NegOneByDenom)
Given Beta & ExtendMatrix,Mod Factors of classes in
each type of band
*/

extend(_,_,[],[],[]) :- !.
extend(Beta, SumVector, [FH|FT], [NTVH|NTVT],
[MFsHead|MFsTail]) :-
 sqr_modulation_factor(Beta,SumVector,FH,NTVH,
MFsHead),!,
 extend(Beta,SumVector, FT,NTVT,MFsTail).

sqr_modulation_factor(_,_,[],[],[]).
sqr_modulation_factor(1.25,[SH|ST],
[FHH|FHT],[NTVHH|NTVHT],[MFH|MFT]) :-
 MFH is 1/(NTVHH*((0.6*(FHH/SH) + 0.22)^2)), %
Linear relation at 1.25
 sqr_modulation_factor(1.25,ST, FHT,NTVHT,MFT).

sqr_modulation_factor(Beta,[SH|ST],
[FHH|FHT],[NTVHH|NTVHT], [MFH|MFT]) :-
 MFH is 1/(NTVHH*(( abs(ln((FHH/SH) + Beta)) )^2)),
 sqr_modulation_factor(Beta, ST, FHT, NTVHT,
MFT).

% BandWise mean_vector,FrequencyAtMeanVector,
NegTwiceOfVariance Vector

means_freqs([], [], [], [], SV, SV) :- !.
means_freqs([H|T], [MH|MT], [FH|FT],
[VarH|VarT],OldSumVector,SumVector) :-
 freq_at_mean(H,MH,FH,VarH),!,
 vector_sum(OldSumVector, FH, NewSumVector),
 means_freqs(T, MT,
FT,VarT,NewSumVector,SumVector).

% freq_at_mean(List, MeanVect, FreqAtMeanVect,
NegTwiceVar)
freq_at_mean([],[],[],[]) :- !.
freq_at_mean([H|T],[Mean|MT], [FreqAtMean|FT],
[TwiceVarH|TwiceVarT]) :-
 sort(H,List,[]), % sorts H to List with duplicates (if
any)
 count_table(List, _, 0, _, 0,0,0, Len, Sum, Sum1),
 !,
 Mean is Sum/Len,
 VarH is Sum1/Len - Mean*Mean,
 TwiceVarH is -2*VarH,

 FreqAtMean is Len/sqrt(6.28571428571429*VarH),
 freq_at_mean(T,MT,FT,TwiceVarT).

% cummfreq_about(IntMean, UniqueSorted,
CummFreq, CummFreqX, CummFreqY)
cummfreq_about(IntMean, [_,Y|_], [Cx,Cy|_], Cx, Cy)
:-
 Y > IntMean,!.
cummfreq_about(IntMean, [_,Y|T], [_,C2|CT], Cx, Cy)
:-
 cummfreq_about(IntMean, [Y|T],[C2|CT],Cx,Cy),!.

apply_op([],_,[]).
apply_op([MatH|MatT], Op, [RH|RT]) :-
 list(MatH),!,
 apply_op(MatH,Op,RH),
 apply_op(MatT,Op,RT).

apply_op([H|T], Op, [RH|RT]) :-
 Exp = Op((H)),
 RH is Exp,
 apply_op(T,Op,RT).

/* apply_op(Mat1, Mat2, Op, Result) apply operator Op
between
 corresponding elements of matrix Mat1, Mat2 */

apply_op([],[],_,[]).
apply_op([MatH|MatT],[MatH1|MatT1], Op,
[MatHead|MatTail]) :-
 list(MatH),!,
 apply_op(MatH,MatH1, Op, MatHead),
 apply_op(MatT,MatT1, Op, MatTail).

apply_op([H|T],[H1|T1], Op, [ResultHead|ResultTail]) :-
 \+list(H), !,
 Exp = Op((H),(H1)),
 ResultHead is Exp,
 apply_op(T,T1, Op, ResultTail).

% standard deviation & Mean
sd([],0,0).
sd(List, SD,Mean) :-
 variance(List, Var, Mean),
 SD is sqrt(Var).

% variance & Mean
variance([],0,0).
variance(List, Var,Mean) :-
 sort(List, UniqueSorted),
 count_table(List, UniqueSorted, Freq),
 len(List, N),
 transpose([Freq], TransFreq),
 matrix_mult([UniqueSorted], TransFreq, [[Sum]]),
 Mean is Sum/N,
 apply_op(UniqueSorted, UniqueSorted, *, Squared),
 matrix_mult([Squared], TransFreq, [[Sum1]]),
 Var is Sum1/N - Mean*Mean.

ranges([],[],[],[]) :- !.
ranges([H|T], [LH|LT], [UH|UT], [MH|MT]) :-
 range(H, LH, UH, MH),
 ranges(T, LT, UT, MT).

range([],[],[],[]) :- !.
range([List|MoreLists], [Lower|LT], [Upper|UT],
[Middle|MT]) :-
 sort(List,Sorted,[]),
 Sorted = [Lower|_],
 last_element(Sorted, Upper),
 Middle is (Lower + Upper)/2,
 range(MoreLists, LT, UT, MT).

means_ranges([],[],[],[],[]) :- !.
means_ranges([H|T], [MLLH|MLLT], [ULMH|ULMT],
[LH|LT], [UH|UT]) :-
 mean_range(H,MLLH, ULMH, LH, UH),
 means_ranges(T, MLLT, ULMT, LT, UT).

mean_range([],[],[],[],[]) :- !.
mean_range([List|MoreLists], [MeanLessLower|MLLT],
[UpperLessMean|ULMT],[Lower|LT], [Upper|UT]) :-
 sort(List, Sorted, []),
 count_table(Sorted, 0,0,0,Len,Sum,Upper),
 Mean is Sum/Len,
 Sorted = [Lower|_],
 MeanLessLower is Mean - Lower,
```

```
 UpperLessMean is Upper - Mean,
 mean_range(MoreLists, MLLT, ULMT, LT, UT).

% count_table(List, OldLen, OldSum, OldSum1, Len,
Sum, Sum1), Given Arg 1,3,5,6&7 gets others
% Mainly for overlap_fuzzy
count_table([],Len,Sum,Sum1,Len,Sum,Sum1) :- !.
count_table([H|T],OldLen,OldSum,OldSum1,Len,Su
m,Sum1) :-
 count(H, [H|T], 0, FH, NewList),
 NewLen is OldLen + FH,
 Term is H*FH,
 NewSum is OldSum + Term,
 NewSum1 is OldSum1 + H*Term,
 count_table(NewList,NewLen,NewSum,
NewSum1,Len,Sum,Sum1).

% count_table(List, OldLen, OldSum,Lower, Len,
Sum,Upper), Given Arg 1,3,5,6&7 gets others
% Mainly for range fuzzy
count_table([], Len, Sum, Upper, Len, Sum, Upper) :-
!.

count_table([H|T], OldLen, OldSum, _, Len, Sum,
Upper) :-
 count(H, [H|T], 0, FH, NewList),
 NewLen is OldLen + FH,
 NewSum is OldSum + H*FH,
 count_table(NewList,NewLen,NewSum,H, Len,Sum,
Upper).

% Mainly for explicit_fuzzy
% count_table(List, UniqueSorted, OldCumm,
CummFreq, OldLen, OldSum, OldSum1, Len, Sum,
Sum1), Given Arg 1,3,5,6&7 gets others

count_table([],[],_,[],Len,Sum,Sum1,Len,Sum,Sum1)
:- !.
count_table([H|T],[UH|UT],OldCumm,
[NewCumm|CummTail],OldLen,OldSum,OldSum1,L
en,Sum,Sum1) :-
 UH = H,
 count(H, [H|T], 0, FH, NewList),
 NewCumm is OldCumm + FH,
 NewLen is OldLen + FH,
 Term is H*FH,
 NewSum is OldSum + Term,
 NewSum1 is OldSum1 + H*Term,

count_table(NewList,UT,NewCumm,CummTail,NewL
en,NewSum, NewSum1,Len,Sum,Sum1).

% Beta Dist
count_table([],[],_,[],Len,Sum,Sum1,Len,Sum,Sum1,
Upper,Upper) :- !.
count_table([H|T],[UH|UT],OldCumm,
[NewCumm|CummTail],OldLen,OldSum,OldSum1,L
en,Sum,Sum1,_,Upper) :-
 UH = H,
 count(H, [H|T], 0, FH, NewList),
 NewCumm is OldCumm + FH,
 NewLen is OldLen + FH,
 Term is H*FH,
 NewSum is OldSum + Term,
 NewSum1 is OldSum1 + H*Term,

count_table(NewList,UT,NewCumm,CummTail,NewL
en,NewSum, NewSum1,Len,Sum,Sum1,H,Upper).

count(_, [], Count, Count, []) :- !.
count(Ele,[Ele],OldCount,Count,[]) :- !, Count is
OldCount + 1.
count(Ele,[Ele,Ele],OldCount,Count,[]) :- !, Count is
OldCount + 2.

count(Ele, [Ele,Ele|T], OldCount, Count,T) :-
 T = [X|_], X \== Ele, !,
 Count is OldCount + 2.

count(Ele, [Ele,Ele|T], OldCount, Count,T1) :-
 T = [Ele|_],
 NewCount is OldCount + 2,!,
 count(Ele, T, NewCount, Count,T1).

count(Ele, [Ele,X|T], OldCount, Count, [X|T]) :- X \== Ele,
Count is OldCount + 1,!.

count(_,[],0) :- !.
count(Ele, List, Count) :-
 count(Ele, List, 0, Count).

count(_,[],Count,Count) :- !.
count(X, [X|T], OldCount, Count) :-
 NewCount is OldCount + 1,
 !,
 count(X, T, NewCount, Count).

count(X, [_|T], OldCount, Count) :-
 count(X, T, OldCount, Count).

dist_matrices(MatrixList, Div, TransDiv, BhatDist, ResultStr)
:-
 ms(divergence(MatrixList,Div),MsDiv),
 TimeDiv is MsDiv/1000,

ms(transformed_divergence(MatrixList,TransDiv),MsTransDi
v),
 TimeTransDiv is MsTransDiv/1000,
 ms(bhat_dist(MatrixList,BhatDist),MsBhatDist),
 TimeBhatDist is MsBhatDist/1000,
 (
 write_matrix('Divergence Matrix', Div),
 nl,
 nl,
 write('Time Taken (Divergence) = '),
 write(TimeDiv),
 nl,
 nl,
 write_matrix('Transformed Divergence Matrix', TransDiv),
 nl,
 nl,
 write('Time Taken (Transformed Divergence) = '),
 write(TimeTransDiv),
 nl,
 nl,
 write_matrix('Bhat Distance Matrix', BhatDist),
 nl,
 nl,
 write('Time Taken (Bhat Dist) = '),
 write(TimeBhatDist)
 ) -> ResultStr,
 nl,
 write(ResultStr),
 nl.

% bhat_dist(ListofMatrices, BhattacharyyaDistanceMatrix)

bhat_dist(Matrices, BhatMat) :-
 matrices_unb_stats(Matrices, UnbCovList, InvUnbCovList,
MeanList,_),
 len(Matrices, Cols),
 StartRow is 1,
 bhat_dist(Cols, StartRow, UnbCovList, InvUnbCovList,
MeanList,BhatMat).

bhat_dist(Cols, StartRow, _, _, _,[]) :-
```

```
Cols < StartRow, !.

bhat_dist(Cols, StartRow, UnbCovList,
InvUnbCovList,
MeanList,[BhatMatHead|BhatMatTail]) :-
 StartRow =< Cols,
 StartCol is 1,
 nth_row_of_bhat_dist(Cols, StartRow, StartCol,
UnbCovList, InvUnbCovList,
MeanList,BhatMatHead),
 StartRowNext is StartRow + 1,
 bhat_dist(Cols, StartRowNext, UnbCovList,
InvUnbCovList, MeanList,BhatMatTail).

nth_row_of_bhat_dist(Cols, _, StartCol, _, _, _,[]) :-
 Cols < StartCol, !.

nth_row_of_bhat_dist(Cols, StartCol, StartCol,
UnbCovList, InvUnbCovList, MeanList,[0|BMHT]) :-
 StartColNext is StartCol + 1,
 nth_row_of_bhat_dist(Cols, StartCol, StartColNext,
UnbCovList, InvUnbCovList, MeanList,BMHT).

nth_row_of_bhat_dist(Cols, StartRow, StartCol,
UnbCovList, InvUnbCovList,
MeanList,[BMHH|BMHT]) :-
 StartCol =< Cols,
 member(UnbCovRow, UnbCovList, StartRow),
 member(UnbCovCol, UnbCovList, StartCol),
 member(MeanRow, MeanList, StartRow),
 member(MeanCol, MeanList, StartCol),
 matrix_difference([MeanRow], [MeanCol], MeanDiff),
 transpose(MeanDiff, TransposedMeanDiff),
 matrix_addition(UnbCovRow, UnbCovCol,
UnbCovSum),

matrix_scalar_mult(UnbCovSum,0.5,HalfOfUnbCovS
um),
 matrix_inverse(HalfOfUnbCovSum,_,_,_,_,
DetHalfOfUnbCovSum,InvHalfOfUnbCovSum),

matrix_mult(MeanDiff,InvHalfOfUnbCovSum,InterMa
t),
 matrix_mult(InterMat, TransposedMeanDiff,
Term1Mat),
 matrix_scalar_mult(Term1Mat,0.125,[[Term1]]),
 matrix_inverse(UnbCovRow,_,_,_,DetUnbCovRow,_),
 matrix_inverse(UnbCovCol,_,_,_,DetUnbCovCol,_),
 Term2 is
0.5*ln(abs(DetHalfOfUnbCovSum)/((DetUnbCovRow^
0.5) * (DetUnbCovCol^0.5) )),
 BMHH is Term1 + Term2,
 StartColNext is StartCol + 1,
 nth_row_of_bhat_dist(Cols, StartRow, StartColNext,
UnbCovList, InvUnbCovList, MeanList,BMHT).

/*
nth_row_of_bhat_dist(Cols, StartRow, StartCol,
UnbCovList, InvUnbCovList,
MeanList,[BMHH|BMHT]) :-
 StartCol =< Cols,
 member(UnbCovRow, UnbCovList, StartRow),
 member(UnbCovCol, UnbCovList, StartCol),
 member(MeanRow, MeanList, StartRow),
 member(MeanCol, MeanList, StartCol),
 matrix_difference([MeanRow], [MeanCol], MeanDiff),
 matrix_addition([MeanRow], [MeanCol], MeanSum),
 transpose(MeanDiff, TransposedMeanDiff),
 matrix_difference(UnbCovRow, UnbCovCol,
UnbCovDiff),
 matrix_addition(UnbCovRow, UnbCovCol,
UnbCovSum),

matrix_scalar_mult(UnbCovSum,0.5,HalfOfUnbCovSum),
matrix_mult(MeanSum,HalfOfUnbCovSum,InterMat),
matrix_mult(InterMat, TransposedMeanDiff, Term1Mat),
matrix_scalar_mult(Term1Mat,0.125,[[Term1]]),
matrix_inverse(UnbCovRow,_,_,_,DetUnbCovRow,_),
matrix_inverse(UnbCovCol,_,_,_,DetUnbCovCol,_),
matrix_scalar_mult(UnbCovDiff,0.5,HalfOfUnbCovDiff),
(
 (
  StartCol =:= StartRow, DetHalfOfUnbCovDiff = 0, Term2 =
0
 );
 (
  matrix_inverse(HalfOfUnbCovDiff,_,_,_,
DetHalfOfUnbCovDiff,_),
  Term2 is -
0.5*ln(abs(DetHalfOfUnbCovDiff)/((DetUnbCovRow^0.5) *
(DetUnbCovCol^0.5) ))
 )
),
BMHH is Term1 + Term2,
StartColNext is StartCol + 1,
nth_row_of_bhat_dist(Cols, StartRow, StartColNext,
UnbCovList, InvUnbCovList, MeanList,BMHT).
*/

% transformed_divergence(ListofMatrices, DivergenceMatrix)
transformed_divergence(Matrices, DivMat) :-
 matrices_unb_stats(Matrices, UnbCovList, InvUnbCovList,
MeanList,_),
 len(Matrices, Cols),
 StartRow is 1,
 transformed_divergence(Cols, StartRow, UnbCovList,
InvUnbCovList, MeanList,DivMat).

transformed_divergence(Cols, StartRow, _, _, _,[]) :-
 Cols < StartRow, !.

transformed_divergence(Cols, StartRow, UnbCovList,
InvUnbCovList, MeanList,[DivMatHead|DivMatTail]) :-
 StartRow =< Cols,
 StartCol is 1,
 nth_row_of_transformed_divergence(Cols, StartRow,
StartCol, UnbCovList, InvUnbCovList,
MeanList,DivMatHead),
 StartRowNext is StartRow + 1,
 transformed_divergence(Cols, StartRowNext, UnbCovList,
InvUnbCovList, MeanList,DivMatTail).

nth_row_of_transformed_divergence(Cols, _, StartCol, _, _,
_,[]) :-
 Cols < StartCol, !.

nth_row_of_transformed_divergence(Cols, StartRow,
StartCol, UnbCovList, InvUnbCovList,
MeanList,[DMHH|DMHT]) :-
 StartCol =< Cols,
 member(UnbCovRow, UnbCovList, StartRow),
 member(UnbCovCol, UnbCovList, StartCol),
 member(InvUnbCovRow, InvUnbCovList, StartRow),
 member(InvUnbCovCol, InvUnbCovList, StartCol),
 member(MeanRow, MeanList, StartRow),
 member(MeanCol, MeanList, StartCol),
 matrix_difference([MeanRow], [MeanCol], MeanDiff),
 transpose(MeanDiff, TransposedMeanDiff),
 matrix_mult(TransposedMeanDiff,MeanDiff,
MeanDiffMult),
 matrix_addition(InvUnbCovRow, InvUnbCovCol,
InvUnbCovSum),
 matrix_difference(InvUnbCovCol, InvUnbCovRow,
InvUnbCovDiff),
 matrix_difference(UnbCovRow, UnbCovCol, UnbCovDiff),
 matrix_mult(InvUnbCovSum, MeanDiffMult, Term1Matrix)
```

```
matrix_mult(UnbCovDiff, InvUnbCovDiff,
Term2Matrix),
trace(Term1Matrix, Term1Trace),
trace(Term2Matrix, Term2Trace),
J_RowCol is (Term1Trace + Term2Trace)/2,
DMHH is 100*(1 - aln(-J_RowCol/8) ),
StartColNext is StartCol + 1,
nth_row_of_transformed_divergence(Cols, StartRow,
StartColNext, UnbCovList, InvUnbCovList,
MeanList,DMHT).

% divergence(ListofMatrices, DivergenceMatrix)
divergence(Matrices, DivMat) :-
 matrices_unb_stats(Matrices, UnbCovList,
InvUnbCovList, MeanList,_),
 len(Matrices, Cols),
 StartRow is 1,
 divergence(Cols, StartRow, UnbCovList,
InvUnbCovList, MeanList,DivMat).

divergence(Cols, StartRow, _, _, _,[]) :-
 Cols < StartRow, !.

divergence(Cols, StartRow, UnbCovList,
InvUnbCovList, MeanList,[DivMatHead | DivMatTail])
:-
 StartRow =< Cols,
 StartCol is 1,
 nth_row_of_divergence(Cols, StartRow, StartCol,
UnbCovList, InvUnbCovList, MeanList,DivMatHead),
 StartRowNext is StartRow + 1,
 divergence(Cols, StartRowNext, UnbCovList,
InvUnbCovList, MeanList,DivMatTail).

nth_row_of_divergence(Cols, _, StartCol, _, _, _,[]) :-
 Cols < StartCol, !.

nth_row_of_divergence(Cols, StartRow, StartCol,
UnbCovList, InvUnbCovList,
MeanList,[DMHH | DMHT]) :-
 StartCol =< Cols,
 member(UnbCovRow, UnbCovList, StartRow),
 member(UnbCovCol, UnbCovList, StartCol),
 member(InvUnbCovRow, InvUnbCovList, StartRow),
 member(InvUnbCovCol, InvUnbCovList, StartCol),
 member(MeanRow, MeanList, StartRow),
 member(MeanCol, MeanList, StartCol),
 matrix_difference([MeanRow], [MeanCol], MeanDiff),
 transpose(MeanDiff, TransposedMeanDiff),
 matrix_mult(TransposedMeanDiff,MeanDiff,
MeanDiffMult),
 matrix_addition(InvUnbCovRow, InvUnbCovCol,
InvUnbCovSum),
 matrix_difference(InvUnbCovCol, InvUnbCovRow,
InvUnbCovDiff),
 matrix_difference(UnbCovRow, UnbCovCol,
UnbCovDiff),
 matrix_mult(InvUnbCovSum, MeanDiffMult,
Term1Matrix),
 matrix_mult(UnbCovDiff, InvUnbCovDiff,
Term2Matrix),
 trace(Term1Matrix, Term1Trace),
 trace(Term2Matrix, Term2Trace),
 DMHH is (Term1Trace + Term2Trace)/2,
 StartColNext is StartCol + 1,
 nth_row_of_divergence(Cols, StartRow,
StartColNext, UnbCovList, InvUnbCovList,
MeanList,DMHT).

matrices_unb_stats([],[],[],[],[]) :- !.
matrices_unb_stats([MatHead | MatTail],[UnbCovHea
d | UnbCovTail],[InvUnbCovHead | InvUnbCovTail],[Me
anHead | MeanTail],[UnbCorelHead | UnbCorelTail]) :-
 matrix_unb_stats(MatHead, MeanHead, UnbCovHead,
InvUnbCovHead,UnbCorelHead),

matrices_unb_stats(MatTail,UnbCovTail,InvUnbCovTail,Mea
nTail,UnbCorelTail).

matrices_stats([],[],[],[],[]) :- !.
matrices_stats([MatHead | MatTail],[CovHead | CovTail],[InvCo
vHead | InvCovTail],[MeanHead | MeanTail],[CorelHead | Corel
Tail]) :-
 matrix_stats(MatHead, MeanHead, CovHead,
InvCovHead,CorelHead),

matrices_stats(MatTail,CovTail,InvCovTail,MeanTail,CorelTai
l).

confusion_matrix(OldList, NewList, ConfusionMatrix, Khat)
:-
 error_matrix(OldList,NewList, ErrorMatrix),
 row_sum_vector(ErrorMatrix, RowSumVector),
 column_sum_vector(ErrorMatrix, ColSumVector),
 trace(ErrorMatrix, Trace),
 trace_vector(ErrorMatrix, TraceVector),
 append(ErrorMatrix, [ColSumVector], ColSumAppended),
 append(RowSumVector, [Trace], RowSumWithTrace),
 transpose(ColSumAppended, Transposed),
 append(Transposed,
[RowSumWithTrace],ConfusionTransposed),
 transpose(ConfusionTransposed,
ConfusionMatrixOnlyWithSums),
 accuracy(TraceVector, RowSumVector, RowWise),
 accuracy(TraceVector, ColSumVector, ColWise),
 sum_and_count(RowSumVector, TotalPixels,_),
 Overall is Trace*100/TotalPixels,
 len(RowWise,Rows),
 Cols = Rows,
 sum_and_count(RowWise, RowWiseSum,_),
 sum_and_count(ColWise, ColWiseSum,_),
 AvgRowWise is RowWiseSum/Rows,
 AvgColWise is ColWiseSum/Cols,
 append(RowWise, [AvgRowWise,Overall],NextCol),
 append(ColWise, [AvgColWise],NextRow),
 append(ConfusionMatrixOnlyWithSums,
[NextRow],CM_WithNextRow),
 transpose(CM_WithNextRow, TransposedWithNextRow),
 append(TransposedWithNextRow,
[NextCol],TransposedWithNextCol),
 transpose(TransposedWithNextCol,ConfusionMatrix),
 transpose([ColSumVector], TransposedColSum),

 matrix_mult([RowSumVector],TransposedColSum,[[SumOfPr
oducts]]),
 Khat is (TotalPixels*Trace -
SumOfProducts)/(TotalPixels*TotalPixels - SumOfProducts).

% Given the error_matrix, get ConfMat & KHAT

confusion_matrix(ErrorMatrix, ConfusionMatrix, Khat) :-
 row_sum_vector(ErrorMatrix, RowSumVector),
 column_sum_vector(ErrorMatrix, ColSumVector),
 trace(ErrorMatrix, Trace),
 trace_vector(ErrorMatrix, TraceVector),
 append(ErrorMatrix, [ColSumVector], ColSumAppended),
 append(RowSumVector, [Trace], RowSumWithTrace),
 transpose(ColSumAppended, Transposed),
 append(Transposed,
[RowSumWithTrace],ConfusionTransposed),
 transpose(ConfusionTransposed,
ConfusionMatrixOnlyWithSums),
 accuracy(TraceVector, RowSumVector, RowWise),
 accuracy(TraceVector, ColSumVector, ColWise),
 sum_and_count(RowSumVector, TotalPixels,_),
 Overall is Trace*100/TotalPixels,
```

```
len(RowWise,Rows),
Cols = Rows,
sum_and_count(RowWise, RowWiseSum,_),
sum_and_count(ColWise, ColWiseSum,_),
AvgRowWise is RowWiseSum/Rows,
AvgColWise is ColWiseSum/Cols,
append(RowWise, [AvgRowWise,Overall],NextCol),
append(ColWise, [AvgColWise],NextRow),
append(ConfusionMatrixOnlyWithSums,
[NextRow],CM_WithNextRow),
transpose(CM_WithNextRow,
TransposedWithNextRow),
append(TransposedWithNextRow,
[NextCol],TransposedWithNextCol),

transpose(TransposedWithNextCol,ConfusionMatrix),
transpose([ColSumVector], TransposedColSum),

matrix_mult([RowSumVector],TransposedColSum,[[S
umOfProducts]]),
Khat is (TotalPixels*Trace -
SumOfProducts)/(TotalPixels*TotalPixels -
SumOfProducts).

accuracy([],[],[]) :- !.
accuracy([TVH|TVT],[SH|ST],[DH|DT]) :-
((SH>0,DH is TVH*100/SH);DH=0),
accuracy(TVT, ST, DT).

error_matrix(OldList,NewList, Matrix) :-
max(OldList, MaxOld),
max(NewList, MaxNew),
Cols is max(MaxNew, MaxOld),
error_matrix(Cols, 1, OldList, NewList, Matrix).

error_matrix(Cols, StartRow, _,_,[]) :-
Cols < StartRow, !.

error_matrix(Cols, StartRow, OldList, NewList,
[CRH|CRT]) :-
StartRow =< Cols,
nth_row_of_error_matrix(Cols, StartRow, 1,
OldList,NewList, CRH),
StartRowNext is StartRow + 1,
error_matrix(Cols, StartRowNext, OldList,NewList,
CRT).

nth_row_of_error_matrix(Cols,_,StartCol, _,_, []) :-
Cols < StartCol, !.

nth_row_of_error_matrix(Cols, StartRow, StartCol,
OldList, NewList, [CRHH|CRHT]) :-
StartCol =< Cols,
error_matrix_element(OldList,NewList, StartRow,
StartCol,Count),
CRHH is Count,
StartColNext is StartCol + 1,

nth_row_of_error_matrix(Cols,StartRow,StartColNext,
OldList,NewList,CRHT).

error_matrix_element([],[],_,_,0) :- !.
error_matrix_element([Row|OldTail],[Col|NewTail],
Row, Col, Count) :-
error_matrix_element(OldTail, NewTail, Row, Col,
CountTail),
Count is CountTail + 1.

error_matrix_element([_|OldTail],[_|NewTail], Row,
Col, Count) :-
error_matrix_element(OldTail, NewTail, Row, Col,
CountTail),
Count is CountTail.

