

Pattern Recognition Techniques as applied to the Classification of Convective Storm Cells

BY

Mark Douglas Alexiuk

**A Thesis Submitted to the Faculty of Graduate Studies in
Partial Fulfilment of the Requirements for the Degree of**

**Master of Science
in Electrical Engineering**

**Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba**

(c) by Mark Douglas Alexiuk September, 1999



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-45019-8

**THE UNIVERSITY OF MANITOBA
FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION PAGE**

**Pattern Recognition Techniques as applied to the
Classification of Convective Storm Cells**

BY

Mark Douglas Alexiuk

**A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University
of Manitoba in partial fulfillment of the requirements of the degree
of
Master of Science**

MARK DOUGLAS ALEXIUK©1999

Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to Dissertations Abstracts International to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Abstract

This thesis investigates the role of preprocessing, rejection classes and sample relabelling in the classification of convective storm cells. This problem is representative of pattern recognition problems displaying high data dimensionality, small sample sets, and imperfect sample labelling. A battery of standard classifiers are compared using preprocessing strategies such as interquartile membership, principal and independent components. Rejection classes initiate the trade-off between improvement of performance and exhaustive classification; this is accomplished by refusing to assign class labels to samples 'near' class boundaries. Classifier specific values are used to define these boundaries. Sample relabelling is based on robust reclassification and median average deviation, fuzzy logic and probabilistic learning. This thesis uses meteorological volumetric radar data to analyse the effectiveness of these concepts. It is determined that the number of independent components to consider should not be based on a cumulative variance in principal components and that interquartile membership is more effective with real variables; rejection classes pay a high price in terms of the number of unlabelled samples although they improve classifier performance; robust reclassification consistently improves classifier performance over a broad range of classifiers. Future validation of the number of event prototypes will confirm the application of robust reclassification to this problem.

Acknowledgements

There are many people who were instrumental in helping me complete this degree. They employed, encouraged, and inspired me. I would like to thank my advisor Dr. Nicolino Pizzi and Dr. Witold Pedrycz for their patience, direction, and encouragement. I have found the staff and faculty of the Electrical and Computer Engineering department open, helpful, and energizing. I would also like to thank for their technical and meteorological advice Dave Westmore, Anthony Keck, and Dave Patrick. Of course, I wish to thank Tressa for her constancy and inspiring hope. I am indebted to NSERC for providing the financial support for this exciting academic chapter of my life.

Table of Contents

Chapter 1 Overview.....	2
1.1 Introduction.....	2
1.2 Preamble.....	2
1.3 Scope.....	2
Chapter 2 Fundamentals of Pattern Recognition.....	4
2.1 Performance Measures.....	7
2.2 Kappa Score.....	7
2.3 Bayes classifier.....	9
2.4 Pseudo-inverse.....	13
2.5 Decision Trees (ID3 / C4.5).....	15
2.6 Using Gain Ratios.....	17
2.7 Pruning Decision Trees.....	18
2.8 Fuzzy Decision Tree (FDT).....	19
2.9 K nearest neighbours (KNN).....	20
2.10 Cluster Analysis.....	21
2.11 Data partition matrices.....	22
2.12 Cluster validity.....	27
2.13 Artificial Neural Network (ANN).....