% MeanVector, Unbiased Covariance, Inverse of Unbiased
Covariance, Unbiased Corelation

matrix_unb_stats([],[],[],[],[]) :- !.
matrix_unb_stats(Matrix, MeanVector, UnbCov, InvUnbCov,
UnbCorel) :-
matrix_stats(Matrix, MeanVector, Cov),
len(Matrix, N),
N > 1,
Scalar is 1/(N-1),
matrix_scalar_mult(Cov, Scalar, UnbCov),
matrix_size(Matrix,_,Cols),
corelation(Cols, 1, UnbCov, UnbCorel),
matrix_inverse(UnbCov,_,_,_,_,InvUnbCov).

% MeanVector, Covariance, Inverse of Covariance,
Corelation

matrix_stats([],[],[],[],[]) :- !.
matrix_stats(Matrix, MeanVector, Cov, InvCov, Corel) :-
matrix_stats(Matrix, MeanVector, Cov),
matrix_size(Matrix,_,Cols),
corelation(Cols, 1, Cov, Corel),
matrix_inverse(Cov,_,_,_,_,InvCov).

matrix_stats(Matrix, MeanVector, Covariance) :-
mean_vector(Matrix,MeanVector),
matrix_scalar_mult([MeanVector], -1, NegativeMeanVector),
covariance(Matrix, NegativeMeanVector, Covariance).

matrix_stats(Matrix, MeanVector, Covariance, Corelation) :-
mean_vector(Matrix,MeanVector),
matrix_scalar_mult([MeanVector], -1, NegativeMeanVector),
covariance(Matrix, NegativeMeanVector, Covariance),
matrix_size(Matrix,_,Cols),
corelation(Cols, 1, Covariance, Corelation).

corelation(Cols, StartRow, _,[]) :-
Cols < StartRow, !.

corelation(Cols, StartRow, Covariance,[CRH|CRT]) :-
StartRow =< Cols,
nth_row_of_corel(Cols, StartRow, 1, Covariance, CRH),
StartRowNext is StartRow + 1,
corelation(Cols, StartRowNext, Covariance,CRT).

nth_row_of_corel(Cols,_,StartCol, _, []) :-
Cols < StartCol, !.

nth_row_of_corel(Cols, StartRow, StartCol, Covariance,
[CRHH|CRHT]) :-
StartCol =< Cols,
matrix_element(Covariance, StartRow, StartCol, VarIJ),
matrix_element(Covariance, StartRow, StartRow, VarII),
matrix_element(Covariance, StartCol, StartCol, VarJJ),
CRHH is VarIJ/sqrt(VarII*VarJJ),
StartColNext is StartCol + 1,

nth_row_of_corel(Cols,StartRow,StartColNext,Covariance,CR
HT).

covariance([],_,[]) :- !.
covariance([MH|MT], NegativeMeanVector, Covariance) :-
matrix_addition([MH],NegativeMeanVector, Diff),
transpose(Diff, Transposed),
matrix_mult(Transposed, Diff, Product),
covariance(MT,NegativeMeanVector, Product1),
matrix_addition(Product, Product1, Covariance).

fuzzy_covariance([],_,_,[]) :- !.
fuzzy_covariance([MH|MT], [MemShipHead|MemShipTail],
NegativeMeanVector, Covariance) :-
```

```
matrix_addition([MH],NegativeMeanVector, Diff),
transpose(Diff, Transposed),
matrix_mult(Transposed, Diff, Product),
matrix_scalar_mult(Product,MemShipHead,
WeightedProduct),
fuzzy_covariance(MT,MemShipTail,
NegativeMeanVector, ProductTail),
matrix_addition(WeightedProduct, ProductTail,
Covariance).

matrix_determinant([],[],[],[],0) :- !.
matrix_determinant([[X]],[[X]],[[X]],[[C]],X) :-
 X =\= 0, !,
 C is -X.

matrix_determinant(Matrix,PowerList,TraceList,CoefL
ist,Determinant) :-
 matrix_size(Matrix,M,M),
 matrix_power(Matrix,1,M,PowerList),
 trace(PowerList,TraceList),
 coef_vector(M,1,TraceList,_, CoefList),
 last_element(CoefList, LastEle),
 Modulus is M mod 2,
 (Modulus = 0 -> Determinant is LastEle ;
  Determinant is (-1)*LastEle).

matrix_inverse([],[],[],[],0,[]) :- !.
matrix_inverse([[X]],[[X]],[[X]],[[C]],X,[[InvX]]) :-
 X =\= 0, !,
 InvX is 1/X,
 C is -X.

matrix_inverse(Matrix,PowerList,TraceList,CoefList,D
eterminant,Inverse) :-
 matrix_size(Matrix,M,M),
 matrix_power(Matrix,1,M,PowerList),
 trace(PowerList,TraceList),
 coef_vector(M,1,TraceList,_, CoefList),
 last_element(CoefList, LastEle),
 Modulus is M mod 2,
 (Modulus = 0 -> Determinant is LastEle ;
  Determinant is (-1)*LastEle ),
 matrix_inversion(Matrix, PowerList, CoefList,
Inverse).

matrix_inversion(Matrix, PowerList, CoefList, Inverse)
:-
 matrix_size(Matrix,M,M),
 sum_scalar_product(M,1,PowerList,CoefList,Sum),
 Pos is M - 1,
 member(Mat,PowerList,Pos),
 member(Coef,CoefList, Pos),
 unit_matrix(M, UnitMatrix),
 matrix_scalar_mult(UnitMatrix, Coef, UMScaled),
 matrix_addition(UMScaled, Mat, RestOfSum),
 matrix_addition(RestOfSum, Sum, Cofactor),
 last_element(CoefList, LastCoef),
 Constant is -1/LastCoef,
 matrix_scalar_mult(Cofactor, Constant, Inverse).

sum_scalar_product(Order, Start, _, _, []) :-
 Start > Order - 2, !.

sum_scalar_product(Order, Start, PowerList,
CoefList, Sum) :-
 Start =< Order - 2,
 member(Ele,CoefList, Start),
 Pos is Order - Start - 1,
 member(Mat,PowerList,Pos),
 matrix_scalar_mult(Mat,Ele,MatScaled),
 Start1 is Start + 1,
 sum_scalar_product(Order, Start1, PowerList,
 CoefList, Sum1),
 matrix_addition(Sum1,MatScaled,Sum).

% coef_vector(Order, Start, TraceList, OldCoefList, CoefList)
coef_vector(Order, Start,_,_,[]) :-
 Start > Order, !.

coef_vector(Order, 1, TraceList, OldCoefList, [Coef|CoefList])
:-
 nth_coef(1, TraceList,OldCoefList, Coef),!,
 coef_vector(Order, 2, TraceList, OldCoefList,CoefList).

coef_vector(Order, Start, TraceList, OldCoefList,
[Coef|NewCoefList1]) :-
 nth_coef(Start, TraceList,OldCoefList, Coef),
 append(OldCoefList, [Coef], NewCoefList),
 StartNext is Start + 1,
 coef_vector(Order, StartNext, TraceList,
NewCoefList,NewCoefList1).

% nt_coef(Nth, TraceList, OldCoefList, Coef) :-
nth_coef(1, [TH|_], [H],H) :-
 H is (-1)*TH, !.

nth_coef(N, TraceList, OldCoefs, Coef) :-
 N > 1,
 N1 is N - 1,
 sum_till(N1, TraceList, OldCoefs, Sum),
 member(S, TraceList, N),
 Coef is (-1/N)*(Sum + S).

sum_till(0,_,_,0) :- !.
sum_till(0,_,[],0) :- !.
sum_till(1, [TH|_], [OCH|_], Sum) :-
 Sum is TH*OCH,!.

sum_till(N, [TH|TT], OldCoef, Sum) :-
 N > 1,
 N1 is N - 1,
 member(OCH, OldCoef, N),
 sum_till(N1, TT, OldCoef, Sum1),
 Sum is Sum1 + TH*OCH.

% matrix_scalar_mult(Matric, Constant, Result)
matrix_scalar_mult([],_,[]) :- !.
matrix_scalar_mult([MH|MT], Constant, [SPH|SPT]) :-
 multiply_each_element(MH, Constant, SPH),!,
 matrix_scalar_mult(MT, Constant, SPT).

rows_scalar_mult([],_,[]) :- !.
rows_scalar_mult([Row|RT], [Scalar|ST], [NewRow|NRT]) :-
 multiply_each_element(Row, Scalar, NewRow),!,
 rows_scalar_mult(RT, ST, NRT).

rows_scalar_div([],_,[]) :- !.
rows_scalar_div([Row|RT], [Scalar|ST], [NewRow|NRT]) :-
 multiply_each_element(Row, (1/Scalar), NewRow),
 rows_scalar_div(RT, ST, NRT).

multiply_each_element([], _, []) :- !.
multiply_each_element([H|T], Constant, [RH|RT]) :-
 RH is H*Constant,
 multiply_each_element(T, Constant, RT).

matrix_difference(From, Subtract, Diff) :-
 matrix_scalar_mult(Subtract,-1, Negative),
 matrix_addition(From, Negative, Diff).

matrix_addition([],[],[]) :- !.
matrix_addition([],M,M) :- !.
matrix_addition(M,[],M) :- !.
matrix_addition(First, Second, []) :-
 matrix_size(First,R,C),
 matrix_size(Second,R1,C1),
```

```
(R =\= R1;C =\=C1), !.

matrix_addition([FH|FT], [SH|ST], [MH|MT]) :-
 vector_sum(FH, SH, MH),
 matrix_addition(FT, ST, MT).

vector_sum([],[],[]) :- !.
vector_sum([FH|FT], [SH|ST], [H|T]) :-
 H is FH + SH,
 vector_sum(FT, ST, T).

trace([],[]) :- !.
trace([MH|MT],[TH|TT]) :-
 trace(MH,TH),
 trace(MT,TT),!.

trace(Matrix,0) :-
 matrix_size(Matrix, M, N),
 M =\= N,!.

trace(Matrix,Trace) :-
 matrix_size(Matrix, M, M),
 trace(Matrix,M,Trace).

trace([],_,0) :- !.
trace(_, 0, 0) :- !.
trace(Matrix, Order,Trace) :-
 member(List, Matrix, Order),
 member(Ele, List, Order),
 Order1 is Order - 1,
 trace(Matrix, Order1, Trace1),
 Trace is Trace1 + Ele.

trace_vector(Matrix,[]) :-
 matrix_size(Matrix, M, N),
 M =\= N,!.

trace_vector(Matrix,TraceVector) :-
 matrix_size(Matrix, M, M),
 trace_vector(Matrix,M,ReversedTraceVector),
 reverse(ReversedTraceVector,TraceVector).

trace_vector([],_,[]) :- !.
trace_vector(_, 0, []) :- !.
trace_vector(Matrix, Order,ReversedTraceVector) :-
 member(List, Matrix, Order),
 member(Ele, List, Order),
 Order1 is Order - 1,
 trace_vector(Matrix, Order1,
ReversedTraceVectorTail),
 append([Ele],
ReversedTraceVectorTail,ReversedTraceVector).

% equal_matrix(Order, AllElementsEqualTo,
SquareMatrix).

all_equal_matrix(0,_,[]) :- !.
all_equal_matrix(Order,Value,Matrix) :-
 all_equal_matrix(Order,Order, Value,Matrix).

% equal_matrix(Rows,Cols, AllElementsEqualTo,
Matrix).

all_equal_matrix(0,0,_,[]) :- !.
all_equal_matrix(Rows,Cols, Value,Matrix) :-
 all_equal_vectors(Rows, Cols,Value,1,Matrix).

all_equal_vectors(Rows, _, _, Start, []) :-
 Start > Rows,!.

all_equal_vectors(Rows, Cols, Value, Start,
[AEVH|AEVT]) :-
 append_times(Cols,Value,AEVH),
 Start1 is Start + 1,
 all_equal_vectors(Rows, Cols, Value, Start1, AEVT).

unit_matrix(0,[]) :- !.
unit_matrix(Order, Matrix) :-
 all_unit_vectors(Order, 1, Matrix).

all_unit_vectors(Order, Start, []) :-
 Start > Order,!.

all_unit_vectors(Order, Start, [UVH|UVT]) :-
 nth_unit_vector(Order, Start, UVH),
 Start1 is Start + 1,
 all_unit_vectors(Order, Start1, UVT).

nth_unit_vector(Order, N, []) :-
 N > Order, !.

nth_unit_vector(Order, N, UnitVector) :-
 N = 1,!,
 N1 is Order - 1,
 append_times(N1,0,Right),
 append([1],Right, UnitVector).

nth_unit_vector(Order, N, UnitVector) :-
 N1 is N - 1,
 append_times(N1, 0, OldLeft),
 append(OldLeft, [1], Left),
 N2 is Order - N,
 append_times(N2,0,Right),
 append(Left,Right, UnitVector).

append_times(0,_,[]) :- !.
append_times(Times, Ele, [Ele|Tail]) :-
 Times1 is Times - 1,
 append_times(Times1, Ele, Tail).

matrix_power(_,From,To,[]) :-
 From > To, !.
matrix_power(Matrix, From, To, [MH|MT]) :-
 matrix_power(Matrix, From, MH),
 Next is From + 1,
 matrix_power(Matrix, Next,To, MT).

matrix_power(Matrix,_,_) :-
 matrix_size(Matrix, R, C),
 R =\= C, !.

matrix_power(Matrix,1,Matrix) :- !.
matrix_power(Matrix,Power,Result) :-
 Power > 1,
 Power1 is Power - 1,
 matrix_power(Matrix, Power1, Result1),
 matrix_mult(Result1, Matrix, Result).

matrix_mult(First, Second, Matrix) :-
 mult_compatible(First, Second),
 transpose(Second, Transposed),
 mult_process(First, Transposed, Matrix).

mult_process([],_,[]).
mult_process([FH|FT], Second, [Row|MT]) :-
 all_vector_mult(FH, Second, Row),
 mult_process(FT, Second, MT).

all_vector_mult(_,[],[]) :- !.
all_vector_mult(RowVect, [MH|MT], [RH|RT]) :-
 vector_mult(RowVect,MH,RH),
 all_vector_mult(RowVect, MT, RT).

mult_compatible(First, Second) :-
 matrix_size(First, _, N),
 matrix_size(Second,N, _).
```

```
vector_mult([],[],0).
vector_mult([H1|T1], [H2|T2], Product) :-
 vector_mult(T1,T2,TailProduct),
 Product is H1*H2 + TailProduct.

% Standard Deviation Vector
sd_vector(CovarianceMatrix, StandardDevVector)
% From Covariance Matrix brings Standard
Deviation Vector

sd_vector(Cov, SD) :-
 StartRow is 1,
 sd_vector(StartRow, Cov, SD).

sd_vector(_,[],[]) :- !.

sd_vector(Row, [CH|CT], [SDH|SDT]) :-
 member(Var, CH, Row),
 SDH is sqrt(Var),
 NextRow is Row + 1,
 sd_vector(NextRow, CT, SDT).

sum_vector(Matrix, SumVector) :-
 column_sum_vector(Matrix,SumVector).

column_sum_vector([],[]) :- !.
column_sum_vector(Matrix,SumVector) :-
 transpose(Matrix, Transposed),
 row_sum_vector(Transposed,SumVector).

row_sum_vector([],[]) :- !.
row_sum_vector([MH|MT],[SH|ST]) :-
 sum_and_count(MH,SH,_),
 row_sum_vector(MT,ST).

%mean_vector(OfMatrix, HasMeanVector)
mean_vector([],[]).
mean_vector(Matrix, Vector) :-
 transpose(Matrix, Transposed),
 means(Transposed,Vector).

%means(Matrix, ListOfMeans)
means([],[]).
means([TH|TT], [MH|MT]) :-
 sum_and_count(TH, Sum, Count),
 MH is Sum/Count,
 means(TT,MT).

% matrix_size(Matrix,HasRows, HasCols)
matrix_size([], 0, 0).
matrix_size(Matrix, Rows, Columns) :-
 length(Matrix, Rows),
 member(FirstRow, Matrix, 1),
 length(FirstRow, Columns).

matrix_element(Matrix, RowIdx, ColIdx, Ele) :-
 member(Row, Matrix, RowIdx),
 member(Ele, Row, ColIdx).

transpose([],[]).
transpose(Matrix, Transposed) :-
 gen_empty(Matrix, OldTrans),
 reversed_trans(Matrix, OldTrans, TransReversed),
 reverse_all(TransReversed, Transposed).

reverse_all([],[]).
reverse_all([H|T], [RH|RT]) :-
 reverse(H,RH),
 reverse_all(T, RT).

gen_empty([],[]).
gen_empty([MH|_],OldTrans) :-
 insert_empty(MH, [], OldTrans).

insert_empty([], O, O).
insert_empty([_|FT],Old, OldTrans) :-
 append(Old, [[]], Old1),
 insert_empty(FT,Old1, OldTrans).

reversed_trans([],T,T).
reversed_trans([FR|RR],Old, Transposed) :-
 insert_row(FR, Old, Trans),
 reversed_trans(RR, Trans, Transposed).

insert_row([],TR,TR).
insert_row([RH|RT],[OTH|OTT], [[RH|OTH]|TT]) :-
 insert_row(RT, OTT, TT).

last_element([H],H) :- !.
last_element([_|T],Ele) :-
 last_element(T,Ele).

sum_and_count([],0,0).
sum_and_count([H|T], Sum, Count) :-
 sum_and_count(T, SumTail, CountTail),
 Sum is SumTail + H,
 Count is CountTail + 1.

max([],0) :- !.
max([X],X) :- !.
max([X|Tail], Max) :-
 max(Tail, Max),
 Max >= X,!.
max([X|_], X).

min([],0) :- !.
min([X],X) :- !.
min([X|Tail], Min) :-
 min(Tail, Min),
 Min =< X,!.
min([X|_], X).

% max_at([],0,0) :- !.
max_at([X],X,1) :- !.
max_at([X|Tail], Max, Pos) :-
 max_at(Tail, Max, Pos1),
 Max >= X,!,
 Pos is Pos1 + 1.
max_at([X|_], X,1).

min_at([],0,0) :- !.
min_at([X],X,1) :- !.
min_at([X|Tail], Min, Pos) :-
 min_at(Tail, Min, Pos1),
 Min =< X,!,
 Pos is Pos1 + 1.
min_at([X|_], X,1).

maxlist([X],X).
maxlist([X,Y|Rest], Max) :-
 maxlist([Y|Rest], MaxRest),!,
 Max is max(X,MaxRest).

minlist([X],X).
minlist([X,Y|Rest], Min) :-
 minlist([Y|Rest], MinRest),!,
 Min is min(X,MinRest).

write_class_stats(Data, ClassList) :-
 sample_classes(Classes),
 write_class_stats(1, Classes, Data,ClassList).

write_class_stats(ClassNo, Classes, _, _) :-
 ClassNo > Classes,!.
```

```
write_class_stats(ClassNo, Classes,Data,ClassList) :-
 class_stats(ClassNo, Data, ClassList, MeanVector,
Cov, _, CovInv),
 (write('For the Class No.: '),
  write(ClassNo)) -> HeadMsg,
 write_matrices(HeadMsg, ['Mean', 'Cov', 'Inverse of
Cov'],[MeanVector, Cov, CovInv]),
 ClassNext is ClassNo + 1,

write_class_stats(ClassNext,Classes,Data,ClassList).

write_sample_stats :-
 sample_classes(Classes),
 write_sample_stats(1, Classes).

write_sample_stats(ClassNo, Classes) :-
 ClassNo > Classes,!.

write_sample_stats(ClassNo, Classes) :-
 sample_stats(ClassNo, MeanVector, Cov, _, CovInv),
 (write('For Training Sample No.: '),
  write(ClassNo)) -> HeadMsg,
 write_matrices(HeadMsg, ['Mean', 'Cov', 'Inverse of
Cov'],[MeanVector, Cov, CovInv]),
 ClassNext is ClassNo + 1,
 write_sample_stats(ClassNext,Classes).

write_matrices(HeadMsg, MsgList, MatList) :-
 nl,nl,
 write(HeadMsg),
 nl,nl,
 write_matrices(MsgList,MatList).

write_matrices([],[]).
write_matrices([MsgHead|MsgTail],[MatHead|MatTai
l]) :-
 write_matrix(MsgHead, MatHead),
 write_matrices(MsgTail, MatTail).

write_matrix(Msg, Matrix) :-
 nl,
 write(Msg),
 nl,nl,
 write_matrix(Matrix).

write_matrix([]) :- !.
write_matrix([MH|MT]) :-
 list(MH),
 nl,
 write_row(MH),
 write_matrix(MT).

write_matrix([MH|MT]) :-
 \+list(MH),
 float(MH),
 fwrite(f,10,-2,MH),
 tab(1),
 write_matrix(MT).

write_matrix([MH|MT]) :-
 \+list(MH),
 integer(MH),
 fwrite(i,7,0,MH),
 tab(4),
 write_matrix(MT).

write_row([]) :- !.
write_row([RH|RT]) :-
 float(RH),
 fwrite(f,10,-2,RH),
 tab(1),
 write_row(RT).

write_row([RH|RT]) :-
 integer(RH),
 fwrite(i,7,0,RH),
 tab(4),
 write_row(RT).
```

E5. Program supporting statistics calculation functionality

```
/* Statistical Utilities */

histogram(List, UniqueSorted, LowestLevel, HighestLevel,
Frequencies, LowestFreq, HighestFreq, CummFreq) :-
 sort(List, LowestLevel, HighestLevel, UniqueSorted),
 count_table(List, UniqueSorted, Frequencies),
 sort(Frequencies, LowestFreq, HighestFreq,_),
 OldSum is 0,
 cumm_sum(Frequencies,OldSum, CummFreq).

means([],0,0,0).
means([H],H,H,H).
means(List, Mean, Median, Mode) :-
 sum_list(List,Sum),
 sort(List, _, _, UniqueSorted),
 count_table(List,UniqueSorted, Frequencies),
 OldSum is 0,
 cumm_sum(Frequencies,OldSum, Cumm),
 lastn(1,Cumm, [Len]),
 Mean is Sum/Len,
 N_Half is Len/2,
 median_class(N_Half,Cumm,Pos),
 member(Median, UniqueSorted,Pos),
 sort(Frequencies, F_Sorted,[]),
 lastn(1, F_Sorted, [ModalFreqPos]),
 member(ModalFreqPos, Frequencies, ModePos),
 member(Mode, UniqueSorted,ModePos).

mode([],0,0).
mode([H],H,1).
mode(List, Mode, ModalFreq) :-
 sort(List, _, _, UniqueSorted),
 count_table(List,UniqueSorted, Frequencies),
 sort(Frequencies, F_Sorted,[]),
 lastn(1,F_Sorted,[ModalFreq]),
 member(ModalFreq, Frequencies, ModePos),
 member(Mode, UniqueSorted,ModePos),!.

% for overlap_fuzzy, MeansVects is size Classes*Bands
% means_and_sds(ListOfLs, MeanVects, SDs)
means_and_sds([],[],[]) :- !.
means_and_sds([H|T],[MH|MT], [SDH|SDT]) :-
 mean_and_sd(H,MH,SDH),
 means_and_sds(T,MT,SDT).

% mean_and_sd(ListOfLists, MeanVect, SD)
mean_and_sd([],[],[]) :- !.
mean_and_sd([H|T],[Mean|MT], [StdDevH|SDT]) :-
 sort(H,List,[]), % sorts H to List with duplicates (if any)
 count_table(List, 0,0,0, Len, Sum, Sum1),
 !,
 Mean is Sum/Len,
 StdDevH is sqrt(Sum1/Len - Mean*Mean),
 mean_and_sd(T,MT,SDT).

median(UniqueSorted, Cumm, Median) :-
 append(_,[Len],Cumm),
 N_Half is Len/2,
 median_class(N_Half,Cumm,Pos),
 member(Median, UniqueSorted,Pos).

median_class(N_Half, [H|_], 1) :-
 N_Half < H,!.
```

```
median_class(N_Half, [_|T], Pos) :-
 median_class(N_Half, T,Pos1),
 Pos is Pos1 + 1.

% distance_weighted_membership(ListOfPixels,
Means, MemShips)
distance_weighted_membership(_,[], _, []).
distance_weighted_membership(ClassNo,
[Pixel|RestOfPixels], Means, [MemShips|MT]) :-
 dist_sum(Pixel, Means, Distances, DistSum),
 memship(Distances, DistSum, MemShipsList),
 member(MemShips,MemShipsList,ClassNo),
 distance_weighted_membership(ClassNo,
RestOfPixels, Means, MT).

% memship(Distances, DistSum, MemShip).
memship([],_,[]).
memship([DH|DT], DistSum, [MH|MT]) :-
 MH is 1 - (DH/DistSum),
 memship(DT,DistSum, MT).

% dist_sum(Pixel, Means, Distances, DistSum)
dist_sum(_, [], [], 0).
dist_sum(Pixel, [MH|MT], [DH|DT], DistSum) :-
 distance(Pixel, MH, _, DH),
 dist_sum(Pixel, MT, DT, DistSumOfTail),
 DistSum is DistSumOfTail + DH.

% distance(Pixel, Mean, DistSqr, Distance)
distance([],[],0,0).
distance([PH|PT], [MH|MT], DistSqr, Distance) :-
 DH is (PH-MH)*(PH-MH),
 distance(PT,MT, DistSqrTail, _),
 DistSqr is DH + DistSqrTail,
 Distance is sqrt(DistSqr).

cumm_sum(List, CummList) :-
 cumm_sum(List, 0, CummList).

cumm_sum([],_,[]).
cumm_sum([H|T],OldSum, [NewSum|SumTail]) :-
 NewSum is OldSum + H,
 cumm_sum(T,NewSum,SumTail).

sum_list(List, Sum) :-
 sum_list(List, 0, Sum).

sum_list([],Sum,Sum).
sum_list([H|T], OldSum, Sum) :-
 NewSum is OldSum+H,
 sum_list(T,NewSum, Sum).

maxes_at([],[],[]) :- !.
maxes_at([X|Tail], [Max|MT], [Pos|PosT]) :-
 max_at(X, Max, Pos),
 maxes_at(Tail, MT, PosT).

mins_at([],[],[]) :- !.
mins_at([X|Tail], [Min|MT], [Pos|PosT]) :-
 min_at(X, Min, Pos),
 mins_at(Tail, MT, PosT).

% sort(List, LowerBound, UpperBound,
SortedUniqueElements)

sort([],0,0,[]) :- !.
sort(List,Lower,Upper,[Lower|SortedTail]) :-
 sort(List,[Lower|SortedTail]),
 last(Upper,[Lower|SortedTail]).

count_table(_,[],[]) :- !.
count_table(List,[H|T],[FH|FT]) :-
 count(H,List,FH),
 count_table(List,T,FT).


sublist(_,_,[],[]).
sublist(StaPos,NoOfEles,List, SubList) :-
 N is StaPos - 1,
 len(Left,N),
 append(Left,Right,List),
 len(SubList, NoOfEles),
 append(SubList,_,Right).

/*
append(ListOfLists, Appended)
i.e. append([ [[1,2],[2,3]], [[11,12],[12,13]] ], List) will
give List = [ [1,2],[2,3],[11,12],[12,13] ].
*/

append([],[]).
append([H|T], List) :-
 append(T, Tail),
 append(H,Tail,List).

flatten([Head|Tail],FlatList) :-
 flatten(Head,FlatHead),
 flatten(Tail, FlatTail),!,
 append(FlatHead, FlatTail, FlatList).

flatten([],[]).
flatten(X,[X]).

unflat(_,[],[]) :- !.
unflat(N, List, [Head|Tail]) :-
 len(Head,N),
 append(Head,Rear,List),
 unflat(N, Rear, Tail).

% separate(List,N,Front,Rear)

separate([],_,[],[]) :- !.
separate(List, N, Front, Rear) :-
 len(Front,N),
 append(Front,Rear,List).

% separate1(List,N,Front,Rear) Slower than above
separate/4
separate1(List,0,[],List) :- !.
separate1([H|T], N, [H|FT], Rear) :-
 RestN is N - 1,
 separate1(T,RestN, FT, Rear).

frontn(_,[],[]).
frontn(N,List, Front) :-
 len(Front,N),
 append(Front,_,List).

lastn(_,[],[]).
lastn(N, List, End) :-
 len(List,Len),
 N1 is Len - N,
 len(Front,N1),
 append(Front,End, List).

last([],[]) :- !.
last(X,List) :-
 append(_,[X],List).

% min_max(ListOfListOfList, MinOfMin_MaxesInRows)
min_max([[H]], M) :- max(H,M),!.
min_max([H|T], Min) :-
 min_max1(H, MaxH),
 min(MaxH, MinH),
 min_max(T, MinT),
 Min is min(MinH, MinT).
```

```
min_max1([],[]) :- !.
min_max1([H|T], [MaxH|MaxT]) :-
 max(H, MaxH),
 min_max1(T,MaxT).

% max_min(ListOfListOfList, MaxOfMinsInRows)
max_min([[H]], M) :- min(H,M),!.
max_min([H|T], Max) :-
 max_min1(H, MaxH),
 max(MaxH, MinH),
 max_min(T, MinT),
 Max is max(MinH, MinT).

max_min1([],[]) :- !.
max_min1([H|T], [MaxH|MaxT]) :-
 min(H, MaxH),
 max_min1(T,MaxT).

% max_max(ListOfListOfList, MaxOfMaxesInRows)
max_max([[H]], M) :- max(H,M),!.
max_max([H|T], Max) :-
 max_max1(H, MaxH),
 max(MaxH, MaxMaxH),
 max_max(T, MaxT),
 Max is max(MaxMaxH, MaxT).

max_max1([],[]) :- !.
max_max1([H|T], [MaxH|MaxT]) :-
 max(H, MaxH),
 max_max1(T,MaxT).

% min_min(ListOfListOfList, MinOfMinsInRows)
min_min([[H]], M) :- min(H,M),!.
min_min([H|T], Min) :-
 min_min1(H, MinH),
 max(MinH, MinMinH),
 min_min(T, MinT),
 Min is min(MinMinH, MinT).

min_min1([],[]) :- !.
min_min1([H|T], [MinH|MinT]) :-
 min(H, MinH),
 min_min1(T,MinT).

% components(Data, InBands, BandWiseList)

components([], Bands, List) :-
 !,
 append_times(Bands,[],List).

components(List, Bands, BandwiseList) :-
 len(Front,Bands),
 append(Front,Rear,List),
 components(Rear, Bands, Tail),
 !,
 append_corresponding(Front, Tail, BandwiseList).

components([], _, _, []) :- !.
components(List, Band, OfBands, [H|T]) :-
 len(Front,OfBands),
 append(Front,Rear,List),
 member(H, Front, Band),
 components(Rear, Band, OfBands, T).

append_corresponding([], [], []).
append_corresponding([FH|FT], [RH|RT],
[BWH|BWT]) :-
  append([FH], RH, BWH),
  append_corresponding(FT, RT, BWT).

/*
split(List, Ratio, FirstList, SecondList)
Given the ratio M:N among each group of
(M+N) members, first M will be part of
FirstList & N will be part of SecondList
The last group with less than (M + N)
elements, will be treated differently
If ratio 0:0, List=FirstList=SecondList
*/

split([],_,[],[]) :- !.
split(List,0:0,List,List) :- !.
split(List,0:_,List,List) :- !.
split(List,_:0,List,List) :- !.

split(List, F:S, First, Second) :-
 GroupOf is F + S,
 split(List, F:S, GroupOf, First, Second).

split([],_,_,[],[]) :- !.
split(List, F:S, GroupOf, First, Second) :-
 separate(List, GroupOf, Front, Tail),
 separate(Front, F, FirstHead,SecondHead),
 split(Tail, F:S, GroupOf, FirstTail, SecondTail),!,
 append(FirstHead, FirstTail, First),
 append(SecondHead, SecondTail, Second).

split(List, F:_, _, First, Second) :-
 separate(List, F, First, Second).

split(List, _:_, List, []).

/*
 round(Num, Till, Rounded)
 If rounding is desired till n places after decimal then
 use n for Till
 If n = 0 this returns simply the integer part.
 If rounding is desired before decimal use -n
*/

round(Num, Till, Rounded) :-
 MultDivBy is 10^Till,
 Rounded is int(Num*MultDivBy + 0.5)/MultDivBy.

/* Times occurences of Ele in 3rd Arg */

append_times(Times, Ele, List) :-
 append_times(Times, 0, Ele, List).

append_times(Times, Times, _, []) :- !.
append_times(Times, TimesSoFar, Ele, [Ele|Tail]) :-
 NextTime is TimesSoFar + 1,
 append_times(Times, NextTime, Ele, Tail).


% gen_within(Lower, Upper, List)
% generates a list of integers from Lower to Upper

gen_within(Upper, Upper, [Upper]) :- !.
gen_within(Lower, Upper, [Lower|Tail]) :-
 Next is Lower + 1,
 gen_within(Next, Upper, Tail).

% converts a list of numbers to atoms
numbers_atoms([],[]).
numbers_atoms([Num|NumTail], [Atom|AtomTail]) :-
 number_atom(Num, Atom),
 numbers_atoms(NumTail, AtomTail).

%matrix_of_stats_of_mcc(BandWiseLists,MeanMatrix,Std
vMatrix,DenomMatrix),!,
matrix_of_stats_of_mcc([],[],[],[]).
matrix_of_stats_of_mcc([Hs|T],[MeanVect|MT],[TwiceVar
t|TVVT],[Denominators|DT]) :-
```

```
stats_of_mcc(Hs, MeanVect, TwiceVarVect,
Denominators),
matrix_of_stats_of_mcc(T,MT,TVVT,DT).

stats_of_mcc([],[],[],[]).
stats_of_mcc([H|T],[Mean|MT],[TwiceVar|TVT],[Deno
minator|DT]) :-
sort(H,List,[]),
count_table(List, _, 0, _, 0,0,0, Len, Sum, Sum1),!,
Mean is Sum/Len,
Var is Sum1/Len - Mean*Mean,
StdDev is sqrt(Var),
TwiceVar is 2*Var,
Denominator is 2.507132682112O4*StdDev,
stats_of_mcc(T,MT,TVT,DT).

matrix_of_parameters_of_beta_pdf([],[],[],[],[],[],[]) :- !.
matrix_of_parameters_of_beta_pdf([Hs|T],[AlphaMin
usOnes|AT], [BetaMinusOnes|BT],
[Denominators|DT],[Lowests|LT],[Highests|HT],[Full
RangeParams|FT]) :-
parameters_of_beta_pdf(Hs,AlphaMinusOnes,
BetaMinusOnes,
Denominators,Lowests,Highests,FullRangeParams),

matrix_of_parameters_of_beta_pdf(T,AT,BT,DT,LT,HT
,FT).

parameters_of_beta_pdf([],[],[],[],[],[],[]) :- !.
parameters_of_beta_pdf([H|T],[AlphaMinusOne|AT],
[BetaMinusOne|BT],
[Denominator|DT],[Lowest|LT],[Highest|HT],[Guass
Params|GT]) :-
sort(H,List,[]), % sorts H to List with duplicates (if
any)
count_table(List, _, 0, _, 0,0,0, Len, Sum,
Sum1,0,Highest),
!,
Mean is Sum/Len,
VarH is Sum1/Len - Mean*Mean,
List = [Lowest|_],
Sd is sqrt(VarH),

% For Actual Range [Lowest,Highest]
%Lowest is Lowest1 - (N-N)*Sd,
%Highest is Highest1 + (N-N)*Sd,
Range is Highest - Lowest,
C1 is (Mean-Lowest)/Range,
C2 is VarH/(Range*Range),
Alpha is ((C1/C2)*(C1 - C1*C1 - C2)),
Beta is  ((1/C1 - 1)*Alpha),
AlphaMinusOne is Alpha - 1,
BetaMinusOne is Beta -1,
SumOfParam is Alpha + Beta,
gf(SumOfParam, GammaOfSumOfParam),
gf(Alpha, GammaOfAlpha),
gf(Beta, GammaOfBeta),
Denominator is (((Range)^(Alpha+Beta-
1))*(GammaOfAlpha*GammaOfBeta/GammaOfSumO
fParam)),

% Normal distribution beyond [Lowest, Highest]
NormalDenom is 2.507132682112O4*Sd,
TwiceVariance is 2*VarH,
GuassParams =
Mean/NormalDenom/TwiceVariance,

parameters_of_beta_pdf(T, AT,BT,DT,LT,HT,GT).

gf(N,1.56625317653525E297) :-
N >= 167.11.

gf(1,1) :- !.

gf(N, Gamma) :-
N > 0,
N < 1,!,
round(N, 3, N1),
Pos is N1*1000 + 1,
gv(ListOfGammaValues),
member(GammaOfOnePlusN, ListOfGammaValues,Pos),
Gamma is GammaOfOnePlusN/(N1+1).

gf(N, Gamma) :-
N1 is N - 1,
gf(N1, GammaN1),
Gamma is N1*GammaN1.

% Each element is Within corresponding Limits
all_within_limits([],_,_) :- !.
all_within_limits([VH|VT],[LH|LT], [UH|UT]) :-
VH >= LH,
VH =< UH,
all_within_limits(VT,LT,UT).

trans_mats([],[]).
trans_mats([H|T], [TransH|TransT]) :-
transpose(H, TransH),
trans_mats(T,TransT).

apply_within_rule(_,_,0, [], []) :- !.
apply_within_rule(Pixel, SoFar, Class, [LLH|_], [ULH|_]) :-
all_within_limits(Pixel,LLH, ULH),
!, Class is SoFar + 1.
apply_within_rule(Pixel, SoFar, Class, [_|LLT], [_|ULT]) :-
SoFar1 is SoFar + 1,
apply_within_rule(Pixel, SoFar1, Class, LLT, ULT).

min_at(_, [],0,0) :- !.
min_at(V, [X],D,1) :- D is abs(V-X),!.
min_at(V, [X|Tail], Min, Pos) :-
min_at(V, Tail, Min, Pos1),
Min < abs(V-X),!,
Pos is Pos1 + 1.
min_at(V, [X|_], D,1) :- D is abs(V-X).

% new_list(SampleList, DataMatrix, ClassList, NewList),
new_list([],_,_,[]).
new_list([SampleHead|SampleTail],Data,ClassList,
[NewListHead|NewListTail]) :-
new_list1(SampleHead, Data, ClassList, NewListHead),
new_list(SampleTail,Data,ClassList, NewListTail).

new_list1(SampleHead, [SampleHead|_],
[NewListHead|_],NewListHead).
new_list1(SampleHead, [_|DT], [_|CT],NewListHead) :-
new_list1(SampleHead, DT, CT, NewListHead).
```

E6. GUI for histogram generation, display and navigation

```
:- dynamic(old_position/2), ensure_loaded(['statutil.pl']).

histo(Band,OfBands, List) :-
components(List,OfBands,BandValueList),
member(Data, BandValueList, Band),
histogram(Data, UniqueSorted, LowVal, HighVal,
Frequencies, LowFreq, HighFreq, CummFreq),
means(Data, Mean, Median, Mode),
unflat(1, Data, Matrix),
matrix_unb_stats(Matrix, _, UnbCov, _, _),
sd_vector(UnbCov, [SD]),
asserta(data(UniqueSorted, LowVal, HighVal, Frequencies,
LowFreq, HighFreq, CummFreq)),
```

```
draw_histo(UniqueSorted, LowVal, HighVal,
Frequencies, LowFreq, HighFreq,
CummFreq,(Mean,Median,Mode,SD)).

draw_histo(DataStr, Band, OfBands) :-
 string_chars(DataStr, Data),
 draw_histo(Data, Band, OfBands).

draw_histo(Data, Band,OfBands) :-
 list(Data),
 components(Data,OfBands,BandValueList),
 member(Data, BandValueList, Band),
 histogram(Data, UniqueSorted, LowVal, HighVal,
Frequencies, LowFreq, HighFreq, CummFreq),
 means(Data, Mean, Median, Mode),
 unflat(1, Data, Matrix),
 matrix_unb_stats(Matrix, _, UnbCov, _, _),
 sd_vector(UnbCov, [SD]),
 asserta(data(UniqueSorted, LowVal, HighVal,
Frequencies, LowFreq, HighFreq, CummFreq)),
 draw_histo(UniqueSorted, LowVal, HighVal,
Frequencies, LowFreq, HighFreq,
CummFreq,(Mean,Median,Mode,SD)).

draw_histo(UniqueSorted, Lower, Upper,
Frequencies, LowestFreq, HighestFreq, CummFreq,
(Mean,Median,Mode,SD)) :-
 retractall(data(_,_,_,_)),
 asserta(data(-1,-1,-1,-1)),
 create_histogram_dialog(UniqueSorted,
Frequencies, CummFreq, (Mean,Median,Mode,SD)),
 window_handler( gfx_histo, gfx_histo_handler ),
 show_dialog( gfx_histo ),

 gfx_pen_create(axis_pen, 0, 120, 120,1),
 gfx_pen_create(freq_pen, 255, 0, 0, 1),
 gfx_pen_create(cum_freq_pen, 0, 0, 255, 1),
 gfx_pen_create(green, 0, 255, 0,1),
 gfx_pen_create(text_pen,120, 0, 120,1),

 OriginX = 75,
 OriginY = 260,

 asserta(old_position(OriginX,OriginY)),

 SizeX is 256,
 ScaleX is SizeX/(Upper - Lower + 1),
 OldX is OriginX,
 GapX is 50,
 OldValX is Lower,
 ValGapX is GapX/ScaleX,

 SizeY is 250,
 ScaleY is SizeY/(HighestFreq - LowestFreq + 1),
 OldY is OriginY,
 GapY is 20,
 OldValY is LowestFreq,
 ValGapY is GapY/ScaleY,

 last(MaxCF,CummFreq),
 ScaleCF is SizeY/(MaxCF - LowestFreq + 1),

 (retractall(data(_,_));true),
 asserta(data((OriginX, SizeX, Lower,
Upper),(OriginY, SizeY, LowestFreq,
HighestFreq,MaxCF))),

 gfx_begin((gfx_histo,900)),
 draw_axis(x, OriginY, SizeX, OldX, GapX, OldValX,
ValGapX),
 draw_axis(y, OriginX, SizeY, OldY, GapY, OldValY,
ValGapY),
 draw_freq(UniqueSorted, Frequencies, CummFreq,
OriginX, OriginY, ScaleX, ScaleY, ScaleCF, Lower,
LowestFreq),
 asserta(old_position(OriginX,OriginY)).

draw_histo_again :-

 data(UniqueSorted, LowVal, HighVal, Frequencies,
LowFreq, HighFreq, CummFreq),

 OriginX = 75,
 OriginY = 260,

 retractall(old_position(_,_)),
 asserta(old_position(OriginX,OriginY)),

 Lower = LowVal,
 Upper = HighVal,
 HighestFreq = HighFreq,
 LowestFreq = LowFreq,

 SizeX is 256,
 ScaleX is SizeX/(Upper - Lower + 1),
 OldX is OriginX,
 GapX is 50,
 OldValX is Lower,
 ValGapX is GapX/ScaleX,

 SizeY is 250,
 ScaleY is SizeY/(HighestFreq - LowestFreq + 1),
 OldY is OriginY,
 GapY is 20,
 OldValY is LowestFreq,
 ValGapY is GapY/ScaleY,

 last(MaxCF,CummFreq),
 ScaleCF is SizeY/(MaxCF - LowestFreq + 1),

 draw_axis(x, OriginY, SizeX, OldX, GapX, OldValX,
ValGapX),
 draw_axis(y, OriginX, SizeY, OldY, GapY, OldValY,
ValGapY),
 draw_freq(UniqueSorted, Frequencies, CummFreq,
OriginX, OriginY, ScaleX, ScaleY, ScaleCF, Lower,
LowestFreq),
 asserta(old_position(OriginX,OriginY)),
 (
  (
   data(LastX, LastY, LastX1, LastY1),
   data((OriginX,_,_,_),(OriginY,_,_,_)),
   gfx(
      (rop = stock(notxorpen_rop)
         -> (pen = green ->
polyline(LastX,OriginY,LastX,LastY1),
            polyline(OriginX,LastY,LastX1,LastY)
           )
      )
   )
  )
  ; (data(-1,_,_,_))
 ).

draw_axis(x, _, SizeX, OldX, GapX, _, _) :-
  OldX > (SizeX + 2*GapX), !.

draw_axis(x, OriginY, SizeX, OldX, GapX, OldValX,
ValGapX) :-
  Y is OriginY + 8,
  Y1 is OriginY + 10,
  Xs is ip(OldX + GapX/2),
  Ys is OriginY + 3,
  (fwrite(f,6,2,OldValX) -> ValStr),
  len(ValStr,Len),
```

```
  Xt is OldX - 4*Len,
  gfx( (pen = axis_pen -> polyline(OldX, OriginY,
OldX, Y),
                text(Xt, Y1, ValStr),
                polyline(Xs, OriginY, Xs, Ys)
    )
   ),
  NewX is OldX + GapX,
  NewValX is OldValX + ValGapX,
  draw_axis(x, OriginY, SizeX, NewX, GapX,
NewValX, ValGapX).

draw_axis(y, _, _, OldY, _, _, _) :-
  OldY =< 0, !.

draw_axis(y, OriginX, SizeY, OldY, GapY, OldValY,
ValGapY) :-
  X is OriginX - 8,
  X1 is OriginX - 70,
  Ys is ip(OldY - GapY/2),
  Xs is OriginX - 3,
  Yt is OldY - 8,
  number_string(OldValY, ValStr),
  gfx( (pen = axis_pen -> polyline(OriginX, OldY, X,
OldY),
                text(X1, Yt, ValStr),
                polyline(OriginX,Ys, Xs, Ys)
    )
   ),
  NewY is OldY - GapY,
  NewValY is OldValY + ValGapY,
  draw_axis(y, OriginX, SizeY, NewY, GapY,
NewValY, ValGapY).

draw_freq([],[],[],_,_,_,_,_,_,_) :- !.
draw_freq([ValueHead|ValueTail], [FH|FT],
[CFH|CFT], OriginX, OriginY, ScaleX, ScaleY,
ScaleCF, Lower, LowestFreq) :-
  X1 is ip(OriginX + ScaleX*(ValueHead-Lower)),
  Y1 is ip(OriginY - ScaleY*(FH-LowestFreq)),
  X2 is X1,
  old_position(OldX,OldY),
  Y2 is ip(OriginY - ScaleCF*(CFH-LowestFreq)),
  gfx(( pen = freq_pen -> polyline(X1,Y1,X1,OriginY)
)),
  gfx(( pen = cum_freq_pen -> polyline(X2, Y2, OldX,
OldY) )),
  retractall(old_position(_,_)),
  asserta(old_position(X2,Y2)),
  draw_freq(ValueTail, FT, CFT, OriginX, OriginY,
ScaleX, ScaleY,ScaleCF, Lower, LowestFreq).

% create the grafix example dialog windows

create_histogram_dialog(ValList, FreqList,
CummFreq, (Mean,Median,Mode,SD)) :-
  wdcreate( gfx_histo,'Histogram Display',
  [ 140, 22, 495, 415, [ws_sysmenu,ws_caption]
),
  wccreate( (gfx_histo,900), grafix, 'Grafix',
      16, 16, 350, 300,
[ws_child,ws_border,ws_visible] ),
  Cstyle = [ws_child,ws_visible,ws_tabstop,ws_vscroll,
  cbs_dropdown,cbs_disablenoscroll,cbs_autohscroll],
  Sstyle = [ws_child, ws_visible, ws_border, ss_left],

  wccreate( (gfx_histo,8), static, 'Frequency', 85, 320, 60, 16,
Sstyle ),
  wccreate( (gfx_histo,5001), combobox, 'Combo2', 85, 340,
60, 120, Cstyle ),

  wccreate( (gfx_histo,9), static, 'Cumm Freq', 155, 320, 60,
16, Sstyle ),
  wccreate( (gfx_histo,5002), combobox, 'Combo2', 155, 340,
60, 120, Cstyle ),

  (
   write('Mean   = '),
   fwrite(f,10,-2,Mean),
   nl,
   write('Median = '),
   fwrite(f,10,-2,Median),
   nl,
   write('Mode   = '),
   fwrite(f,10,-2,Mode),
   nl,nl,
   write('Std Dev = '),
   fwrite(f,10,-2,SD)
  ) ~> MeansText,

  wccreate( (gfx_histo,10), static, MeansText, 245, 320, 120,
70, Sstyle ),

  fill_combo((gfx_histo,5000), ValList),
  fill_combo((gfx_histo,5001), FreqList),
  fill_combo((gfx_histo,5002), CummFreq).

% handle the close message by returning the atom "close"

gfx_histo_handler(_, msg_close, _, close).

% when focus, change or scroll the combobox

gfx_histo_handler(Win, Msg, _, _) :-
  (
   (
    Win = (gfx_histo,5000) ;
    Win = (gfx_histo,5001) ;
    Win = (gfx_histo,5002)
   )
  ),
  (Msg = msg_change; Msg = msg_focus; Msg = msg_vert),
  data(LastX, LastY, LastX1, LastY1),
  data((OriginX,_,_,_),(OriginY,_,_,_,_)),
  gfx_begin((gfx_histo,900)),
  gfx( (rop = stock(notxorpen_rop)
        -> (pen = green ->
polyline(LastX,OriginY,LastX,LastY1),
        polyline(OriginX,LastY,LastX1,LastY)
    ))
  ),
  retractall(data(_,_,_,_)),
  asserta(data(-1,-1,-1,-1)).

gfx_histo_handler(Win, Msg, _, _) :-
  (
   (
    Win = (gfx_histo,5000) ;
    Win = (gfx_histo,5001)
   )
  ),
  (Msg = msg_change; Msg = msg_focus; Msg = msg_vert),
  data(-1, _, _, _),!,
  retractall(data(_,_,_,_)).

% handle the selection in the combo

  handler(Win, msg_select, _, _) :-
```

```
(
(Win = (gfx_histo,5000) ->
    Win1 = (gfx_histo,5001),
    Win2 = (gfx_histo,5002) )
;
(Win = (gfx_histo,5001) ->
    Win1 = (gfx_histo,5000),
    Win2 = (gfx_histo,5002) )
;
(Win = (gfx_histo,5002) ->
    Win1 = (gfx_histo,5000),
    Win2 = (gfx_histo,5001) )
),
get_selected_pos(Win,Pos),

wlbxsel(Win1, Pos, 1),
wlbxget(Win1, Pos, Text),
wtext(Win1,Text),

wlbxsel(Win2, Pos, 1),
wlbxget(Win2, Pos, Text2),
wtext(Win2,Text2),

wlbxget((gfx_histo,5000), Pos, ValStr),
wlbxget((gfx_histo,5001), Pos, FreqStr),

number_string(Value, ValStr),
number_string(Freq, FreqStr),
data((OriginX, SizeX, Lower, Upper),(OriginY, SizeY,
LowestFreq, HighestFreq, _)),
X is ip(OriginX + (SizeX/(Upper - Lower +
1))*(Value-Lower)),
Y is ip(OriginY - (SizeY/(HighestFreq - LowestFreq +
1))*(Freq-LowestFreq) ),
wsize((gfx_histo,900),_,_,_, Dy),
X1 is 0,
Y1 is Dy,
(
(
data(LastX, LastY, LastX1, LastY1),
data((OriginX,_,_,_),(OriginY,_,_,_,_)),
gfx_begin((gfx_histo,900)),
gfx( (rop = stock(notxorpen_rop)
      -> (pen = green ->
polyline(LastX,OriginY,LastX,LastY1),
        polyline(OriginX,LastY,LastX1,LastY)
   ))
  )
 ); (data(-1,_,_,_))
 ),

 gfx_begin((gfx_histo,900)),
 gfx( (rop = stock(notxorpen_rop)
      -> (pen = green -> polyline(X,OriginY,X,Y1),
        polyline(OriginX,Y,X1,Y)
   ))),

 (retractall(data(_,_,_,_));true),
 asserta(data(X,Y,X1,Y1)).

% handle mouse move in grafix window

gfx_histo_handler((gfx_histo,900), msg_mousemove,
(PosX,PosY), _) :-
 data((OriginX, SizeX, Lower, Upper),(OriginY, SizeY,
LowestFreq, HighestFreq, _)),

 Value is ip(Lower + (PosX - OriginX)*(Upper - Lower
+ 1)/SizeX),
 Freq is ip(LowestFreq + (OriginY -
                    Freq + 1)/SizeY),
```

```
Win1 = (gfx_histo,5001),
Win2 = (gfx_histo,5002),
(
(
 Value >= Lower,
 Value =< Upper,
 number_string(Value,ValStr),
 wlbxfnd( Win, 0, ValStr, AtPos),
 AtPos >= 0,
 wtext(Win,ValStr),

 wlbxsel(Win1, AtPos, 1),
 wlbxget(Win1, AtPos, Text1),
 wtext(Win1,Text1),

 wlbxsel(Win2, AtPos, 1),
 wlbxget(Win2, AtPos, Text2),
 wtext(Win2,Text2)
);
(
 Freq >= LowestFreq,
 Freq =< HighestFreq,
 number_string(Freq,ValStr1),
 wlbxfnd( Win1, 0, ValStr1, AtPos1),
 AtPos1 >= 0,
 wtext(Win1,ValStr1),

 wlbxsel(Win, AtPos1, 1),
 wlbxget(Win, AtPos1, Text),
 wtext(Win,Text),

 wlbxsel(Win2, AtPos1, 1),
 wlbxget(Win2, AtPos1, Text2),
 wtext(Win2,Text2)
)
),

wtext(Win, GreyLevelStr),
wtext(Win1, FrequencyStr),

number_string(GreyLevel, GreyLevelStr),
number_string(Frequency, FrequencyStr),
X is ip(OriginX + (SizeX/(Upper - Lower + 1))*(GreyLevel-
Lower)),
Y is ip(OriginY - (SizeY/(HighestFreq - LowestFreq +
1))*(Frequency-LowestFreq) ),
wsize((gfx_histo,900),_,_,_, Dy),
X1 is 0,
Y1 is Dy,
(
(
data(LastX, LastY, LastX1, LastY1),
data((OriginX,_,_,_),(OriginY,_,_,_,_)),
gfx_begin((gfx_histo,900)),
gfx( (rop = stock(notxorpen_rop)
      -> (pen = green ->
polyline(LastX,OriginY,LastX,LastY1),
        polyline(OriginX,LastY,LastX1,LastY)
   ))
  )
 ); (data(-1,_,_,_))
 ),
 gfx_begin((gfx_histo,900)),
 gfx( (rop = stock(notxorpen_rop)
      -> (pen = green -> polyline(X,OriginY,X,Y1),
        polyline(OriginX,Y,X1,Y)
   ))),
 (retractall(data(_,_,_,_));true),
 asserta(data(X,Y,X1,Y1)).

gfx_histo_handler((gfx_histo,900), msg_mousemove, _, _) :-
 (
  data(-1, _, _, _);
```

```
(
data(LastX, LastY, LastX1, LastY1),
data((OriginX,_,_,_),(OriginY,_,_,_)),
gfx_begin((gfx_histo,900)),
gfx( (rop = stock(notxorpen_rop)
      -> (
          pen = green ->
polyline(LastX,OriginY,LastX,LastY1),
          polyline(OriginX,LastY,LastX1,LastY)
         )
    )
  ),
 retractall(data(_,_,_,_)),
 asserta(data(-1,-1,-1,-1))
 )
).

% handle paint messages by starting, drawing some
graphics, and ending

gfx_histo_handler( Win, msg_paint, grafix, _ ) :-
  Win = (gfx_histo,900),
  gfx_paint( Win ),
  draw_histo_again,
  gfx_end( Win ).


% fill the named combobox with the list of items,
% preselecting the first item

fill_combo(_,[]) :- !.
fill_combo(Window,[H|T]) :-
 number(H),!,
 number_string(H,StrH),
 wlbxadd( Window, -1, StrH),
 fill_combo(Window,T).

fill_combo(Window,[H|T]) :-
 string(H),!,
 wlbxadd( Window, -1, H ),
 fill_combo(Window,T).

fill_combo(Window,[H|T]) :-
 string_chars(S,H),
 wlbxadd( Window, -1, S ),
 fill_combo(Window,T).

% get the selected item

get_selected_item( Window, Item ) :-
  sndmsg( Window, cb_getcursel, 0, 0, R ),
  ( R = -1
  -> Item = ''
  ; sndmsg( Window, cb_getlbtext, R, combo, _ ),
    wintxt( combo, 0, Item )
  ).

% get the position of the selected item

get_selected_pos( Window, R ) :-
  sndmsg( Window, cb_getcursel, 0, 0, R ).

% get the selected item

get_selected_item( Window, Item ) :-
  sndmsg( Window, cb_getcursel, 0, 0, R ),
  ( R = -1
  -> Item = ''
  ; sndmsg( Window, cb_getlbtext, R, combo, _ ),
    wintxt( combo, 0, Item )
  ).
```

E7. Program containing all classification algorithms

```
/* init_mlc, mlc(explicit_fuzzy,'pln_l3.pl','pln_class.pl',2:1,
1.25,all,test,_,_),tidy_mlc.
More Algos: mcc, mlc, fuzzy_mlc, sigma_mlc, sigma_fuzzy,
beta_fuzzy, beta1_fuzzy, max_min, range_fuzzy, overlap,
mean_range_fuzzy, fuzzy_dwm_mlc */

:- consult(['matrixfn.pl','statutil.pl', 'gamma_val.pl',
'fuzzy_stats.pl']).

init_mlc :-

dynamic([class_stats/5,sample_stats/9,fuzzy_sample_stats/
9,fuzzy_membership/2]),

dynamic([sample/3,sample/4,old_args/4,bandwise/1,bands
/1]).

tidy_mlc :-

abolish([class_stats/5,sample_stats/9,fuzzy_sample_stats/9
,fuzzy_membership/2]),

abolish([sample/3,sample/4,old_args/4,bandwise/1,bands/
1,data_size/3]),
  abolish([p/2,image_size/3,data_size/3]).


mlc(Algo,
DataFileName,SampleFileName,X:Y,HowManySigma,
WhichBandsMayBeUnsorted, WhichData, ResultStr, Result)
:-
  sort_bands(WhichBandsMayBeUnsorted,WhichBands),
  collect_and_split_sample_pixels(DataFileName,
SampleFileName,X:Y,WhichBands),
  findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
  arrange(ListOfTrainingPixels,BandWiseLists), /* For
Explicit_Fuzzy, Arranging Training Samples Bandwise */
  asserta(bandwise(BandWiseLists)),
  retractall(old_args(_,_,_,_)),

asserta(old_args(DataFileName,SampleFileName,X:Y,Which
Bands)),
  data_name(WhichData, WhichDataName),    /* Test or
Training Samples */
  merge_required_data(WhichDataName, Matrix, Classes,
OldList),
  ms(classify(Algo, HowManySigma, Classes, Matrix,
ClassList, AssignTime),MS),
  % SampleList = Matrix,
  % new_list(SampleList, Matrix, ClassList,NewList),
  % confusion_matrix(OldList,NewList,ConfMatrix,Khat),
  confusion_matrix(OldList,ClassList,ConfMatrix,Khat),

  len(Matrix, SampleCount),
  findall(All, sample(_,All,_, _), AllPixels),
  append(AllPixels, AllList),
  len(AllList, TotalSamples),
  lwrupr(WhichDataName, WhichDataNameInUpper),
  algo_name(Algo, HowManySigma, TypeName),
  Time is MS/1000, StatCalTime is Time - AssignTime,
  (
  write('/* Confusion Matrix by '),
  write(TypeName),
  write(' on '),
  write(SampleCount),
  tab(1),
  write(WhichDataNameInUpper),
  write(' samples of '),
  write(TotalSamples),write(.),
  nl, tab(3),
```

```
 write(['Bands :
WhichBands/DataFileName/SampleFileName/X:Y]),
 write(' */')
 ) -> Msg,
 (
 write_matrix(Msg, ConfMatrix),
 nl,
 nl,write(' Time (sec) for 1. Classification  : Time),
 nl,tab(16), write('2. Class Assignment :AssignTime),
 nl, tab(16), write('3. Statistics      :StatCalTime),
 tab(7), write('Khat : '), fwrite(f,5,-3, Khat)
 ) -> ResultStr,
 write(ResultStr), nl,
 last(LastRow,
ConfMatrix),last(UnRoundedOverall,LastRow),
 round(UnRoundedOverall, 2, Overall),
 round(Khat,3,KhatRounded),
 round(Time,2,TimeRounded),
 round(AssignTime,2,AssignTimeRounded),
 round(StatCalTime, 2, StatCalTimeRounded),

Result=Overall/KhatRounded/TimeRounded/Assign
TimeRounded/StatCalTimeRounded.

sort_bands(all, all) :- !.
sort_bands(UnSorted, Sorted) :-
  sort(UnSorted, Sorted).

% merge_required_data(WhichData, DataMatrix,
MaxClasses)
merge_required_data(WhichDataName, Matrix,
Classes, OldList) :-
  (
   (WhichDataName = test, findall(Pixels,
sample(_,_,_, Pixels), SamplePixels))
   ;
   (WhichDataName = training,findall(Pixels,
sample(_,_,Pixels,_), SamplePixels))
   ;
   (WhichDataName = all,findall(Pixels,
sample(_,Pixels,_,_), SamplePixels))
  ),
  append(SamplePixels, Matrix),
  findall(SNo, sample(SNo,_,_), List),
  len(List, Classes),
  old_list(SamplePixels,OldList).

data_name(WhichData, test) :-
 member(WhichData,[e,s,ts,tst,test]).
data_name(WhichData, training) :-
 member(WhichData,[g,i,n,r,tr,trn,training]).
data_name(WhichData, all) :-
 member(WhichData,[a,l,al,all]).

algo_name(Algo, Sigma, Name) :-
 algo_name(Algo, Sigma, 5, Name).
algo_name(Algo, Sigma, Name) :- len(Algo, Len), Pos
is Len-5,
 algo_name(Algo, Sigma, Pos, Name).
algo_name(Algo, _, Name) :- lwrupr(Algo, Name).

algo_name(Algo, Sigma, Pos, Name) :-
 cat(L, Algo, [Pos]),
 member(sigma,L),!,
 number_atom(Sigma, Atom),
 lwrupr(Algo, Name1),
 cat([Atom,' ', Name1], Name,_).

old_list(PixelsClassWise, ListOfClasses) :-
 old_list(PixelsClassWise, 1, ListOfClasses).

old_list([],_,[]).
old_list([H|T],SNo,List) :-
 old_list1(H,SNo,H1),
 Next is SNo + 1,
 old_list(T,Next,T1),
 append(H1,T1,List).

old_list1([],_,[]).
old_list1([_|T],SNo,[SNo|T1]) :-
 old_list1(T,SNo,T1).

pixel_count([],0).
pixel_count([[X0,Y0,X1,Y1]|T], Count) :-
 pixel_count(T, PixelsInTail),
 Count is PixelsInTail + (X1 - X0 + 1)*(Y1 - Y0 + 1).

class_stats(ClassNo, Data, ClassList, [], [], [], []) :-
 gather_class_pixels(ClassNo,Data,ClassList, ClassPixels,
_),
 len(ClassPixels,0),!.

class_stats(ClassNo, Data, ClassList, MeanVector, Cov,
LnCovDet, CovInv) :-
 gather_class_pixels(ClassNo,Data,ClassList, ClassPixels,
_),
 matrix_stats(ClassPixels,MeanVector,BiasedCov),
 len(ClassPixels,N),
 N > 1,
 Scalar is 1/(N-1),
 matrix_scalar_mult(BiasedCov,Scalar,Cov),
 matrix_inverse(Cov,_,_,_,CovDeterminant,CovInv),
 LnCovDet is ln(abs(CovDeterminant)).

% gather_class_pixels(ClassNo, Data, ClassList,
PixelsOfClassNo, Other)
gather_class_pixels(_,[],[],[],[]) :- !.
gather_class_pixels(CNo, [H|T],[CNo|CT],[H|PT],OL) :-
 !, gather_class_pixels(CNo, T, CT, PT, OL).
gather_class_pixels(CNo, [H|T],[_|CT], PL,[H|OT]) :-
 gather_class_pixels(CNo,T,CT, PL, OT).

classify(mlc, HowManySigma, Classes, Data, ClassList,
AssignTime) :-
 assert_sample_stats(1, Classes, HowManySigma),
 findall(MeanVector,sample_stats(_,MeanVector,_,_,_,_,_,_),
MeanList),
 matrix_scalar_mult(MeanList, -1, NegativeMeanVector),
 findall(LnDet,sample_stats(_,_,_,_,LnDet,_,_,_),
LnDetCovList),
 findall(CovInv,sample_stats(_,_,_,CovInv,_,_,_,_),
CovInvList),
 ms(assign_pixel_class(Data, ClassList,NegativeMeanVector,
LnDetCovList,CovInvList),MS),
 AssignTime is MS/1000, !.

classify(fuzzy_mlc, HowManySigma, Classes, Data,
ClassList, AssignTime) :-
 assert_sample_stats(1, Classes, HowManySigma),
 bands(Bands),
 TwoPiToPowerNegBandsByTwo is (6.28571428571429)^(-
Bands/2),
 assert_fuzzy_membership(1,Classes,TwoPiToPowerNegBands
ByTwo),
 assert_fuzzy_sample_stats(1, Classes, HowManySigma),
 findall(MeanVector,fuzzy_sample_stats(_,MeanVector,_,_,_,
_,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1, NegativeMeanVector),
 findall(LnDet,fuzzy_sample_stats(_,_,_,_,LnDet,_,_,_),
LnDetCovList),
 findall(CovInv,fuzzy_sample_stats(_,_,_,CovInv,_,_,_,_),
CovInvList),
```

```
ms(assign_pixel_class(Data,
ClassList,NegativeMeanVector,
LnDetCovList,CovInvList),MS),
 AssignTime is MS/1000, !.


classify(fuzzy_is_mlc, HowManySigma, Classes,
Data, ClassList, AssignTime) :-
 assert_sample_stats(1, Classes, HowManySigma),
 bands(Bands),
 TwoPiToPowerNegBandsByTwo is
(6.28571428571429)^(-Bands/2),

findall(MeanVector,sample_stats(_,MeanVector,_,_,_,
_,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1,
NegativeMeanVector),
 findall(CovDet,sample_stats(_,_,_,_,CovDet,_,_,_,_),
CovDetList),
 findall(CovInv,sample_stats(_,_,_,CovInv,_,_,_,_,_),
CovInvList),
 constants(CovDetList,
TwoPiToPowerNegBandsByTwo, Constants),
 findall(Tr, sample(_,_,Tr,_), Trs),
 append(Trs, Matrix),
 normal_memship(Matrix, Constants,
NegativeMeanVector, CovInvList, MemShips),
 all_equal_matrix(Bands, 0, FuzOldCovSum),
 fuzzy_stats(Matrix, MemShips, FuzOldCovSum, _,
_, FuzNegMeanVector, FuzCovInvList,
FuzLnDetCovList),
 ms(assign_pixel_class(Data,
ClassList,FuzNegMeanVector,
FuzLnDetCovList,FuzCovInvList),MS),
 AssignTime is MS/1000, !.

classify(fuzzy_dwm_mlc, HowManySigma, Classes,
Data, ClassList, AssignTime) :-
 assert_sample_stats(1, Classes, HowManySigma),

findall(MeanVector,sample_stats(_,MeanVector,_,_,_,_
,_,_), Means),
 assert_fuzzy_dw_membership(1,Classes,Means),
 assert_fuzzy_sample_stats(1, Classes,
HowManySigma),

findall(MeanVector,fuzzy_sample_stats(_,MeanVector,
_,_,_,_,_,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1,
NegativeMeanVector),

findall(LnDet,fuzzy_sample_stats(_,_,_,_,_,LnDet,_,_,_
), LnDetCovList),

findall(CovInv,fuzzy_sample_stats(_,_,_,CovInv,_,_,_,_,
_), CovInvList),
 ms(assign_pixel_class(Data,
ClassList,NegativeMeanVector,
LnDetCovList,CovInvList),MS),
 AssignTime is MS/1000, !.

classify(sigma_mlc, HowManySigma, Classes, Data,
ClassList, AssignTime) :-
 assert_sample_stats(1, Classes, HowManySigma),

findall(MeanVector,sample_stats(_,MeanVector,_,_,_,_
,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1,
NegativeMeanVector),
 findall(LnDet,sample_stats(_,_,_,_,_,LnDet,_,_,_),
LnDetCovList),
 findall(CovInv,sample_stats(_,_,_,CovInv,_,_,_,_,_),
CovInvList),

findall(Lower,sample_stats(_,_,_,_,_,_,_,Lower,_),LowerList),
findall(Upper,sample_stats(_,_,_,_,_,_,_,Upper),UpperList),
ms(assign_pixel_class(Data, ClassList,NegativeMeanVector,
LnDetCovList,CovInvList,LowerList,UpperList),MS),
 AssignTime is MS/1000, !.

/*
classify(sigma_fuzzy, HowManySigma, Classes, Data,
ClassList, AssignTime) :-
 bands(Bands),
 assert_sample_stats(1, Classes, HowManySigma),
 TwoPiToPowerNegBandsByTwo is (6.28571428571429)^(-
Bands/2),

assert_fuzzy_membership(1,Classes,TwoPiToPowerNegBands
ByTwo),
 assert_fuzzy_sample_stats(1, Classes, HowManySigma),

findall(MeanVector,fuzzy_sample_stats(_,MeanVector,_,_,_,
_,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1, NegativeMeanVector),
 findall(LnDet,fuzzy_sample_stats(_,_,_,_,_,LnDet,_,_,_),
LnDetCovList),
 findall(CovInv,fuzzy_sample_stats(_,_,_,CovInv,_,_,_,_,_),
CovInvList),

findall(Lower,fuzzy_sample_stats(_,_,_,_,_,_,_,Lower,_),Lower
List),

findall(Upper,fuzzy_sample_stats(_,_,_,_,_,_,_,Upper),Upper
List),
 ms(assign_pixel_class(Data, ClassList,NegativeMeanVector,
LnDetCovList,CovInvList,LowerList,UpperList),MS),
 AssignTime is MS/1000, !.
*/

classify(sigma_fuzzy, HowManySigma, Classes, Data,
ClassList, AssignTime) :-
 bands(Bands),
 assert_sample_stats(1, Classes, HowManySigma),
 TwoPiToPowerNegBandsByTwo is (6.28571428571429)^(-
Bands/2),

assert_fuzzy_membership(1,Classes,TwoPiToPowerNegBands
ByTwo),
 assert_fuzzy_sample_stats(1, Classes, HowManySigma),

findall(MeanVector,fuzzy_sample_stats(_,MeanVector,_,_,_,
_,_,_), MeanList),
 matrix_scalar_mult(MeanList, -1, NegativeMeanVector),
 findall(LnDet,fuzzy_sample_stats(_,_,_,_,_,LnDet,_,_,_),
LnDetCovList),
 findall(CovInv,fuzzy_sample_stats(_,_,_,CovInv,_,_,_,_,_),
CovInvList),

findall(Lower,fuzzy_sample_stats(_,_,_,_,_,_,_,Lower,_),Lower
List),

findall(Upper,fuzzy_sample_stats(_,_,_,_,_,_,_,Upper),Upper
List),
 ms(assign_pixel_class_sf1(sf1,
Data,OldClassList,LowerList,UpperList),MS1),
 ms(assign_pixel_class_sf1(sf1,
Data,OldClassList,ClassList,NegativeMeanVector,
LnDetCovList,CovInvList),MS2),
 AssignTime is (MS1+MS2)/1000, !.


classify(explicit_fuzzy, HowManyBeta, _, Data, ClassList,
AssignTime) :-
 findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
 arrange(ListOfTrainingPixels,BandWiseLists),
```

```
bands(Bands),
modulation(HowManyBeta,Bands, BandWiseLists,
MeanMatrix,DenomMatrix),!,
ms(assign_pixel_class(Data, ClassList,
MeanMatrix,DenomMatrix),MS),
AssignTime is MS/1000, !

classify(mean_range_fuzzy, _, _, Data, ClassList,
AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
arrange(ListOfTrainingPixels,BandWiseLists),
means_ranges(BandWiseLists, MeanLessLower,
UpperLessMean, Lower, Upper),
ms(assign_pixel_class(mrf, Data, ClassList,
MeanLessLower, UpperLessMean, Lower,
Upper),MS),
AssignTime is MS/1000, !.

classify(range_fuzzy, _, _, Data, ClassList,
AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
arrange(ListOfTrainingPixels,BandWiseLists),
ranges(BandWiseLists, Lower, Upper, Middle),!,
ms(assign_pixel_class(rf, Data, ClassList, Lower,
Upper, Middle),MS),
AssignTime is MS/1000, !.

classify(overlap_max_max, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap, max_max, Classes, Data,
ClassList, AssignTime).

classify(overlap_max_min, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap, max_min, Classes, Data, ClassList,
AssignTime).

classify(overlap_min_max, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap, min_max, Classes, Data, ClassList,
AssignTime).

classify(overlap_min_min, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap, min_min, Classes, Data, ClassList,
AssignTime).


classify(overlap_fuzzy_max_max, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap_fuzzy, max_max, Classes, Data,
ClassList, AssignTime).

classify(overlap_fuzzy_max_min, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap_fuzzy, max_min, Classes, Data,
ClassList, AssignTime).

classify(overlap_fuzzy_min_max, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap_fuzzy, min_max, Classes, Data,
ClassList, AssignTime).

classify(overlap_fuzzy_min_min, _, Classes, Data,
ClassList, AssignTime) :-
classify(overlap_fuzzy, min_min, Classes, Data,
ClassList, AssignTime).


classify(overlap, MinMaxType, Classes, Data,
ClassList, AssignTime) :-
findall(Tr, sample(_,_,Tr,_), TrList),
ks(TrList,MatK),
lwrupr(MinMaxLower, MinMaxType),
(
 (
  member(MinMaxLower, ['min_max', 'max_min',
'max_max','min_min']),
  MinMaxRule = MinMaxLower
 ); MinMaxRule = min_max
),
call(MinMaxRule(MatK, K_Sigma)),!,
classify(sigma_mlc, K_Sigma, Classes, Data, ClassList,
AssignTime), !.

classify(overlap_fuzzy, MinMaxType, Classes, Data,
ClassList, AssignTime) :-
findall(Tr, sample(_,_,Tr,_), TrList),
ks(TrList,MatK),
lwrupr(MinMaxLower, MinMaxType),
(
 (
  member(MinMaxLower, ['min_max', 'max_min',
'max_max','min_min']),
  MinMaxRule = MinMaxLower
 ); MinMaxRule = min_max
),
call(MinMaxRule(MatK, K_Sigma)),!,
classify(sigma_fuzzy, K_Sigma, Classes, Data, ClassList,
AssignTime), !.

classify(beta_fuzzy, _, _, Data, ClassList, AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
arrange(ListOfTrainingPixels,BandWiseLists),

matrix_of_parameters_of_beta_pdf(BandWiseLists,AlphaMin
usOneMatrix,BetaMinusOneMatrix,DenomMatrix,LowestsM
atrix, HighestsMatrix,FullRangeParamsMatrix),!,
ms(assign_pixel_class(beta_fuzzy, Data, ClassList,
AlphaMinusOneMatrix,BetaMinusOneMatrix,DenomMatrix,
LowestsMatrix,
HighestsMatrix,FullRangeParamsMatrix),MS),
AssignTime is MS/1000, !.

classify(beta1_fuzzy, WeightsData,_,Data, ClassList,
AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
arrange(ListOfTrainingPixels,BandWiseLists),

matrix_of_parameters_of_beta_pdf(BandWiseLists,AlphaMin
usOneMatrix,BetaMinusOneMatrix,DenomMatrix,LowestsM
atrix, HighestsMatrix,GaussParamsMatrix),!,
ms(assign_pixel_class(beta1_fuzzy, Data, ClassList,
AlphaMinusOneMatrix,BetaMinusOneMatrix,DenomMatrix,
LowestsMatrix,
HighestsMatrix,GaussParamsMatrix,WeightsData),MS),
AssignTime is MS/1000, !.

classify(mcc, _, _, Data, ClassList, AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
arrange(ListOfTrainingPixels,BandWiseLists),

matrix_of_stats_of_mcc(BandWiseLists,MeanMatrix,StdDev
Matrix,DenomMatrix),!,
ms(assign_pixel_class(mcc, Data, ClassList,
MeanMatrix,StdDevMatrix,DenomMatrix),MS),
AssignTime is MS/1000, !.

classify(max_min, _, _, Data, ClassList, AssignTime) :-
findall(TrainingPixels, sample(_,_,TrainingPixels, _),
ListOfTrainingPixels),
```

```
arrange(ListOfTrainingPixels,BandWiseLists),

matrix_of_stats_of_mcc(BandWiseLists,MeanMatrix,S
tdDevMatrix,DenomMatrix),!,
 ms(assign_pixel_class(max_min, Data, ClassList,
MeanMatrix,StdDevMatrix,DenomMatrix),MS),
 AssignTime is MS/1000, !.

% arrage(ClassWisePixels, BandWisePixels)
arrange([],[]).
arrange([SPH|SPT],[BWH|BWT]) :-
 transpose(SPH,BWH),
 arrange(SPT, BWT).


% for sigma_fuzzy_mlc
assign_pixel_class_sf1(sf1, [], [], _,_) :- !.
assign_pixel_class_sf1(sf1, [MH|MT], [CLH|CLT],
LowerList,UpperList) :-
 apply_within_rule(MH,0, CLH,LowerList,UpperList),
 assign_pixel_class_sf1(sf1, MT,CLT,LowerList,
UpperList).

% for  sigma_fuzzy1
assign_pixel_class_sf1(sf1, [], [],_, _, _, _) :- !.
assign_pixel_class_sf1(sf1, [MH|MT],[0|OT],
[CLH|CLT], Means, LnDets, CovInvs) :-
 apply_max_rule(MH, CLH, Means, LnDets,
CovInvs),
 assign_pixel_class_sf1(sf1, MT, OT, CLT,Means,
LnDets, CovInvs).
assign_pixel_class_sf1(sf1, [_|MT],[OH|OT],
[OH|CLT], Means, LnDets, CovInvs) :-
 assign_pixel_class_sf1(sf1, MT, OT, CLT,Means,
LnDets, CovInvs).

% For range_fuzzy (rf)
assign_pixel_class(rf, [], [], _, _, _) :- !.
assign_pixel_class(rf, [PH|PT], [CH|CT], Lower,
Upper, Middle) :-
 range_fuzzy(PH, Lower, Upper, Middle, MinVect),
 max_at(MinVect,_,CH), !,
 assign_pixel_class(rf, PT,CT, Lower, Upper, Middle).

% For mean_range_fuzzy (mrf)
assign_pixel_class(mrf, [], [], _, _, _, _) :- !.
assign_pixel_class(mrf, [PH|PT], [CH|CT],
MeanLessLower, UpperLessMean, Lower, Upper) :-
 mean_range_fuzzy(PH, MeanLessLower,
UpperLessMean, Lower, Upper, MinVect),
 max_at(MinVect,_,CH), !,
 assign_pixel_class(mrf, PT,CT, MeanLessLower,
UpperLessMean, Lower, Upper).

% For explicit_fuzzy
assign_pixel_class([],[],_,_) :- !.
assign_pixel_class([Pixel|RestOfPixels], [CH|CT],
MeanMatrix,DenomMatrix) :-
 matrix_of_fuzzy_input(Pixel, MeanMatrix,
DenomMatrix, MinVect),
 max_at(MinVect,_,CH), !,
 assign_pixel_class(RestOfPixels, CT,
MeanMatrix,DenomMatrix).

% For beta_fuzzy
assign_pixel_class(beta_fuzzy, [],[],_,_,_,_,_,_,_) :- !.
assign_pixel_class(beta_fuzzy, [Pixel|RestOfPixels],
[CH|CT],
AlphaMinusOneMatrix,BetaMinusOneMatrix,Denom
Matrix,LowestsMatrix,
HighestsMatrix,FullRangeParamsMatrix) :-
 matrix_of_beta_fuzzy_input(Pixel,
AlphaMinusOneMatrix, BetaMinusOneMatrix,
DenomMatrix, LowestsMatrix, HighestsMatrix,
FullRangeParamsMatrix, MinVect),
 max_at(MinVect,_,CH),!,
 assign_pixel_class(beta_fuzzy, RestOfPixels, CT,
AlphaMinusOneMatrix,BetaMinusOneMatrix,DenomMatrix,
LowestsMatrix, HighestsMatrix,FullRangeParamsMatrix).

% For beta1_fuzzy
assign_pixel_class(beta1_fuzzy, [],[],_,_,_,_,_,_,_) :- !.
assign_pixel_class(beta1_fuzzy, [Pixel|RestOfPixels],
[CH|CT],
AlphaMinusOneMatrix,BetaMinusOneMatrix,DenomMatrix,
LowestsMatrix, HighestsMatrix,GaussParamsMatrix,K1/K2)
:-
 matrix_of_beta1_fuzzy_input(Pixel, AlphaMinusOneMatrix,
BetaMinusOneMatrix, DenomMatrix, LowestsMatrix,
HighestsMatrix, GaussParamsMatrix, MinVect/MaxVect),
 matrix_scalar_mult([MinVect],K1, K1TimesMins),
 matrix_scalar_mult([MaxVect], (1-K1),
OneLessK1TimesMaxes),
 matrix_addition(K1TimesMins, OneLessK1TimesMaxes,
[Xs]),
 % maxlist(Xs, FinalMax),
 % minlist(Xs, FinalMin),
 max(Xs, FinalMax),
 min(Xs, FinalMin),
 Final is K2*FinalMin + (1-K2)*FinalMax,
 abs_diffs(Final, Xs, Diffs),
 min_at(Diffs,_,CH),
 assign_pixel_class(beta1_fuzzy, RestOfPixels, CT,
AlphaMinusOneMatrix,BetaMinusOneMatrix,DenomMatrix,
LowestsMatrix, HighestsMatrix,GaussParamsMatrix,K1/K2).

abs_diffs(_,[],[]).
abs_diffs(V,[H|T],[DH|DT]) :-
 DH is abs(V-H),
 abs_diffs(V,T,DT).

min_at(_, [],0,0) :- !.
min_at(_, [X],X,1) :- !.
min_at(Final, [X|Tail], Min, Pos) :-
 min_at(Final, Tail, Min, Pos1),
 Min < abs(X-Final),!,
 Pos is Pos1 + 1.
min_at(Final, [X|_], Min,1) :- Min is abs(X-Final).

% for mlc & fuzzy_mlc
assign_pixel_class([], [], _, _, _) :- !.
assign_pixel_class([MH|MT], [CLH|CLT], Means, LnDets,
CovInvs) :-
 apply_max_rule(MH, CLH, Means, LnDets, CovInvs),
 assign_pixel_class(MT,CLT,Means, LnDets, CovInvs).

% For mcc
assign_pixel_class(mcc, [],[],_,_,_) :- !.
assign_pixel_class(mcc,[Pixel|RestOfPixels], [CH|CT],
MeanMatrix,StdDevMatrix, DenomMatrix) :-
 matrix_of_mcc(Pixel, MeanMatrix, DenomMatrix,
StdDevMatrix, Matrix),
 transpose(Matrix, Transpose),
 maxes_at(Transpose, Maxes, MaxesAt),
 mins_at(Matrix, Mins, _),
 mcc_class(Maxes,MaxesAt,Mins,CH),
 assign_pixel_class(mcc, RestOfPixels, CT,
MeanMatrix,StdDevMatrix, DenomMatrix).

% For max_min
assign_pixel_class(max_min, [],[],_,_,_) :- !.
assign_pixel_class(max_min,[Pixel|RestOfPixels], [CH|CT],
MeanMatrix,StdDevMatrix, DenomMatrix) :-
 matrix_of_mcc(Pixel, MeanMatrix, DenomMatrix,
StdDevMatrix, Matrix),
 mins_at(Matrix, Mins, _),
```

```
max_at(Mins,_,CH),
assign_pixel_class(max_min, RestOfPixels, CT,
MeanMatrix,StdDevMatrix, DenomMatrix).

mcc_class(_,MaxesAt,_,Class) :-
sort(MaxesAt,UniqueSorted),
count_table(MaxesAt,UniqueSorted, Frequencies),
sort(Frequencies, F_Sorted,[]),
append(Rest, [ModalFreq], F_Sorted), ModalFreq >
1,
not(member(ModalFreq, Rest)),!,
member(ModalFreq, Frequencies, ModePos),
member(Class, UniqueSorted,ModePos).
mcc_class(_, _,Mins,Class) :-
max_at(Mins,_,Class).

% for sigma_mlc & sigma_fuzzy_mlc
assign_pixel_class([], [], _, _, _,_,_) :- !.
assign_pixel_class([MH|MT], [CLH|CLT], Means,
LnDets, CovInvs,LowerList,UpperList) :-
 ((apply_within_rule(MH,0,
CLH,LowerList,UpperList), CLH > 0);
  apply_max_rule(MH, CLH, Means, LnDets,
CovInvs)),
 assign_pixel_class(MT,CLT,Means, LnDets,
CovInvs,LowerList, UpperList).

% matrix_of_beta_fuzzy_input(Pixel, ,MinVect)
MinVect o/p
matrix_of_beta_fuzzy_input(_,[],_,_,_,_,_,[]) :- !.
matrix_of_beta_fuzzy_input(Pixel,
[AlphaMinusOneH|AT], [BetaMinusOneH|BT],
[DenominatorH|DT], [LowestH|LT], [HighestH|HT],
[GuassParamsH|GT],[MinH|MinT]) :-
 beta_fuzzy_memship(Pixel,AlphaMinusOneH,
BetaMinusOneH, DenominatorH, LowestH,
HighestH, GuassParamsH, MemShips),
 minlist(MemShips, UnNormalisedMinH),
 %((sum_and_count(MemShips,Sum,_),Sum =\= 0,
MinH is UnNormalisedMinH/Sum);MinH =
UnNormalisedMinH),
 MinH = UnNormalisedMinH,
 matrix_of_beta_fuzzy_input(Pixel,AT,BT,DT,LT,HT,
GT, MinT).

% matrix_of_beta1_fuzzy_input(Pixel,...
,MinVect/MaxVect)
matrix_of_beta1_fuzzy_input(_,[],_,_,_,_,_,[]/[]) :- !.
matrix_of_beta1_fuzzy_input(Pixel,
[AlphaMinusOneH|AT], [BetaMinusOneH|BT],
[DenominatorH|DT], [LowestH|LT], [HighestH|HT],
[GuassParamsH|GT],[MinH|MinT]/[MaxH|MaxT]) :-
 beta_fuzzy_memship(Pixel,AlphaMinusOneH,
BetaMinusOneH, DenominatorH, LowestH,
HighestH, GuassParamsH, MemShips),
 minlist(MemShips, MinH), maxlist(MemShips,
MaxH),
 matrix_of_beta1_fuzzy_input(Pixel,AT,BT,DT,LT,HT,
GT, MinT/MaxT).

beta_fuzzy_memship([], _,_,_,_,_,_,[]) :- !.
beta_fuzzy_memship([PH|PT],
[AlphaMinusOne|AT],[BetaMinusOne|BT],[Denomin
ator|DT],[Lowest|LT],[Highest|HT],[_|GT],
[MemShipH|MT]) :-
 PH - Lowest > 0, Highest - PH > 0,
 MemShipH is ( ((PH-Lowest)^AlphaMinusOne)*
((Highest-PH)^BetaMinusOne) )/(Denominator),!,
 beta_fuzzy_memship(PT, AT, BT, DT, LT, HT, GT,
MT).

beta_fuzzy_memship([PH|PT],
[_|AT],[_|BT],[_|DT],[_|LT],[_|HT],[GuassParams|GT],
[MemShipH|MT]) :-
 GuassParams = Mean/NormalDenom/TwiceVariance,
 MemShipH is aln( -(abs(PH-Mean)^2/TwiceVariance)
)/NormalDenom,
 beta_fuzzy_memship(PT, AT, BT, DT, LT, HT, GT, MT).

% matrix_of_mcc(Pixel, MeanMatrix, DenomMatrix,
TwiceVarMatrix, Matrix),
matrix_of_mcc(_,[],_,_,[]).
matrix_of_mcc(Pixel, [MeanVect|MT], [TwiceVarVect|TVVT],
[Denominators|DT],[Probs|PT]) :-
 mcc_probabilities(Pixel, MeanVect, TwiceVarVect,
Denominators, Probs),
 matrix_of_mcc(Pixel, MT, TVVT, DT, PT).

mcc_probabilities([],_,_,_,[]).
mcc_probabilities([PH|PT], [PH|MT],[_|TVT],
[Denominator|DT], [Prob|ProbTail]) :-
 Prob is 1/Denominator,!,
 mcc_probabilities(PT, MT, TVT, DT, ProbTail).
mcc_probabilities([PH|PT], [Mean|MT],
[TwiceVariance|TVT], [Denominator|DT], [Prob|ProbTail]) :-
 NegQty is - (abs(PH -Mean)^2)/TwiceVariance,
 ((NegQty > - 705, Prob is aln(NegQty) / Denominator); Prob
= 0),
 !,
 mcc_probabilities(PT, MT, TVT, DT, ProbTail).

% For Overlap sigma_range(MeanVect, SD_Vect,
ListOfSigma, [Low/Upr|LowUprOfOtherBands])
sigma_range([],[], _, []) :- !.
sigma_range([MH|MT],[SDH|SDT], [N|NT], [LH/UH|Tail]) :-
 LH is MH - N*SDH, UH is MH + N*SDH,
 sigma_range(MT,SDT, NT,Tail).

% For Sigma MLC & Sigma Fuzzy sigma_range(MeanVect,
SD_Vect, Sigma, LowVect, UprVect)
sigma_range([],[], _, [], []) :- !.
sigma_range([MH|MT],[SDH|SDT], N, [LH|LT], [UH|UT]) :-
 LH is MH - N*SDH, UH is MH + N*SDH,
 sigma_range(MT,SDT, N,LT, UT).

apply_max_rule(Pixel,Class,Means,LnDets,CovInvs) :-
 find_likelihood(Pixel,Likelihoods, Means, LnDets, CovInvs),
 max_at(Likelihoods, _, Class).

find_likelihood(_,[],[],[],[]).
find_likelihood(Pixel, [LKH|LKT], [MH|MT], [LDH|LDT],
[CIH|CIT]) :-
 matrix_addition([Pixel], [MH], DiffFromMean),
 transpose(DiffFromMean, Transposed),
 matrix_mult(DiffFromMean, CIH, InterMat),
 matrix_mult(InterMat, Transposed, [[Q|_]|_]),
 LKH is -LDH - Q,
 find_likelihood(Pixel, LKT, MT, LDT, CIT).

assert_sample_stats(FromClass, ToClass, HowManySigma) :-
 sample_stats(FromClass, ToClass, HowManySigma).
sample_stats(ClassNo, Classes, _) :-
 ClassNo > Classes,!.
sample_stats(ClassNo, Classes,HowManySigma) :-
 find_sample_stats(ClassNo,MeanVector, Cov, CovInv,
CovDet,LnCovDet, SampleCount),!,
 sd_vector(Cov,SD_Vector),
 sigma_range(MeanVector, SD_Vector, HowManySigma,
LowerSigmaLimit, UpperSigmaLimit),
 (retractall(sample_stats(ClassNo,_,_,_,_,_,_,_,_)) ; true),
 assertz(sample_stats(ClassNo, MeanVector, Cov, CovInv,
CovDet, LnCovDet, SampleCount, LowerSigmaLimit,
UpperSigmaLimit)),
 ClassNext is ClassNo + 1,
```

```
sample_stats(ClassNext,Classes, HowManySigma).

find_sample_stats(ClassNo, MeanVector,Cov, CovInv,
CovDet,LnCovDet, SampleCount) :-
  sample(ClassNo, _,Matrix,_),
  matrix_stats(Matrix,MeanVector,BiasedCov),
  len(Matrix,SampleCount),
  SampleCount > 1,
  Scalar is 1/(SampleCount-1),
  matrix_scalar_mult(BiasedCov,Scalar,Cov),
  matrix_inverse(Cov,_,_,_,CovDet,CovInv),
  LnCovDet is ln(abs(CovDet)).

% constants(CovDets,
TwoPiToPowerNegBandsByTwo, Constants) :-
constants([],_,[]).
constants([CDH|CDT],
TwoPiToPowerNegBandsByTwo, [CH|CT]) :-
 CH is (TwoPiToPowerNegBandsByTwo)*(CDH^(-0.5)),
 constants(CDT,TwoPiToPowerNegBandsByTwo,CT).


normal_memship([], _, _, _, []).
normal_memship([Pixel|Tail], ConstantFactorList,
NegativeMeanVectorList, CovInvList,
[MemShipH|MemShipT]) :-
  normal_memship(Pixel, ConstantFactorList,
NegativeMeanVectorList, CovInvList, MemShips, 0,
Sum),
  matrix_scalar_mult([MemShips],(1/Sum),
[MemShipH]), !,
  normal_memship(Tail, ConstantFactorList,
NegativeMeanVectorList, CovInvList, MemShipT).

normal_memship([], _, _, _, _, [],Sum, Sum).
normal_memship([Pixel|Tail], ClassNo,
ConstantFactorList, NegativeMeanVectorList,
CovInvList, [MemShipH|MemShipT],OldSum1,
Sum1) :-
  normal_memship(Pixel, ConstantFactorList,
NegativeMeanVectorList, CovInvList, MemShips, 0,
Sum),
  member(MemShip, MemShips, ClassNo),
  MemShipH is MemShip/Sum,
  NewSum1 is OldSum1 + MemShipH,
  normal_memship(Tail, ClassNo,
ConstantFactorList, NegativeMeanVectorList,
CovInvList, MemShipT,NewSum1,Sum1).


% normal_memship(Pixel, ConstantFactorList,
CovInvList, MemShip)
NegativeMeanVectorList, CovInvList, MemShips, 0,
Sum).
normal_memship(_, [], [], [],[], Sum, Sum).
normal_memship(Pixel, [CH|CT], [NH|NT],
[CIH|CIT],[MemShip|MemShipT], OldSum, Sum) :-
  matrix_addition([Pixel], [NH], DiffFromMean),
  transpose(DiffFromMean, Transposed),
  matrix_mult(DiffFromMean, CIH, InterMat),
  matrix_mult(InterMat, Transposed, [[Q]]),
  NegQbyTwo is -Q/2,
  % MemShip is ((2*22/7)^(-Bands/2))*(CovDet^(-
1/2))*aln(-Q/2),
  (
   (NegQbyTwo > -708,
    MemShip is aln(NegQbyTwo)*CH
   ); MemShip = 0
  ),
  NewSum is OldSum + MemShip,!,
  normal_memship(Pixel, CT, NT, CIT,MemShipT,
NewSum, Sum).

assert_fuzzy_membership(FromClass,
ToClass,TwoPiToPowerNegBandsByTwo) :-

assert_fuzzy_membership(FromClass,TwoPiToPowerNegBan
dsByTwo),
  NextClass is FromClass + 1,
  assert_fuzzy_membership(NextClass,
ToClass,TwoPiToPowerNegBandsByTwo).

assert_fuzzy_membership(ClassNo,TwoPiToPowerNegBands
ByTwo) :-
  sample(ClassNo, _,Matrix,_),
  sample_stats(ClassNo, MeanVector, _, CovInv, CovDet,
_,_,_,_),
  ConstantFactor is
(TwoPiToPowerNegBandsByTwo)*(CovDet^(-0.5)),
  matrix_scalar_mult([MeanVector],-1,NegativeMeanVector),
  norm_mships(Matrix, ConstantFactor,
NegativeMeanVector, CovInv, MemShipList, 0,
SumOfMemShip),
  Scalar is 1/SumOfMemShip,
  matrix_scalar_mult( [MemShipList], Scalar,
[NormalizedMemShipList]),
  (retractall(fuzzy_membership(ClassNo, _)) ; true),
  assertz(fuzzy_membership(ClassNo,
NormalizedMemShipList)).


norm_mships([], _, _, _,
[],SumOfMemShips,SumOfMemShips) :- !.
norm_mships([Pixel|Tail], ConstantFactor,
NegativeMeanVector, CovInv,
[MemShip|MemShipTail],OldSum,Sum) :-
  matrix_addition([Pixel], NegativeMeanVector,
DiffFromMean),
  transpose(DiffFromMean, Transposed),
  matrix_mult(DiffFromMean, CovInv, InterMat),
  matrix_mult(InterMat, Transposed, [[Q]]),
  NegQbyTwo is -Q/2,
  (
   (NegQbyTwo > -708,
    MemShip is
ConstantFactor*aln(NegQbyTwo)/ConstantFactor
   ); MemShip = 0
  ),
  NewSum is OldSum + MemShip,
  norm_mships(Tail, ConstantFactor, NegativeMeanVector,
CovInv, MemShipTail,NewSum,Sum).


assert_fuzzy_dw_membership(FromClass, ToClass,_) :-
  FromClass > ToClass,!.
assert_fuzzy_dw_membership(FromClass, ToClass,Means) :-
  assert_fuzzy_dw_membership(FromClass,Means),
  NextClass is FromClass + 1,
  assert_fuzzy_dw_membership(NextClass, ToClass,Means).
assert_fuzzy_dw_membership(ClassNo,Means) :-
  sample(ClassNo, _,Matrix,_),
  distance_weighted_membership(ClassNo, Matrix, Means,
NormalizedMemShipList),
  (retractall(fuzzy_membership(ClassNo, _)); true),
  assertz(fuzzy_membership(ClassNo,
NormalizedMemShipList)).

assert_fuzzy_sample_stats(FromClass, ToClass,
HowManySigma) :-
  fuzzy_sample_stats(FromClass, ToClass, HowManySigma).

fuzzy_sample_stats(ClassNo, Classes, _) :-
  ClassNo > Classes,!.
fuzzy_sample_stats(ClassNo, Classes, HowManySigma) :-
```

```
find_fuzzy_sample_stats(ClassNo, MeanVector,Cov,
CovInv, CovDet,LnCovDet, SampleCount),
 matrix_inverse(Cov,_,_,_,CovDeterminant,CovInv),
 LnCovDet is ln(abs(CovDeterminant)),
 sd_vector(Cov,SD_Vector),
 sigma_range(MeanVector, SD_Vector,
HowManySigma, LowerSigmaLimit,
UpperSigmaLimit),

retractall(fuzzy_sample_stats(ClassNo,_,_,_,_,_,_,_,_)),
 assertz(fuzzy_sample_stats(ClassNo, MeanVector,
Cov, CovInv, CovDet, LnCovDet,
SampleCount,LowerSigmaLimit, UpperSigmaLimit)),
 ClassNext is ClassNo + 1,
 fuzzy_sample_stats(ClassNext,Classes,
HowManySigma)

find_fuzzy_sample_stats(ClassNo, MeanVector,Cov,
CovInv, CovDet,LnCovDet, SampleCount) :-
 sample(ClassNo, _,Matrix,_),
 fuzzy_membership(ClassNo, MemShipList),
 matrix_mult([MemShipList],
Matrix,FuzzSumsVector),
 sum_list(MemShipList,SumMemShip),
 OneBySumMemShip is 1/SumMemShip,
 matrix_scalar_mult(FuzzSumsVector,
OneBySumMemShip, FuzzyMeanVector),
 FuzzyMeanVector = [MeanVector],
 matrix_scalar_mult(FuzzyMeanVector,-1,
NegativeMeanVector),
 fuzzy_covariance(Matrix, MemShipList,
NegativeMeanVector, Covariance),
 matrix_scalar_mult(Covariance,
OneBySumMemShip, Cov),
 matrix_inverse(Cov,_,_,_,CovDet,CovInv),
 LnCovDet is ln(abs(CovDet)),
 len(Matrix,SampleCount).

% For Overlap Fuzzy
ks([],[]).
ks(List, MatK) :-
 trans_mats(List, TransLists),
 means_and_sds(TransLists, Means, SDs),
 ks(Means, SDs, MatK).

ks([],[],[]).
ks([_|[]], [_|[]], []) :- !.

ks([MVH|MVT], [SDVH|SDVT], [KH|KT]) :-
 ks(MVH, MVT, SDVH, SDVT, KH),
 ks(MVT, SDVT, KT).

ks(_,[],_,_,[]).
ks(MVH, MVT, SDVH, SDVT, [KVH|KVT]) :-
 MVT = [MVTH|MVTT],
 SDVT = [SDVTH|SDVTT],
 matrix_addition([SDVH],[SDVTH],[SumSDV]),
 matrix_scalar_mult([MVTH], -1, [NegMV]),
 matrix_addition([MVH], [NegMV], [DiffMV]),
 abs_div(DiffMV, SumSDV, KVH),
 ks(MVH, MVTT, SDVH, SDVTT, KVT).

abs_div([],[],[]).
abs_div([DMH|DMT], [SSDH|SSDT],[K|KTail]) :-
 K is abs(DMH)/SSDH,
 abs_div(DMT, SSDT, KTail).

/* Merges the samples with the same name &
Assertz samples as sample/3 */
merge_similar :-
 findall(Name, sample_details(_,Name,_), NameList),
 unique(NameList, Unique),
 forall( member(SampleName, Unique),
  (
   findall(Region, sample_details(_,SampleName,
Region), RegionList),
   member(SampleName, Unique, SampleNo),
   (retractall(sample(SampleNo,_,_)); true),
   assertz(sample(SampleNo, SampleName, RegionList))
  )
 ).

split_samples(X:Y) :-
 findall(SNo, sample(SNo,_), SNoList),
 unique(SNoList, Unique),
 forall( member(SampleNo, Unique),
  (
   sample(SampleNo, PixelList),
   split(PixelList, X:Y, Training, Test),
   (retractall(sample(SampleNo,_,_,_)); true),
   assertz(sample(SampleNo, PixelList, Training, Test))
  )
 ).

/* unique(List, Unique) duplicate members of List
 appears once (& ordered as in List) in Unique
 Example: unique([1,2,3,1,0,2,4], [1,2,3,0,4]) */
unique(List, Unique) :-
 reverse(List, Reversed),
 unik(Reversed, ReversedUnique),
 reverse(ReversedUnique, Unique).

/* unik(List, Unique) duplicate members of List
 appears once in Unique, but a member with
 duplicate entries appears only at its last position
 Example: unik([1,2,3,1,0,2,4], [3,1,0,2,4]) */
unik([],[]).
unik([H|T],L):-member(H,T), !, unik(T,L).
unik([H|T],[H|L]):-unik(T,L).

collect_and_split_sample_pixels(DataFile,
SampleFile,X:Y,WhichBands) :-
 consult(SampleFile),
 merge_similar,
 abolish(sample_details,3),
 findall(SNo, sample(SNo,_,_), SNoList),
 len(SNoList, ToClass),
 consult(DataFile),
 image_size(_,_,Bands),
 findall(N,integer_bound(1, N, Bands), BandList),
 template(BandList, WhichBands, WhichBandsTemplate),
 count(y,WhichBandsTemplate,ReducedBands),
 asserta(bands(ReducedBands)),
 collect_sample_pixels(1, ToClass,
Bands,X:Y,WhichBandsTemplate),
 abolish(p,2).

collect_sample_pixels(FromClass, ToClass,_,_,_) :-
 FromClass > ToClass,!.

collect_sample_pixels(FromClass, ToClass,
Bands,X:Y,WhichBandsTemplate) :-
 collect_sample_pixels(FromClass, Bands, Data),
 unflat(Bands, Data, PixelListAllBands),
 sub_bands(PixelListAllBands,
WhichBandsTemplate,PixelList),
 split(PixelList, X:Y,Training, Test),
 (retractall(sample(FromClass,_,_,_));true),
 assertz(sample(FromClass, PixelList,Training,Test)),
 ClassNow is FromClass + 1,
 collect_sample_pixels(ClassNow, ToClass,
Bands,X:Y,WhichBandsTemplate).

collect_sample_pixels(ClassNo, Bands, Data) :-
 sample(ClassNo,_,RegionList),
 extract_sample_pixels(RegionList, Bands, Data),
```

```
!.

extract_sample_pixels([], _, _, []).
extract_sample_pixels([AreaHead|AreaTail], Bands,
Data) :-
  AreaHead = [X0,Y0,X1,Y1], % Count is (X1 - X0 +
1)*(Y1 - Y0 + 1),
  FromRow is X0,
  ToRow is X1,
  FromEle is Bands*Y0 + 1,
  ToEle is Bands*(Y1 + 1),
  NoOfEles is ToEle - FromEle + 1,
  StartCol is FromEle,
  extract_sample_pixels(FromRow, ToRow, StartCol,
NoOfEles, DataHead),
  extract_sample_pixels(AreaTail, Bands, DataTail),
  append([DataHead, DataTail], Data).

extract_sample_pixels(Row, ToRow, _, _, []) :-
  Row > ToRow.
extract_sample_pixels(Row, ToRow, StartCol,
NoOfEles, Data) :-
  p(Row, String),
  string_chars(String, List),
  sublist(StartCol, NoOfEles, List, DataHead),
  RowNow is Row + 1,
  extract_sample_pixels(RowNow, ToRow, StartCol,
NoOfEles, DataTail),
  append(DataHead, DataTail, Data).

% sub_bands to obtain req bands
sub_bands(List, WhichBands, OfBands,
SubBandList) :-
  findall(N,integer_bound(1, N, OfBands), BandList),
  template(BandList, WhichBands, Template),
  sub_bands(List, Template, SubBandList).

template(BandList, all, Template) :-
  template(BandList, BandList, Template).
template([],_,[]).
template(_,[],[]).
template([H|T],[H|T1],[y|TT]) :-
  template(T,T1,TT).
template([_|T],L,[n|TT]) :-
  template(T,L,TT).

sub_bands([],_,[]) :- !.
sub_bands([H|T], Template, [SBH|SBT]) :-
  match_template(H, Template, SBH),!,
  sub_bands(T, Template, SBT).

match_template(_,[],[]) :- !.
match_template([],_,[]) :- !.
match_template([H1,H2|T],[y,y|TT],[H1,H2|BT]) :-
  match_template(T,TT,BT).
match_template([H1,_|T],[y,_|TT],[H1|BT]) :-
  match_template(T,TT,BT).
match_template([_,H2|T],[_,y|TT],[H2|BT]) :-
  match_template(T,TT,BT).
match_template([_,_|T],[_,_|TT],BT) :-
  match_template(T,TT,BT).
match_template([H|T],[y|TT],[H|BT]) :-
  match_template(T,TT,BT).
match_template([_|T],[_|TT],BT) :-
  match_template(T,TT,BT).
```

E8. GUI for executing selected algorithm and generate statistics

```
:- ensure_loaded([system(pdm), system(pdc),
'hist...mlc_test.pl']).
synonym_hook( exitbutton, (button, value('Exit'),
callback(close)) ).

run :-
  dynamic(attached_to/1),
  dir('*.rsd',0,AllDataFiles),
  dir('*.smp',0,AllTrainFiles),
  AllDataFiles = [DataFile|_],
  asserta(attached_to('#none#')),
  attached_trainfiles(DataFile, AllTrainFiles,
AttachedTrainFiles),
  AttachedTrainFiles = [TrainFile|_],
  mlc_data(DataFile, TrainFile, SampleNameList, ClassList,
BandList, ClassNameList,SampleList,PixelsCountList),
  collect_and_split_sample_pixels(DataFile, TrainFile,2:1,_),
  md(AllDataFiles, AttachedTrainFiles, SampleList,
SampleNameList, ClassList, BandList,
ClassNameList,PixelsCountList).

attached_trainfiles(_,[],[]).
attached_trainfiles(DataFile,
[TrainFile|OtherFiles],[TrainFile|TFT]) :-
  asserta(attached_to('#none#')),
  consult(TrainFile),
  attached_to(SomeFile),
  abolish_files([TrainFile]),
  SomeFile == DataFile,
  attached_trainfiles(DataFile, OtherFiles,TFT).

attached_trainfiles(DataFile, [_|OtherFiles],TFT) :-
  attached_trainfiles(DataFile, OtherFiles,TFT).

md(DataFiles, SampleFiles, SampleList, SampleNameList,
ClassList, BandList, ClassNameList,PixelsCountList) :=
  invoke(modeless),
  at(0,0),
  title( 'Classfiers & Graphics' )

  ; list(data_file),
  header('Data File'),
  size(121,40),
  value(DataFiles),
  layout(popup),
  preset(first#),
  select(one),
  callback(
    (
     dir('*.smp',0,AllTrainFiles),
     attached_trainfiles(data_file?, AllTrainFiles,
AttachedTrainFiles),
     AttachedTrainFiles = [TrainFile|_],
     mlc_data(data_file?, TrainFile, SampleNameList1,
ClassList1, BandList1,
ClassNameList1,SampleList1,PixelsCountList1),
     number_atom(TrainRatio, train_ratio?),
     number_atom(TestRatio, test_ratio?),
     Ratio = TrainRatio:TestRatio,
     numbers_atoms(Bands, BandList1),
     collect_and_split_sample_pixels(data_file?,
TrainFile,Ratio,Bands),

     reset(sample_file!,sample_file#,AttachedTrainFiles,first
     reset(reduced_bands!,reduced_bands#, BandList1, all)
     reset(reduced_samples!, reduced_samples#,
SampleList1, all),
     reset(class!, class#, ClassList1, all),
     reset(class_names!, class_names#, ClassNameList1,
all),
     reset(pixel_counts!, pixel_counts#,
PixelsCountList1,all),
     reset(site_names!,site_names#,SampleNameList1,all)
     )
```

```
    )

; list(sample_file),
  header('Sample File'),
  down,
  size(121,40),
  value(SampleFiles),
  layout(popup),
  preset(first#),
  select(one),
  callback(
   (
    mlc_data(data_file?, sample_file?,
SampleNameList2, ClassList2, _,
ClassNameList2,SampleList2,PixelsCountList2),
    number_atom(TrainRatio, train_ratio?),
    number_atom(TestRatio, test_ratio?),
    Ratio = TrainRatio:TestRatio,
    numbers_atoms(Bands, reduced_bands?),
    collect_and_split_sample_pixels(data_file?,
sample_file?,Ratio,Bands),
    reset(reduced_samples!, reduced_samples#,
SampleList2, all),
    reset(class!, class#, ClassList2, all),
    reset(class_names!, class_names#,
ClassNameList2, all),
    reset(pixel_counts!, pixel_counts#,
PixelsCountList2,all),

reset(site_names!,site_names#,SampleNameList2,all)

    )
       )

  ; list(reduced_samples),
    header('Sites'),
    size(40,48),
    down,
    value(SampleList),
    layout(scrollbar),
    preset(all#),
    select(many),
    callback(
          (
           sel_in_as([@(site_names), @(pixel_counts)],
@(reduced_samples))
          )
         )

  ; list(pixel_counts),
    header('# Pxls'),
    size(71,48),
    across,
    value(PixelsCountList),
    layout(scrollbar),
    preset(all#),
    select(many),
    callback(
          (
           sel_in_as([@(reduced_samples),
@(site_names)], @(pixel_counts) )
          )
         )

  ; list(site_names),
    header('Site Name'),
    size(121,48),
    below(reduced_samples),
    value(SampleNameList),
    layout(scrollbar),
    preset(all#),
    select(many),
    callback(
          (
           sel_in_as([@(reduced_samples), @(pixel_counts)],
@(site_names))
          )
         )

  ; list(train_ratio),
    header('Trn'),
    down,
    size(30,40),
    value(['0','1','2','3','4','5','6','7','8','9']),
    layout(popup),
    preset('2'),
    select(one ),
    callback((train_ratio? = '0' -> test_ratio# = '0'))

  ; text(dummy),
    right(train_ratio,0),
    value(':')

  ; list(test_ratio),
    header('Tst'),
    size(30,40),
    right(dummy,0),
    value(['0','1','2','3','4','5','6','7','8','9']),
    layout(popup),
    preset('1'),
    select(one ),
    callback((test_ratio? = '0' -> train_ratio# = '0'))

  ; list(which_data),
    header('RunOn'),
    size(44,40),
    across,
    value(['All', 'Training', 'Test']),
    layout(popup),
    preset('Test'),
    select(one),
    callback(!(result) = '')

  ; text(alpha_label),
    below(train_ratio),
    size(40,16),
    value('Alpha')

  ; edit(alpha),
    below(alpha_label, 0),
    size(40,16),
    value('1')

  ; text(sigma_label),
    right(alpha_label),
    size(40,16),
    value('Sigma')

  ; edit(sigma),
    right(alpha),
    size(40,16),
    value('1')

  ; list(reduced_bands),
    header('Bands'),
    size(40,48),
    right(which_data,15),
    value(BandList),
    layout(scrollbar),
    preset(all#),
    select(many),
    callback(
          (
           number_atom(TrainRatio, train_ratio?),
           number_atom(TestRatio, test_ratio?),
           Ratio = TrainRatio:TestRatio,
```

```
        numbers_atoms(Bands, reduced_bands?),
        collect_and_split_sample_pixels(data_file?,
sample_file?,Ratio,Bands)
       )
      )

; list(class),
  header('Class'),
  across,
  size(40,48),
  value(ClassList),
  layout(scrollbar),
  preset(all#),
  select(many),
  callback(
      (
       sel_in_as([@(class_names)], @(class)),
       (result)' =
      )
     )

; list(class_names),
  header('Name'),
  across,
  size(121,48),
  value(ClassNameList),
  layout(scrollbar),
  preset(all#),
  select(many),
  callback(
      (
       sel_in_as([@(class)], @(class_names)),
       (result)' =
      )
     )

; list(band),
  header('Band'),
  size(36,40),
  across,
  value(BandList),
  layout(popup),
  preset(first#),
  select(one)

; list(max_min_type),
  header('Max/Min'),
  size(70,40),
  across,
  value(['Default', 'Max_Min',
'Min_Max', 'Max_Max', 'Min_Min']),
  layout(popup),
  preset('Min_Max'),
  select(one),
  callback(
      (
       result! = '',
       findall(Tr, sample(_,_,Tr,_), TrList),
       ks(TrList,MatK),
       (write_kmat(MatK) -> StrK),
       write(StrK),
       atom_string(StrKatom, StrK),
       result! = StrKatom,
       MaxMinType = max_min_type?,
       lwrupr(MinMaxLower,MaxMinType),
       (
        (MinMaxLower == 'default', MinMaxRule =
'min_max')
        ;
        (MinMaxRule = MinMaxLower)
       ),
       call(MinMaxRule(MatK, K_SigmaNum)),!,
       number_atom(K_SigmaNum, K_Sigma),
       sigma! = K_Sigma
      )
     )

; list(graph),
  header('Grafix'),
  size(60,40),
  across,
  value(['Freq', 'Cumm', 'LnCumm', 'Normal','Equaliz']),
  layout(popup),
  preset(none),
  select(one)

; list(curve_type),
  header('Shape'),
  size(50,40),
  across,
  value(['Bar', 'Curve', 'Line', 'Polyn']),
  layout(popup),
  preset(first#),
  select(one),
  callback(!(result) = '')

; list(mlc_algo),
  label(left, 'Algo: '),
  below(reduced_bands),
  size(169,30),
  value(['mlc', 'fuzzy_mlc', 'fuzzy_dwm_mlc', 'sigma_mlc',
'sigma_fuzzy', 'explicit_fuzzy', 'beta_fuzzy', 'beta1_fuzzy',
'max_min', 'mcc', 'mean_range_fuzzy', 'range_fuzzy',
'overlap', 'overlap_fuzzy']),
  layout(popup),
  select(one),
  callback(
      (
       !(result) = '',
       sigma_beta(mlc_algo?, Label, Value),
       !(sigma_label) = Label,
       !(sigma) = Value
      )
     )

; button(btn_run),
  across,
  value('Run'),
  callback(
      (
       able(btn_run,off),
       able(btn_graph,off),
       able(btn_stats,off),
       MLtype = mlc_algo?,

       number_atom(HowManySigma, sigma?),
       number_atom(HowManyAlpha, alpha?),
       lwrupr(WhichData, which_data?),
       (
        reduced_bands? = BandList,
        WhichBands = all
       )
       ;
       numbers_atoms(WhichBands, reduced_bands?)
       ),
       number_atom(X, train_ratio?),
       number_atom(Y, test_ratio?),
       Ratio = X:Y,
       init_mlc,
       (
        MLtype='beta1_fuzzy',
        mlc(MLtype,
data_file?,sample_file?,X:Y,HowManyAlpha/HowManySigma
WhichBands,WhichData, ResultStr_)
```

```
        )
        ;
        (
        (MLtype='overlap' ;
MLtype='overlap_fuzzy'),
        mlc(MLtype,
data_file?,sample_file?,X,Y,max_min_type?,WhichBa
nds,WhichData, ResultStr,_)
        )
        ;
        mlc(MLtype,
data_file?,sample_file?,X,Y,HowManySigma,WhichBa
nds,WhichData, ResultStr,_)
        ),
        atom_string(Result, ResultStr),

        !(result1) = Result,
        able(btn_run,on),
        able(btn_graph,on),
        able(btn_stats,on)
       )
      )

 ; button(btn_stats),
  across,
  value('StsCl'),
  callback(
       (able(btn_run,off),
        able(btn_graph,off),
        able(btn_stats,off),
        class? = [Class|_],
        class_names? =[NameOfClass|_],

        graph_data(which_data?, Class, band?, _,
_,_,Pixels),
        matrix_unb_stats(Pixels, MeanVector, Cov,
_, Corel),
        (
         (
          write('Mean Vector'), tab(7), write('('),
          write(which_data?),
          write(' Data of Class '),
          write(Class/NameOfClass), write(')')
         ) ~> Msg,
         write_matrix(Msg, MeanVector),
         nl,
         write_matrix('Covariance Matrix', Cov),
         nl,
         write_matrix('Corelation Matrix', Corel)
         ) ~> ResultStr
        ),
        write(ResultStr),
        atom_string(Result, ResultStr),
        !(result) = Result,
        able(btn_run,on),
        able(btn_graph,on),
        able(btn_stats,on)
       )
      )

  ; button(btn_all_stats),
   across,
   value('Sts'),
   callback(
        (
         number_atom(TrainRatio, train_ratio?),
         number_atom(TestRatio, test_ratio?),
         Ratio = TrainRatio:TestRatio,
         numbers_atoms(Bands, reduced_bands?),
         collect_and_split_sample_pixels(data_file?,
sample_file?,Ratio,all),
         findall(TrainingPixels, sample(_,_,TrainingPixels,_),
TrnMatrixList),
         dist_matrices(TrnMatrixList, TrnDiv, TrnTransDiv,
TrnBhatDist, TrnDistMatStr),
         findall(TestPixels, sample(_,_,_,TestPixels),
TstMatrixList),
         dist_matrices(TstMatrixList, TstDiv, TstTransDiv,
TstBhatDist, TstDistMatStr),
         findall(ClassNo, sample(ClassNo,_,_,_),
ClassNoList),
         reverse(ClassNoList, Reversed),
         Reversed = [MaxClassNo|_],
         sample_stats_string(1, MaxClassNo,
ListOfStrings),

append([TrnDistMatStr],[TstDistMatStr],DistMatStr),
         append(DistMatStr,ListOfStrings, BigList),
         cat(BigList, ResultStr,_),
         tell('dist_str.txt'),
         write(ResultStr),
         told,
         !(result) = file('dist_str.txt'),
         collect_and_split_sample_pixels(data_file?,
sample_file?,Ratio,Bands)
        )
       )

  ; button(btn_graph),
   across,
   value('Draw'),
   callback(
        (able(btn_run,off),
         able(btn_graph,off),
         able(btn_stats,off),
         class? = [Class|_],
         graph_data(which_data?, Class, band?, Band,
OfBands,Data,Pixels),
         histo(Band,OfBands,Data),
         able(btn_run,on),
         able(btn_graph,on),
         able(btn_stats,on)
        )
       )

  ; exitbutton,
   across

  ; textbox(result),
   next(right, 15, data_file),
   size(450,184),
   value('')

  ; textbox(result1),
   below(alpha),
   size(585,150),
   value('').

sigma_beta(explicit_fuzzy, 'Beta', '1.25').
sigma_beta(beta1_fuzzy, 'Beta', '0.1').
sigma_beta(_, 'Sigma', '1').

mlc_data(DataFileName, SampleFileName, SampleNameList,
ClassList, BandList, ClassNameList,SampleList,
PixelsCountList) :-
 consult(SampleFileName),
 findall(SNo, sample_details(SNo,_,_), SampleSNos),

findall(SampleName,sample_details(_,SampleName,_),Sampl
eNames),
 findall(Span,sample_details(_,_,Span),SpansOfSamples),
 abolish_files([SampleFileName]),
 numbers_atoms(SampleSNos, SampleList),
 sort(SampleNames,UniqueSorted),
```

```
 len(UniqueSorted,Classes),
 gen_within(1,Classes, ClassSNos),
 numbers_atoms(ClassSNos, ClassList),
 unique(SampleNames, ClassNames),
 stringlist_atomlist(ClassNames, ClassNameList),
 stringlist_atomlist(SampleNames,
SampleNameList),
 pixels(SpansOfSamples, Pixels),
 numbers_atoms(Pixels,PixelsCountList),
 see(DataFileName),
 read(Term),
 seen,
 Term = image_size(_,_,Bands),
 gen_within(1,Bands, BandNoList),
 numbers_atoms(BandNoList, BandList)

stringlist_atomlist([], [])
stringlist_atomlist([SH|ST], [AH|AT]) :-
 atom_string(AH,SH),
 stringlist_atomlist(ST, AT)

pixels([],[])
pixels([[X0,Y0,X1,Y1]|T], [Count|PixelsInTail]) :-
  Count is (X1 - X0 + 1)*(Y1 - Y0 + 1),
  pixels(T, PixelsInTail)


% reset(Choices, Preset, NewList, Type)
reset([],[],[],_)
reset(_,_,_,[])
reset(List, List, List,all).
reset([First|Others], First, [First|Others],first).
reset(Choices, [Last], Choices,last) :-
 reverse(Choices, [Last|_]).
reset(Choices, [NthTerm], Choices,Nth) :-
 integer(Nth),
 member(NthTerm, Choices, Nth).
reset(Choices, Terms, Choices,[Nth|Others]) :-
 members(Terms, Choices, [Nth|Others]).

reset(List, Preset, List, Preset)

members([],_,[]).
members([NthTerm|OtherTerms], List, [Nth|Others])
:-
 member(NthTerm, List, Nth),
 members(OtherTerms, List, Others).
members([FirstTerm|OtherTerms],
[FirstTerm|OtherChoices], [first|Others]) :-
 members(OtherTerms,[FirstTerm|OtherChoices],
Others).
members([LastTerm|OtherTerms], List,
[last|Others]) :-
 reverse(List, [LastTerm|_]).
 members(OtherTerms, List, Others).
members([Term|OtherTerms], List, [Term|Others]) :-
 members(OtherTerms, List, Others).

sel_in_as(Lbxs, AsLbx) :-
 StaPos = 0,
 sel_in_as(Lbxs, AsLbx, StaPos).

sel_in_as(Lbxs,AsLbx, Pos) :-
 wlbxsel(AsLbx, Pos, Sflag),
 sel_in(Lbxs, Pos, Sflag),
 NextPos is Pos + 1,
 sel_in_as(Lbxs, AsLbx, NextPos).

sel_in_as(_,_,_).

sel_in([],_,_).
sel_in([LbxH|LbxT], Pos, Sflag) :-
 wlbxsel(LbxH, Pos, Sflag),
 sel_in(LbxT, Pos, Sflag).

sel_in(_,_,_).

sel_flags(Lbx, Flags) :-
 sel_flags(Lbx, 0, Flags).

sel_flags(Lbx, N, []) :-
 \+wlbxsel(Lbx, N, _), !.

sel_flags(Lbx,N, [S|T]) :-
 wlbxsel(Lbx,N,S),
 S = 1,
 Next is N + 1,
 sel_flags(Lbx,Next,T).

sel_flags(Lbx,N, [S|T]) :-
 wlbxsel(Lbx,N,S),
 S = 0,
 Next is N + 1,
 sel_flags(Lbx,Next,T).

sel_flags(_,_,[]).


%%%%

selected(Win, MemNos) :-
 selected(Win, 0, MemNos).

selected(Win,N, [H|T]) :-
 wlbxsel(Win,N,S),
 S = 1,
 H is N + 1,
 selected(Win,H,T).

selected(Win, N, T) :-
 wlbxsel(Win,N,S),
 S = 0,
 H is N + 1,
 selected(Win,H,T).

selected(_,_,[]).


graph_data(DataType, CharClass, CharBand, Band, Bands,
Data, Pixels) :-
 number_atom(Band, CharBand),
 number_atom(Class, CharClass),
 image_size(_,_,Bands),
 (
 (DataType = 'All', sample(Class, Pixels, _,_))
 ;
 (DataType = 'Training', sample(Class, _, Pixels,_))
 ;
 (DataType = 'Test', sample(Class, _,_,Pixels))
 ),
 flatten(Pixels,Data).

sample_stats_string(FromClassNo, ToClassNo, []) :-
 FromClassNo > ToClassNo,!.

sample_stats_string(FromClassNo, ToClassNo,
[HeadStr|TailStr]) :-
 sample(FromClassNo, All, Train, Test),

 matrix_unb_stats(All, MeanVectorAll, CovAll, _, CorelAll),
 matrix_unb_stats(Train, MeanVectorTrain, CovTrain, _,
CorelTrain),
 matrix_unb_stats(Test, MeanVectorTest, CovTest, _,
CorelTest),
 (
 nl,nl,
```

```
write('Statistics of Samples of Class No '),
write(FromClassNo),

nl,nl,
write('All Pixels'),
nl,nl,

write_matrix('Mean Vector', MeanVectorAll),
nl,nl,
write_matrix('Covariance Matrix', CovAll),
nl,nl,
write_matrix('Corelation Matrix', CorelAll),
nl,nl,
write('Training Pixels'),
nl,nl,
write_matrix('Mean Vector', MeanVectorTrain),
nl,nl,
write_matrix('Covariance Matrix', CovTrain),
nl,nl,
write_matrix('Corelation Matrix', CorelTrain),
nl,nl,
write('Test Pixels'),
nl,nl,
write_matrix('Mean Vector', MeanVectorTest),
nl,nl,
write_matrix('Covariance Matrix', CovTest),
nl,nl,
write_matrix('Corelation Matrix', CorelTest)
) ~> HeadStr,

NextClassNo is FromClassNo + 1,
sample_stats_string(NextClassNo, ToClassNo,
TailStr).
```

E9. GUI based program to generate multiple plots on screen or paper

```
:- consult(['auto_code.pl', 'auto_db.pl']).
:- ensure_loaded([system(pdm), system(pdc)]),
   gfx_pen_create(ltgray_pen, 196, 196, 196,1),
   gfx_pen_create(ltgray_dot_pen, 196, 196, 196,dot),
   gfx_pen_create(purple_pen, 255, 0, 255,2),
   gfx_brush_create(purple_brush, 255, 0, 255,solid).

synonym_hook( exitbutton, (button, value('Exit'),
callback(close)) ).

run :-
   gfx_begin(0),
   gfx_resolution(MaxX1, MaxY1, Px, Py),
   dynamic(resolution/4),
   asserta(resolution(MaxX1, MaxY1, Px, Py)),
   gfx_end(0),

   data(DataFiles, SampleFiles, Bands,
Graphs,(AlgoCodes,AlgoNames)),
   gui(DataFiles, SampleFiles, Bands,
Graphs,(AlgoCodes,AlgoNames)).

gui(DataFiles, SampleFiles, Bands,
Graphs,(AlgoCodes,AlgoNames)) ::=
    invoke(modeless),
    at(0,0),
    title('Draw Graphs')

   ;list(datafile),
    size(90,24),
    layout(popup),
    value(DataFiles)

   ;list(samplefile),
    size(90,24),
    layout(popup),
    value(SampleFiles)

   ;list(band),
    layout(popup),
    size(40,24),
    value(Bands),
    callback((
          band? == all -> ( Graphs = [FirstGraph|_], graph# =
FirstGraph )
          ))

   ;list(graph),
    across,
    layout(popup),
    size(40,24),
    value(Graphs),
    callback((
          graph? == all -> (reverse(Bands, [_,MaxBands|_]),
band# = MaxBands )
          ))

   ;list(algo_names),
    below(band),
    size(190, 80),
    value(AlgoNames),
    layout(scrollbar),
    select(many),
    preset(none#),
    callback((
          options(AlgoNames, Status, algo_names?),
          options(AlgoCodes, Status, CodesToSelect),
          algo_codes# = CodesToSelect
          ))

   ;list(algo_codes),
    across,
    size(50, 80),
    value(AlgoCodes),
    layout(scrollbar),
    select(many),
    preset(none#),
    callback((
          options(AlgoCodes, Status, algo_codes?),
          options(AlgoNames, Status, NamesToSelect),
          algo_names# = NamesToSelect
          ))

   ; button(select_all),
    value('Select All'),
    below(algo_names),
    size(100,20),
    callback(( algo_names# = AlgoNames, algo_codes# =
AlgoCodes ))

   ; button(invert_selection),
    value('Invert'),
    across,
    size(60,20),
    callback((
          options(AlgoCodes, Status, algo_codes?),
          invert(Status, Inverted),
          options(AlgoNames, Inverted, NamesToSelect),
          algo_names# = NamesToSelect,
          options(AlgoCodes, Inverted, CodesToSelect),
          algo_codes# = CodesToSelect
          ))

   ; button(select_none),
    value('None'),
    below(algo_codes),
    size(50,20),
    callback(( algo_names# = [], algo_codes# = [] ))
```

```
; button(btn_print),
 right(graph),
 value('Print'),
 callback(
        (
         prnbox('p_temp $p$',_,_,_),
         print(datafile?, samplefile?, NoOfBands,
graph?, algo_codes?),
         prnend(0)
        )
       )

 ;button(btn_draw),
 right(btn_print, 25),
 value('Draw'),
 callback((
        algo_codes? \= [],
        (
         (band? \== all,
number_atom(NoOfBands,band?))
          ;
          NoOfBands = all
        ),
        run(datafile?, samplefile?, NoOfBands,
graph?, algo_codes?)
       ))

 ; button(btn_cancel), right(samplefile),
value('Cancel'),
  callback(( (prnstt(3), prnend(0)) ; true))

 ; exitbutton, right(btn_cancel, 20)

 ; button(btn_set), right(datafile), value('Set Printer'),
  callback((
         prnbox([],_,_,_),
         wfocus(@(band))  % wfocus(gui_4)
        )).

data(DataFiles, SampleFiles, Bands, Graphs,
(AlgoCodes, AlgoNames)) :-
 findall(Graph,graph_code(Graph,_,_), Graphs1),

findall(SampleFile,trainfile_code(_,SampleFile,_,_,_,_,
_), SampleFiles),
 findall(DataFile, datafile_code(_, DataFile, _),
DataFiles),
 DataFiles = [FirstDataFile|_],
 datafile_code(_, FirstDataFile, NoOfBands),
 gen_within(1,NoOfBands,BandCntList),
 atoms_numbers(Bands1, BandCntList),
 append(Bands1, [all], Bands),
 append(Graphs1, [all], Graphs),
 findall(AlgoCode, algo_code(AlgoCode,_),
AlgoCodes),
 findall(AlgoName, algo_code(_, AlgoName),
AlgoNames).

atoms_numbers([], []).
atoms_numbers([AH|AT], [NH|NT]) :-
 number_atom(NH,AH),
 atoms_numbers(AT, NT).

% graph_code(Graph, GraphName, ShortName)
graph_code(oa, 'Algo / Overall Accuracy (OA %)','O
A').
graph_code(kp, 'Algo / Kappa Coefficient (KC %)','K
C').
graph_code(tt, 'Algo / Classification Time (CT ms)','C
T').
%graph_code(st, 'Algo / Statistics Computing Time
(SCT ms)','S C T').
graph_code(st, 'Algo / Stats Compute Time (SCT ms)','S C
T').
graph_code(at, 'Algo / Class Assignment Time (CAT ms)','C
A T').

run(DataFileCode, TrainFileCode, SetsOfLen,
Graph,AlgoCodes) :-
 integer(DataFileCode),
 integer(TrainFileCode),
 !,
 datafile_code(DataFileCode, DataFile, _),
 trainfile_code(TrainFileCode, TrainFile, _, _, _, _, _),
 run(DataFile, TrainFile, SetsOfLen,Graph,
(0,0,500,400),AlgoCodes).

run(DataFile, TrainFile, all,Graph,AlgoCodes) :-
 gfx_begin(0),
 gfx_resolution(MaxX1, MaxY1, _, _),
 gfx_end(0),
 MaxX is MaxX1 - 1,
 MaxY is MaxY1 - 1,
 create_auto_dialog(MaxX,MaxY),
 window_handler(gfx_auto, gfx_auto_handler),
 show_dialog( gfx_auto ),
 gfx_begin((gfx_auto,900)),

 MidX is ip(MaxX/2),
 MidY is ip(MaxY/2),

 List = [1/(0,0,MidX,MidY), 2/(MidX,0,MidX,MidY),
      3/(0,MidY,MidX,MidY), 4/(MidX,MidY,MidX,MidY)],

 forall(member(Ele, List),
      (
       Ele = SetsOfLen/OffSets,
       run(DataFile, TrainFile, SetsOfLen,Graph,
OffSets,AlgoCodes)
      )
     ),

 gfx((pen = purple_pen -> polygon(MidX, 0, MidX, MaxY))),
 gfx((pen = purple_pen -> polygon(0, MidY, MaxX, MidY))),
 gfx_end((gfx_auto,900)).

run(DataFile, TrainFile, SetsOfLen, all,AlgoCodes) :-
 gfx_begin(0),
 gfx_resolution(MaxX1, MaxY1, _, _),
 gfx_end(0),
 MaxX is MaxX1 - 1,
 MaxY is MaxY1 - 1,
 create_auto_dialog(MaxX,MaxY),
 window_handler(gfx_auto, gfx_auto_handler),
 show_dialog( gfx_auto ),
 gfx_begin((gfx_auto,900)),

 MidX is ip(MaxX/2),
 MidY is ip(MaxY/2),

 List = [oa/(0,0,MidX,MidY), tt/(MidX,0,MidX,MidY),
      st/(0,MidY,MidX,MidY), at/(MidX,MidY,MidX,MidY)],

 forall(member(Ele, List),
      (
       Ele = Graph/OffSets,
       run(DataFile, TrainFile, SetsOfLen,Graph,
OffSets,AlgoCodes)
      )
     ),

 gfx((pen = purple_pen -> polygon(MidX, 0, MidX, MaxY))),
 gfx((pen = purple_pen -> polygon(0, MidY, MaxX, MidY))),
 gfx_end((gfx_auto,900)).
```

```
run(DataFile, TrainFile, SetsOfLen,Graph,AlgoCodes)
:-
 SetsOfLen \== all,
 OffSets = (0, 0, 500, 400),
 OffSets = (_, _,MaxX, MaxY),
 create_auto_dialog(MaxX,MaxY),
 window_handler(gfx_auto, gfx_auto_handler),
 show_dialog( gfx_auto ),
 gfx_begin((gfx_auto,900)),
 run(DataFile, TrainFile, SetsOfLen,Graph,
OffSets,AlgoCodes),
 gfx_end((gfx_auto,900))

run(DataFile, TrainFile, SetsOfLen,Graph,
OffSets,AlgoCodes) :-
 SetsOfLen \== all,
 datafile_code(DataFileCode, DataFile, Bands),
 gen_within(1,Bands,BandList),
 sorted_subsets(BandList, SubSets),
 SubSets = [_|BandSubSets],
 findall(BandSet, (member(BandSet, BandSubSets),
len(BandSet, SetsOfLen)), BandSets),
 findall(Code, (member(BandSet1, BandSets),
band_code(Code, Bands, BandSet1)),
BandCodesUnSorted),
 % findall(AlgoCode, algo_code(AlgoCode,_),
AlgoCodes),
 trainfile_code(TrainFileCode, TrainFile, _, _, _, _, _),
 sort(BandCodesUnSorted,BandCodes),
 all_overall(BandCodes, AlgoCodes, DataFileCode,
TrainFileCode, OverAlls,Mins, Maxes,Graph),
 min(Mins, Min),
 max(Maxes, Max),
 graph_code(Graph,GraphName, NameY),
 write(DataFile/TrainFile) -> Hdr,
 write([GraphName]) -> SubHdr,
 draw_runs(AlgoCodes, BandSets, OverAlls, Min,
Max,Hdr,SubHdr,NameY, OffSets).

print(DataFile, TrainFile, all,Graph, AlgoCodes) :-
 %gfx_begin(0),
 resolution(MaxX1, MaxY1, _, _),
 %gfx_end(0),
 MaxX is MaxX1 - 1,
 MaxY is MaxY1 - 1,
 MidX is ip(MaxX/2),
 MidY is ip(MaxY/2),

 prnpag(_),
 gfx_begin(([])),
 gfx_resolution(Xw,Yw,_,_),

 %MidX is ip(Xw/2),
 %MidY is ip(Yw/2),
 PrnMidX is ip(MidX*Xw/MaxX1),
 PrnMidY is ip(MidY*Yw/MaxY1),

 List = [1/(0,0,MidX,MidY),
2/(PrnMidX,0,MidX,MidY),
      3/(0,PrnMidY,MidX,MidY),
4/(PrnMidX,PrnMidY,MidX,MidY)],


 % nl,write(Xw/Yw),

 gfx_mapping(MaxX1, MaxY1, Xw, Yw),

 %gfx_mapping(1, 1, 3, 3),

 forall(member(Ele, List),
     (
      Ele = SetsOfLen/OffSets,
      run(DataFile, TrainFile, SetsOfLen,Graph, OffSets,
AlgoCodes)
     )
    ),

 %gfx((pen = purple_pen -> polygon(MidX, 0, MidX, MaxY))),
 %gfx((pen = purple_pen -> polygon(0, MidY, MaxX, MidY))),

 gfx_end([]).

run(DataFile, TrainFile, SetsOfLen,Graph, AlgoCodes) :-
 SetsOfLen \== all,
 OffSets = (0, 0, 500, 400),
 OffSets = (_,_,MaxX, MaxY),

 prnpag(_),
 gfx_begin([]),

 gfx_resolution(Xw,Yw,_,_),
 gfx_mapping(MaxX, MaxY, Xw, Yw),
 %gfx_mapping(1, 1, 3, 3),

 run(DataFile, TrainFile, SetsOfLen,Graph, OffSets,
AlgoCodes),

 gfx_end([]).


all_overall([],_,_,_,[],[],[],_).
all_overall([BH|BT], Algos,DataFile,TrainFile,
[OH|OT],[Min|MinT], [Max|MaxT],Type) :-
 band_overall(BH, Algos, DataFile, TrainFile, OH, Type),
 min(OH, Min),
 max(OH, Max),
 all_overall(BT, Algos, DataFile, TrainFile,
OT,MinT,MaxT,Type).

band_overall(_, [], _, _, [],_).

band_overall(BandCode, [AlgoCode|AT],DataFileCode,
TrainFileCode, [OH|OT],oa) :-
 % run(Overall, Khat, Time, Assign, StatCalTime, RunNo,
AlgoCode, DataFileCode, TrainFileCode, BandCode,
WhichDataCode, RatioCode, Alpha,Beta,na,na),
 run(OH, _, _, _, _, _, AlgoCode, DataFileCode, _,
TrainFileCode, BandCode, tst, _, _,_,_,_),
 band_overall(BandCode, AT, DataFileCode, TrainFileCode,
OT,oa).

band_overall(BandCode, [AlgoCode|AT],DataFileCode,
TrainFileCode, [OH|OT],kp) :-
 run(_, Khat, _, _, _, _, AlgoCode, DataFileCode,
TrainFileCode, BandCode, tst, _, _,_,_,_),
 (Khat < 0 -> OH = 0;
 OH is 100*Khat),
 band_overall(BandCode, AT, DataFileCode, TrainFileCode,
OT,kp).

band_overall(BandCode, [AlgoCode|AT],DataFileCode,
TrainFileCode, [OH|OT],tt) :-
 run(_,_,TotTime, _, _, _, AlgoCode, DataFileCode,
TrainFileCode, BandCode, tst, _, _,_,_,_),
 OH is 1000*TotTime,
 band_overall(BandCode, AT, DataFileCode, TrainFileCode,
OT,tt).

band_overall(BandCode, [AlgoCode|AT],DataFileCode,
TrainFileCode, [OH|OT],at) :-
 run(_, _, _, AssignTime, _, _, AlgoCode, DataFileCode,
TrainFileCode, BandCode, tst, _, _,_,_,_),
 OH is 1000*AssignTime,
```

```
 band_overall(BandCode, AT, DataFileCode,
TrainFileCode, OT,at)

band_overall(BandCode,
[AlgoCode|AT],DataFileCode, TrainFileCode,
[OH|OT],st) :-
 run(_,_,_,_, StatTime, _, AlgoCode, DataFileCode,
TrainFileCode, BandCode, tst, _, _,_,_),
 OH is 1000*StatTime,
 band_overall(BandCode, AT, DataFileCode,
TrainFileCode, OT,st)

draw_runs(Algos, BandSets, Overall, Min, Max, Hdr,
SubHdr, NameY,(OffSetX, OffSetY, WinWidth,
WinHight)) :-

 gfx_pen_create(axis_pen, 0, 0, 255,1),
 gfx_pen_create(text_pen,120, 0, 120,1),

 % seed((7,11)), % Seed is (2^23) - 1, seed(Seed),

 bandset_gfx_objects(BandSets, 1,
BandSetGfxObjects),

 LowMin1 is ip(Min),
 HighMax1 is ip(Max+0.99999),

 OriginX = 60,
 OriginY is WinHight - 60,

 SizeX is WinWidth - 70,
 len(Algos, AlgoCnt),
 ScaleX is SizeX/(AlgoCnt),
 GapX is ScaleX,
 OldX is OriginX+4,
 EndX is int(OldX + SizeX - GapX + 0.5),

  (
   (LowMin1 =< 100, HighMax1 =< 100) -> ValGapY
= 10;
   intervals(LowMin1, HighMax1, ValGapY,_)
  ),

 LowMin is ValGapY*(LowMin1//ValGapY),
 HighMax is ValGapY*(1 + HighMax1//ValGapY),
 len(HighMax,MaxLen),
 % nl, write(Min /Max / LowMin1 / HighMax1),

 SizeY is WinHight - 130,
 ScaleY is SizeY/(HighMax - LowMin),
 OldValY is LowMin,
 OldY is OriginY-4,
 GapY is ValGapY*ScaleY,
 EndY is int(OldY - SizeY + 0.5),

 band_set_str(BandSets, '-',BandSetStr),
 Font = font(0),
 wfdata(Font,_,FontSize,_,Ascent),
 wfsize(Font, Hdr, WidthHdr, _),
 wfsize(Font, SubHdr, WidthSubHdr, _),
 wfsize(Font, BandSetStr, WidthBandSetStr,_),

 HdrPosX is ip((WinWidth - 1.25*WidthHdr)/2 - 0.5),
 HdrPosY is 2,

 SubHdrPosX is ip((WinWidth -
1.25*WidthSubHdr)/2 - 0.5),
 SubHdrPosY is FontSize + 2,

 PosX is ip((WinWidth - 1.25*WidthBandSetStr)/2 -
0.5),
 PosY is 2*FontSize + 2,

 gfx_origin(OffSetX,OffSetY),
 gfx(text(HdrPosX, HdrPosY, Hdr)),
 gfx(text(SubHdrPosX, SubHdrPosY, SubHdr)),

 draw_axis(x, OldX, OriginY, OldX, GapX, Algos, _,
Xs,EndY),
 draw_axis(y, OldY, OriginX, SizeY, OldY, GapY, OldValY,
ValGapY,MaxLen,HighMax,EndX),

 draw_accuracies(BandSetGfxObjects, BandSets, Overall,
OriginY, ScaleY, LowMin, Xs,PosX,PosY),
 len(NameY, LenNameY),
 NamePosY is ip(OriginY - (SizeY - Ascent*LenNameY)/2 +
0.5),
 write_vertical(NameY, 2, NamePosY,up),
 ArrowX is 5, ArrowY is NamePosY + 40,
 draw_arrow(ArrowX, ArrowY, 20, 90),
 XaxisArrowY is OriginY + 10,
 draw_arrow(2, XaxisArrowY, 20,0),
 NameX_axisPosY is OriginY + 20,
 gfx(text(2,NameX_axisPosY, 'Algo')).

write_vertical(String, X, OldY,up) :-
 lwrupr(String, UprString),
 string_chars(UprString, Chars1),
 reverse(Chars1, Chars),
 Font = font(0),
 wfdata(Font,_,_,_,GapY),
 write_vertical(Chars, X, OldY, GapY, up).

write_vertical(String, X, OldY,down) :-
 lwrupr(String, UprString),
 string_chars(UprString, Chars),
 Font = font(0),
 wfdata(Font,_,_,_,GapY),
 write_vertical(Chars, X, OldY, GapY, down).

write_vertical([], _, _, _,_).
write_vertical([H|T], X, OldY, GapY,up) :-
 string_chars(Str,[H]),
 gfx(text(X, OldY, Str)),
 NewY is OldY - GapY,
 write_vertical(T, X, NewY,GapY,up).

write_vertical([H|T], X, OldY, GapY,down) :-
 string_chars(Str,[H]),
 gfx(text(X, OldY, Str)),
 NewY is OldY + GapY,
 write_vertical(T, X, NewY,GapY,down).

draw_arrow(X, Y, Len, Slope) :-
 draw_arrow(X, Y, Len, Slope, 30, 8).

draw_arrow(X, Y, Len, Slope, AngHead, LenHead) :-
 HeadX is int(X + Len*cos(Slope) + 0.5),
 HeadY is int(Y - Len*sin(Slope) + 0.5),
 X1 is int(HeadX - LenHead*cos(AngHead+Slope)+ 0.5),
 Y1 is int(HeadY + LenHead*sin(AngHead+Slope)+ 0.5),
 X2 is int(HeadX + LenHead*cos(180 - AngHead + Slope)+
0.5),
 Y2 is int(HeadY - LenHead*sin(180 - AngHead + Slope)+
0.5),
 gfx(( polyline(X,Y,HeadX, HeadY), polygon(HeadX, HeadY,
X1, Y1, X2,Y2) )).

draw_axis(x, StartX, OriginY, OldX, GapX, [], _, [],_) :-
 EndX is int(OldX - GapX +0.5),
 gfx((pen = axis_pen -> polyline(StartX,
OriginY,EndX,OriginY))),!.

draw_axis(x, StartX, OriginY, OldX, GapX, [ValX|T],
ValGapX, [OldX|XT],EndY) :-
```

```
Y is OriginY + 8,
Y1 is OriginY + 10,
(write(ValX) -> ValStr),
Xt is OldX - 4,
gfx( (pen = axis_pen -> polyline(OldX, OriginY,
OldX, Y))),
Ystart is OriginY - 4,
gfx( (pen = ltgray_pen -> polyline(OldX, Ystart,
OldX, EndY) )),
write_vertical(ValStr, Xt, Y1,down),
NewX is ip(OldX + GapX+0.5),
draw_axis(x, StartX, OriginY, NewX, GapX, T,
ValGapX,XT, EndY)

draw_axis(y, StartY, OriginX, _, OldY, _, OldValY,
_,MaxLen,MaxValY,EndX) :-
OldValY >= (MaxValY-10^(-MaxLen)),
!,
X is OriginX - 8,
X1 is OriginX - 8*(MaxLen+1),
Yt is OldY - 8,
IntOldValY is int(OldValY),
fwrite(i,MaxLen,0,IntOldValY) -> ValStr,

gfx( (pen = axis_pen -> polyline(OriginX, OldY, X,
OldY),
text(X1, Yt, ValStr),
polyline(OriginX, OldY, OriginX,

StartY)
)
),
Xstart is OriginX + 4,
gfx( (pen = ltgray_pen -> polyline(Xstart, OldY,
EndX, OldY) )).

draw_axis(y, StartY, OriginX, SizeY, OldY, GapY,
OldValY, ValGapY,MaxLen,MaxValY,EndX) :-
X is OriginX - 8,
X1 is OriginX - 8*(MaxLen+1),
Ys is ip(OldY - GapY/2),
Xs is OriginX - 3,
Yt is OldY - 8,

IntOldValY is int(OldValY),
fwrite(i,MaxLen,0,IntOldValY) -> ValStr,
gfx( (pen = axis_pen -> polyline(OriginX, OldY, X,
OldY),
text(X1, Yt, ValStr),
polyline(OriginX,Ys, Xs, Ys)

)
),

Xstart is OriginX + 4,
gfx( (pen = ltgray_pen -> polyline(Xstart, OldY,
EndX, OldY) )),
gfx( (pen = ltgray_dot_pen -> polyline(Xstart, Ys,
EndX, Ys) )),

NewY is ip(OldY - GapY + 0.5),
NewValY is ip(OldValY + ValGapY+0.5),
draw_axis(y, StartY, OriginX, SizeY, NewY, GapY,
NewValY, ValGapY,MaxLen,MaxValY,EndX).

band_set_str([BandSet], OldStr, Str) :-
write(BandSet) -> BandSetStr,
cat([OldStr, BandSetStr], Str, _).

band_set_str([BandSet|BT], OldStr, Str) :-
(write(BandSet), write('  ')) -> BandSetStr,
cat([OldStr, BandSetStr], NewStr, _),
band_set_str(BT, NewStr, Str).

draw_accuracies(_,_,[],_,_,_,_,_,_).
```

```
draw_accuracies([BandSetPen,BandSetFore|SPT],
[BandSet|BT], [Overalls|OT], OriginY,
ScaleY,LowMin,Xs,OldX,Y) :-
pos_list(Xs, Overalls, OriginY, ScaleY, LowMin, PosList),
append([polyline], PosList, PolyPosList),
Term =.. PolyPosList,
gfx(( pen = BandSetPen -> Term )),
(write(BandSet), write('  ')) -> BandSetStr,

gfx((fore = BandSetFore ->  text(OldX,Y, BandSetStr))),

%   wfont((gfx_auto,900), Font),
Font = font(0),
wfsize(Font,BandSetStr,Width,_),
NewX is OldX + 1.25*Width,
draw_accuracies(SPT, BT, OT, OriginY,
ScaleY,LowMin,Xs,NewX,Y).

pos_list(_,[],_,_,_,[]).
pos_list([X|XT], [OverAll|OT], OriginY, ScaleY,
LowMin,[X,Y|T]) :-
Y is ip(OriginY - ScaleY*(OverAll-LowMin)),
pos_list(XT,OT,OriginY, ScaleY,LowMin,T).

len_cat([],[]).
len_cat([H|T], [Len-H|Tail]) :-
len(H,Len),
len_cat(T,Tail).

len_uncat([],[]).
len_uncat([_-H|T],[H|Tail]) :-
len_uncat(T,Tail).

sorted_subsets(Set, SortedSubSets) :-
findall(SubSet,subset(Set,SubSet),SubSets),
len_cat(SubSets, List),
keysort(List,Sorted),
len_uncat(Sorted, SortedSubSets).

subset([],[]).
subset([First|Rest], [First|Sub]) :-
subset(Rest, Sub).

subset([_|Rest], Sub) :-
subset(Rest, Sub).

% gen_within(Lower, Upper, List)
% generates a list of integers from Lower to Upper

gen_within(Upper, Upper, [Upper]) :- !.
gen_within(Lower, Upper, [Lower|Tail]) :-
Next is Lower + 1,
gen_within(Next, Upper, Tail).

band_code(Code, Bands, all) :-
Code is 2^Bands - 1.

band_code(Code, Bands, LessBands) :-
\+var(LessBands),
sort(LessBands, SortedLessBands),
gen_within(1,Bands,BandList),
sorted_subsets(BandList, [_|SubSets]),
member(SortedLessBands, SubSets, Code).

band_code(Code, Bands, LessBands) :-
gen_within(1,Bands,BandList),
sorted_subsets(BandList, [_|SubSets]),
member(LessBands, SubSets, Code).

bandset_gfx_objects([], _, []).
bandset_gfx_objects([_|BT], Cnt,
[BandSetPen,BandSetFore|BSPT]) :-
number_atom(Cnt, CntAtom),
```

```
cat([pen, CntAtom], BandSetPen, _),
cat([fore, CntAtom], BandSetFore, _),

(
 color(Cnt, (R,G,B))
 ;
 (
  B is ip(rand(255)),
  G is ip(rand(255)),
  R is ip(rand(255))
 )
),

gfx_pen_create(BandSetPen, R, G, B, 1),
gfx_fore_create(BandSetFore, R, G, B),
Next is Cnt + 1,
bandset_gfx_objects(BT, Next, BSPT).

color( 1, (127,  25, 255)).
color( 2, (255,   0,   0)).
color( 3, (  0, 255,   0)).
color( 4, (  0,   0, 255)).
color( 5, (139,  35,  35)).
color( 6, (255,   0, 255)).
color( 7, (184, 134,  11)).
color( 8, (139,   0, 139)).
color( 9, (210, 105,  30)).
color(10, (  0, 255, 255)).
color(11, (255, 140,   0)).

range(Min, Max, LowMin, HighMax) :-
  IpMin is ip(Min),
  len(IpMin, LenIpMin),
  RoundTo is -(LenIpMin-1),
  Adj is (10^(-RoundTo))/2,
  round((Min-Adj), RoundTo, LowMin),
  round((Max+Adj), RoundTo, HighMax).

round(Num, Till, Rounded) :-
  MultDivBy is 10^Till,
  Rounded is int(Num*MultDivBy + 0.5)/MultDivBy.

max([],0) :- !.
max([X],X) :- !.
max([X|Tail], Max) :-
  max(Tail, Max),
  Max >= X,!
max([X|_], X).

min([],0) :- !.
min([X],X) :- !.
min([X|Tail], Min) :-
  min(Tail, Min),
  Min =< X,!
min([X|_], X).

% create the grafix example dialog windows

create_auto_dialog(Width, Hight) :-
  wdcreate( gfx_auto, '',
        0, 0, Width, Hight, [ws_border] ),
  wccreate( (gfx_auto,900), grafix, '',
        0, 0, Width, Hight,
  [ws_child,ws_visible,ws_border] ).

gfx_auto_handler((gfx_auto,900), msg_leftdouble, _, _) :-
  wclose((gfx_auto,900)),
  wclose(gfx_auto).

/*
% handle paint messages by starting, drawing some
graphics, and ending

gfx_auto_handler( Win, msg_paint, grafix, _ ) :-
  Win = (gfx_auto,900),
  gfx_paint( Win ),
  draw_runs_again,
  gfx_end( Win ).
*/

intervals(Min, Max, Gap, List) :-
  Diff is int(Max - Min),
  len(Diff, Len),
  Len1 is Len - 2,
  Multiple is 10^Len1,
  Precision is 10^(-(Len+1)),
  gap(Diff, Multiple, Precision, Gap),
  gen_intervals(Min, Max, Gap, [Min], List).

gap(Diff, Multiple, Precision, Multiple) :-
  Diff =< (6+Precision)*Multiple.

gap(Diff, Multiple, Precision, Gap) :-
  Diff =< (15+Precision)*Multiple,
  Gap is 2*Multiple.

gap(Diff, Multiple, Precision, Gap) :-
  Diff =< (30+Precision)*Multiple,
  Gap is 5*Multiple.

gap(Diff, Multiple, Precision, Gap) :-
  Diff =< (60+Precision)*Multiple,
  Gap is 10*Multiple.

gap(Diff, Multiple, Precision, Gap) :-
  Diff =< (150+Precision)*Multiple,
  Gap is 20*Multiple.

gen_intervals(OldValue, Max, Gap, OldList, List) :-
  Max - OldValue < Gap,
  append(OldList, [Max], List).

gen_intervals(OldValue, Max, Gap, OldList, List) :-
  NewValue is OldValue + Gap,
  append(OldList, [NewValue], NewList),
  gen_intervals(NewValue, Max, Gap, NewList, List).


% options(OptList, Status, SelectedOpts)
options([],[],[]).
options([OH|OT], [1|ST], [OH|OPT]) :-
  options(OT, ST, OPT).

options([_|OT], [0|ST], OPT) :-
  options(OT, ST, OPT).

% invert(Status, Inverted),

invert([],[]).
invert([1|ST], [0|IT]) :-
  invert(ST,IT).
invert([0|ST], [1|IT]) :-
  invert(ST,IT).
```

E10. Program containing clauses of fuzzy statistics and values of Gamma function

```
fuzzy_stats(Matrix, MemShips, OldCovSum, Means,
Covs, NegMeans, InvCovs, LnDetCovs) :-
  fuzzy_means(Matrix, MemShips, Means,
TransMemShips, MemShipSums),
  matrix_scalar_mult(Means,-1,NegMeans),
  fuzzy_covs(Matrix, TransMemShips, MemShipSums,
NegMeans, OldCovSum, Covs, InvCovs, LnDetCovs).

fuzzy_covs(_,[],[],[],_,[],[],[]). %Trans of MemShipMat
fuzzy_covs(Matrix, [MemShipHead|MT],
[MemShipSum|MST], [NegMeanVector|NMT],
OldCovSum, [Cov|CT], [InvCov|ICT], [LnDetCov|LDCT])
:-
  fuzzy_covariance(Matrix, MemShipHead,
NegMeanVector, OldCovSum, CovSum),
  matrix_scalar_mult(CovSum,(1/MemShipSum),Cov),
  matrix_inverse(Cov,_,_,_,Det,InvCov),
  LnDetCov is ln(abs(Det)),

fuzzy_covs(Matrix,MT,MST,NMT,OldCovSum,CT,ICT,LDC
T).

fuzzy_covariance([],[],_,CovSum,CovSum).
fuzzy_covariance([MH|MT],
[MemShipHead|MemShipTail], NegMeanVector,
OldCovSum, CovSum) :-
  matrix_addition([MH],[NegMeanVector], Diff),
  transpose(Diff, Transposed),
  matrix_mult(Transposed, Diff, Product),
  matrix_scalar_mult(Product,MemShipHead,
WeightedProduct),
  matrix_addition(OldCovSum,WeightedProduct,
NewCovSum),
  fuzzy_covariance(MT,MemShipTail, NegMeanVector,
NewCovSum, CovSum).

fuzzy_means(Matrix, MemShips, Means) :-
  fuzzy_means(Matrix, MemShips, Means,_, _).

fuzzy_means(Matrix, MemShips,
Means,TransMemShips, SumVector) :-
  transpose(MemShips, TransMemShips),
  matrix_mult(TransMemShips, Matrix, FuzzySums),
  sum_vector(MemShips, SumVector),
  rows_scalar_div(FuzzySums,SumVector,Means).

near_by([],[],_).

near_by([OCH|OldCensTail], [NCH|NewCensTail], Prec)
:-
  near_by_values(OCH,NCH,Prec),
  near_by(OldCensTail,NewCensTail,Prec).

near_by_values([],[],_).
near_by_values([H|T],[H1|T1],Prec) :-
  abs(H-H1) =< Prec,
  near_by_values(T,T1,Prec).
```

E11. List of Gamma Values

```
gv([1
,0.999424
,0.99885
,0.998277
,0.997707
,0.997139
,0.996572
,0.996008
,0.995445
,0.994885
,0.994326
,0.993769
,0.993214
,0.992661
,0.99211
,0.991561
,0.991014
,0.990469
,0.989925
,0.989384
,0.988844
,0.988306
,0.987771
,0.987236
,0.986704
,0.986174
,0.985645
,0.985119
,0.984594
,0.984071
,0.98355
,0.983031
,0.982513
,0.981997
,0.981484
,0.980972
,0.980461
,0.979953
,0.979446
,0.978941
,0.978438
,0.977937
,0.977437
,0.97694
,0.976444
,0.975949
,0.975457
,0.974966
,0.974477
,0.97399
,0.973504
,0.97302
,0.972538
,0.972058
,0.971579
,0.971103
,0.970627
,0.970154
,0.969682
,0.969212
,0.968744
,0.968277
,0.967812
,0.967349
,0.966887
,0.966427
,0.965969
,0.965512
,0.965057
,0.964604
,0.964152
,0.963702
,0.963254
,0.962807
,0.962362
,0.961918
,0.961476
,0.961036
,0.960598
,0.960161
,0.959725
,0.959292
,0.958859
,0.958429
,0.958
,0.957573
,0.957147
,0.956723
,0.9563
,0.955879
,0.955459
,0.955042
,0.954625
,0.954211
,0.953797
,0.953386
,0.952976
,0.952567
,0.95216
,0.951755
,0.951351
,0.950948
,0.950548
,0.950148
,0.94975
,0.949354
,0.948959
,0.948566
,0.948174
,0.947784
,0.947396
,0.947008
,0.946623
,0.946238
,0.945856
,0.945474
,0.945095
,0.944716
,0.944339
,0.943964
,0.94359
,0.943218
,0.942847
,0.942477
,0.942109
,0.941743
,0.941378
,0.941014
,0.940652
,0.940291
,0.939931
,0.939574
,0.939217
,0.938862
,0.938508
,0.938156
,0.937805
,0.937456
,0.937108
,0.936761
,0.936416
,0.936072
,0.93573
,0.935389
,0.935049
,0.934711
,0.934374
,0.934039
,0.933705
,0.933372
,0.933041
,0.932711
,0.932382
,0.932055
,0.931729
,0.931405
,0.931082
,0.93076
,0.93044
,0.930121
,0.929803
,0.929487
,0.929172
,0.928858
,0.928546
,0.928235
,0.927925
,0.927617
,0.92731
,0.927004
,0.9267
,0.926397
,0.926095
,0.925794
,0.925495
,0.925197
,0.924901
,0.924606
,0.924312
,0.924019
,0.923728
,0.923438
,0.923149
,0.922862
,0.922575
,0.92229
,0.922007
,0.921724
,0.921443
,0.921164
,0.920885
,0.920608
,0.920332
,0.920057
,0.919783
,0.919511
,0.91924
,0.918971
,0.918702
,0.918435
,0.918169
,0.917904
,0.91764
,0.917378
,0.917117
,0.916857
,0.916599
,0.916341
,0.916085
,0.91583
,0.915577
,0.915324
,0.915073
,0.914823
,0.914574
,0.914326
,0.91408
,0.913834
,0.91359
,0.913348
,0.913106
,0.912866
,0.912626
,0.912388
,0.912151
,0.911916
,0.911681
,0.911448
,0.911216
,0.910985
,0.910755
,0.910526
,0.910299
,0.910073
,0.909847
,0.909623
,0.909401
,0.909179
,0.908959
,0.908739
,0.908521
,0.908304
,0.908088
,0.907874
,0.90766
,0.907448
,0.907236
,0.907026
,0.906817
,0.906609
,0.906403
,0.906197
,0.905992
,0.905789
,0.905587
,0.905386
,0.905186
,0.904987
,0.904789
,0.904593
,0.904397
,0.904203
,0.90401
,0.903817
,0.903626
,0.903436
,0.903247
,0.90306
,0.902873
,0.902688
,0.902503
,0.90232
```

,0.902138
,0.901956
,0.901776
,0.901597
,0.901419
,0.901243
,0.901067
,0.900892
,0.900719
,0.900546
,0.900375
,0.900204
,0.900035
,0.899867
,0.8997
,0.899534
,0.899369
,0.899205
,0.899042
,0.89888
,0.898719
,0.898559
,0.898401
,0.898243
,0.898087
,0.897931
,0.897777
,0.897623
,0.897471
,0.897319
,0.897169
,0.89702
,0.896872
,0.896725
,0.896578
,0.896433
,0.896289
,0.896146
,0.89600
,0.895863
,0.895723
,0.895584
,0.895447
,0.89531
,0.895174
,0.895039
,0.894905
,0.894772
,0.894641
,0.89451
,0.89438
,0.894251
,0.894124
,0.893997
,0.893871
,0.893746
,0.893623
,0.8935
,0.893378
,0.893257
,0.893138
,0.893019
,0.892901
,0.892785
,0.892669
,0.892554
,0.89244
,0.892327
,0.892216
,0.892105

,0.891671
,0.891566
,0.891461
,0.891357
,0.891254
,0.891152
,0.891051
,0.890951
,0.890851
,0.890753
,0.890656
,0.89056
,0.890465
,0.89037
,0.890277
,0.890185
,0.890093
,0.890003
,0.889913
,0.889825
,0.889737
,0.889651
,0.889565
,0.88948
,0.889397
,0.889314
,0.889232
,0.889151
,0.889071
,0.888992
,0.888914
,0.888837
,0.88876
,0.888685
,0.888611
,0.888537
,0.888465
,0.888393
,0.888323
,0.888253
,0.888184
,0.888117
,0.88805
,0.887984
,0.887919
,0.887855
,0.887791
,0.887729
,0.887668
,0.887607
,0.887548
,0.887489
,0.887432
,0.887375
,0.887319
,0.887264
,0.88721
,0.887157
,0.887105
,0.887054
,0.887003
,0.886954
,0.886905
,0.886858
,0.886811
,0.886765
,0.88672
,0.886676
,0.886633
,0.886591
,0.886549

,0.886431
,0.886393
,0.886356
,0.88632
,0.886285
,0.886251
,0.886218
,0.886185
,0.886154
,0.886123
,0.886094
,0.886065
,0.886037
,0.88601
,0.885984
,0.885958
,0.885934
,0.88591
,0.885888
,0.885866
,0.885845
,0.885825
,0.885806
,0.885787
,0.88577
,0.885753
,0.885738
,0.885723
,0.885709
,0.885696
,0.885684
,0.885673
,0.885662
,0.885652
,0.885644
,0.885636
,0.885629
,0.885623
,0.885618
,0.885613
,0.88561
,0.885607
,0.885605
,0.885604
,0.885604
,0.885605
,0.885606
,0.885609
,0.885612
,0.885616
,0.885621
,0.885627
,0.885634
,0.885642
,0.88565
,0.885659
,0.885669
,0.88568
,0.885692
,0.885705
,0.885718
,0.885733
,0.885748
,0.885764
,0.885781
,0.885799
,0.885817
,0.885837
,0.885857
,0.885878
,0.8859
,0.885923
,0.885946

,0.885971
,0.885996
,0.886022
,0.886049
,0.886077
,0.886105
,0.886135
,0.886165
,0.886196
,0.886228
,0.886261
,0.886295
,0.886329
,0.886364
,0.8864
,0.886437
,0.886475
,0.886513
,0.886553
,0.886593
,0.886634
,0.886676
,0.886719
,0.886762
,0.886806
,0.886852
,0.886898
,0.886944
,0.886992
,0.88704
,0.88709
,0.88714
,0.88719
,0.887242
,0.887295
,0.887348
,0.887402
,0.887457
,0.887513
,0.887569
,0.887627
,0.887685
,0.887744
,0.887804
,0.887864
,0.887926
,0.887988
,0.888051
,0.888115
,0.88818
,0.888245
,0.888311
,0.888378
,0.888446
,0.888515
,0.888584
,0.888655
,0.888726
,0.888798
,0.88887
,0.888944
,0.889018
,0.889093
,0.889169
,0.889246
,0.889323
,0.889402
,0.889481
,0.889561
,0.889641
,0.889723
,0.889805
,0.889888

,0.889972
,0.890057
,0.890142
,0.890229
,0.890316
,0.890404
,0.890492
,0.890582
,0.890672
,0.890763
,0.890855
,0.890947
,0.891041
,0.891135
,0.89123
,0.891326
,0.891422
,0.89152
,0.891618
,0.891717
,0.891816
,0.891917
,0.892018
,0.89212
,0.892223
,0.892327
,0.892431
,0.892537
,0.892643
,0.892749
,0.892857
,0.892965
,0.893074
,0.893184
,0.893295
,0.893406
,0.893519
,0.893632
,0.893746
,0.89386
,0.893976
,0.894092
,0.894209
,0.894326
,0.894445
,0.894564
,0.894684
,0.894805
,0.894927
,0.895049
,0.895172
,0.895296
,0.895421
,0.895547
,0.895673
,0.8958
,0.895928
,0.896056
,0.896186
,0.896316
,0.896447
,0.896579
,0.896711
,0.896844
,0.896978
,0.897113
,0.897249
,0.897385
,0.897522
,0.89766
,0.897799
,0.897938
,0.898078

,0.898219
,0.898361
,0.898504
,0.898647
,0.898791
,0.898936
,0.899081
,0.899228
,0.899375
,0.899523
,0.899672
,0.899821
,0.899971
,0.900122
,0.900274
,0.900427
,0.90058
,0.900734
,0.900889
,0.901044
,0.901201
,0.901358
,0.901516
,0.901674
,0.901834
,0.901994
,0.902155
,0.902317
,0.902479
,0.902642
,0.902806
,0.902971
,0.903137
,0.903303
,0.90347
,0.903638
,0.903806
,0.903976
,0.904146
,0.904317
,0.904489
,0.904661
,0.904834
,0.905008
,0.905183
,0.905358
,0.905535
,0.905712
,0.905889
,0.906068
,0.906247
,0.906427
,0.906608
,0.90679
,0.906972
,0.907155
,0.907339
,0.907523
,0.907709
,0.907895
,0.908082
,0.90827
,0.908458
,0.908647
,0.908837
,0.909028
,0.909219
,0.909411
,0.909604
,0.909798
,0.909993
,0.910188
,0.910384

,0.910581
,0.910778
,0.910977
,0.911176
,0.911376
,0.911576
,0.911778
,0.91198
,0.912183
,0.912386
,0.912591
,0.912796
,0.913002
,0.913208
,0.913416
,0.913624
,0.913833
,0.914043
,0.914253
,0.914464
,0.914676
,0.914889
,0.915102
,0.915317
,0.915532
,0.915747
,0.915964
,0.916181
,0.916399
,0.916618
,0.916838
,0.917058
,0.917279
,0.917501
,0.917724
,0.917947
,0.918171
,0.918396
,0.918622
,0.918848
,0.919075
,0.919303
,0.919532
,0.919761
,0.919992
,0.920222
,0.920454
,0.920687
,0.92092
,0.921154
,0.921389
,0.921624
,0.92186
,0.922098
,0.922335
,0.922574
,0.922813
,0.923053
,0.923294
,0.923536
,0.923778
,0.924021
,0.924265
,0.92451
,0.924755
,0.925001
,0.925248
,0.925496
,0.925744
,0.925993
,0.926243
,0.926494
,0.926746

,0.926998
,0.927251
,0.927505
,0.927759
,0.928014
,0.928271
,0.928527
,0.928785
,0.929043
,0.929302
,0.929562
,0.929823
,0.930084
,0.930346
,0.930609
,0.930873
,0.931137
,0.931403
,0.931669
,0.931935
,0.932203
,0.932471
,0.93274
,0.93301
,0.93328
,0.933552
,0.933824
,0.934097
,0.93437
,0.934645
,0.93492
,0.935196
,0.935472
,0.93575
,0.936028
,0.936307
,0.936587
,0.936867
,0.937148
,0.93743
,0.937713
,0.937997
,0.938281
,0.938566
,0.938852
,0.939138
,0.939426
,0.939714
,0.940003
,0.940293
,0.940583
,0.940874
,0.941166
,0.941459
,0.941753
,0.942047
,0.942342
,0.942638
,0.942934
,0.943232
,0.94353
,0.943829
,0.944129
,0.944429
,0.94473
,0.945032
,0.945335
,0.945639
,0.945943
,0.946248
,0.946554
,0.94686
,0.947168
,0.947476
,0.947785
,0.948095
,0.948405
,0.948716
,0.949029
,0.949341
,0.949655
,0.949969
,0.950284
,0.9506
,0.950917
,0.951235
,0.951553
,0.951872
,0.952192
,0.952512
,0.952834
,0.953156
,0.953479
,0.953802
,0.954127
,0.954452
,0.954778
,0.955105
,0.955432
,0.955761
,0.95609
,0.95642
,0.95675
,0.957082
,0.957414
,0.957747
,0.958081
,0.958416
,0.958751
,0.959087
,0.959424
,0.959762
,0.9601
,0.96044
,0.96078
,0.961121
,0.961462
,0.961805
,0.962148
,0.962492
,0.962837
,0.963182
,0.963529
,0.963876
,0.964224
,0.964573
,0.964922
,0.965273
,0.965624
,0.965976
,0.966328
,0.966682
,0.967036
,0.967391
,0.967747
,0.968103
,0.968461
,0.968819
,0.969178
,0.969538
,0.969898
,0.97026
,0.970622
,0.970985
,0.971349
,0.971713
,0.972079
,0.972445
,0.972812
,0.97318
,0.973548
,0.973917
,0.974288
,0.974658
,0.97503
,0.975403
,0.975776
,0.97615
,0.976525
,0.976901
,0.977277
,0.977655
,0.978033
,0.978412
,0.978791
,0.979172
,0.979553
,0.979935
,0.980318
,0.980702
,0.981087
,0.981472
,0.981858
,0.982245
,0.982633
,0.983021
,0.983411
,0.983801
,0.984192
,0.984584
,0.984976
,0.98537
,0.985764
,0.986159
,0.986555
,0.986952
,0.987349
,0.987747
,0.988147
,0.988546
,0.988947
,0.989349
,0.989751
,0.990154
,0.990558
,0.990963
,0.991369
,0.991775
,0.992182
,0.99259
,0.992999
,0.993409
,0.993819
,0.994231
,0.994643
,0.995056
,0.99547
,0.995884
,0.9963
,0.996716
,0.997133
,0.997551
,0.99797
,0.998389
,0.99881
,0.999231
,0.999653
,1]).

APPENDIX F — *Derivation of Parameters of Beta-distribution*

For a univariate beta-distribution f(x), with parameters $\alpha > 0$ and $\beta > 0$, and defined over the range [a, b], we can derive the two parameters α and β, of the distribution (to be fit to the univariate data) if we know the mean (μ) and standard deviation (σ)of univariate data, as follows:

We know that for beta-distribution, $f(x) = \dfrac{(x-a)^{\alpha-1}(b-x)^{\beta-1}}{B(\alpha,\beta)(b-a)^{\alpha+\beta-1}}$, where $B(\alpha,\beta) = \dfrac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ is the Beta-function and Γ evaluates Gamma Function, the mean = $\mu = a + \dfrac{\alpha}{\alpha+\beta}(b-a)$ and variance = $\sigma^2 = \dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}(b-a)^2$.

Let $c_1 = (\mu - a) / (b - a) = \alpha / (\alpha + \beta)$

$c_1 = \alpha / (\alpha + \beta)$

$c_1(\alpha + \beta) = \alpha$

$(c_1 - 1)\alpha = -c_1\beta$

$(1 - c_1)\alpha = c_1\beta$

$\beta = [(1 - c_1) / c_1]\alpha$

Let $c_2 = \sigma^2 / (b - a)^2 = \alpha\beta / \{(\alpha + \beta)^2(\alpha - \beta + 1)\}$

$c_2 = [\alpha^2 \{(1 - c_1) / c_1\}] / \{[\alpha + \{(1 - c_1) / c_1\}\alpha]^2 [\alpha + \{(1 - c_1) / c_1\}\alpha + 1]\}$

$c_2 = [(1 - c_1) / c_1] / \{[1 + \{(1 - c_1) / c_1\}]^2 [(\alpha c_1 + \alpha - c_1\alpha + c_1) / c_1]\}$

$c_2 = [(1 - c_1) / c_1] / [1 / \{c_1^3(\alpha + c_1)\}] = [(1 - c_1)c_1^2] / (\alpha + c_1)$

$c_2 = [(1 - c_1) / c_1][c_1^3 / \alpha + c_1]$

$c_2\alpha + c_2 c_1 = (1 - c_1)c_1^2$

$\alpha = [(1 - c_1)c_1^2 - c_2 c_1] / c_2$

$\alpha = (c_1^2 - c_1^3 - c_2 c_1) / c_2$

$\alpha = c_1 / c_2 (c_1 - c_1^2 - c_2)$

and $\beta = [(1 - c_1) / c_1]\alpha$

APPENDIX G

List of Research Publications and Presentations

G1. Papers Presented and/or Published

1. Mukesh Kumar Rohil, "Natural Language Processing to Query a Geographical Information System (India) Knowledgebase", Map India 2000, 3rd International Conference on GIS, GPS and RS, New Delhi; April 10-11, 2000
2. Rajiv Gupta, M K Bhatt & M K Rohil, "Digital Imaging in Teaching Construction Process", 6th Annual Convention & Seminar on "Education and Training of Building Professionals", New Delhi; April 20-22, 2000 & Journal of Indian Buildings Congress; New Delhi; April 2000, 7:1, 47-53
3. Mukesh Kumar Rohil; "Exploring Possible Applications of Fuzzy Logic in Simulation of Three Dimensional Visualization using Remote Sensing Data"; Annual Symposium of Indian Society of Remote Sensing; IIT Kanpur; 20-21 Nov 2000
4. Mukesh Kumar Rohil; "An Artificial Intelligence based GIS for Optimal Routing for Accessibility in Rural Regions"; International Seminar on Accessibility & Rural Development Planning; Birla Institute of Technology and Science, Pilani; Nov 25-26, 2000
5. Rajiv Gupta and Mukesh Kumar Rohil; "Rehabilitation Of Water-Distribution Network Using GIS"; 8th Annual Convention & Seminar on "Education and Training of Building Professionals", New Delhi; June 14-16, 2002 & Journal of Indian Buildings Congress; New Delhi; June 2002, 9:1, 216-223
6. Rajiv Gupta; "Network Flow Model using Temporal GIS"; 5th Annual International Conference "Map India 2002", New Delhi; Feb 06-08, 2002 (Presented by Mukesh Kumar Rohil)

G2. Awarded Papers

1. Commendation Certificate in recognition of highly commended paper [Listed at SNo. 2 in G1 above]
2. ISRS OPTO-MECH Award for the best paper [Listed at SNo. 3 in G1 above]

G3. Papers Under Review

1. Mukesh Kumar Rohil and Rajiv Gupta; "A New Fuzzy Supervised Classification Algorithm and An Improved Explicit Fuzzy Supervised Classification Method"; International Journal of Remote Sensing
2. Mukesh Kumar Rohil and Rajiv Gupta; "A Comparative Study of Some Univariate Fuzzy Supervised Classifiers with Maximum Likelihood Classifier for Classification of Remote Sensing Data "; IEEE Transactions on Geoscience and Remote Sensing

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA

Thesis

"Efficient Implementation of some Statistical and Fuzzy Classifiers for Remote Sensing Data in Context of Geographical Information System"

Submitted By
Mukesh Kumar Rohil (1996PHXF004)

Under the Supervision of
Prof. Rajiv Gupta

ERRATA

S. No.	Page No.	Line No. From		Error Text	Correct Text
		Top	Bottom		
1.	VI	16	12	$\max_i(E)$	$\min_i(E)$
2.	5	11	23	their weaknesses	their above-mentioned weaknesses
3.	3	13	20	techniques has	techniques have
4.	6	28	5	quit	quite
5.	11	23	8	equal priori	equal a priori
6.	13	22	7	number natural	number of natural
7.	15	12	22	classification pixel	classification a pixel
8.	17	24	10	i.e. distribution free	i.e. it is distribution free
9.	19	5	30	Requires priori	Requires a priori
10.	22	22	11	... to analystLack of ...	... to analyst • Lack of ...
11.	23	15	9	data are suggested	data
12.	23	18	6	approaches has	approaches have
13.	31	20	4	= priori	= a priori
14.	44	3	23	has been	have been
15.	45	15	9	where use	where
16.	50	4	26	neither neither	neither
17.	66	N.A.	4	**(e)**	**(e)** JM DISTANCE MATRIX
18.	97	N.A.	1	**Fig. 9.5**	**Fig. 7.5**
19.	100	25	8	samples. The	samples the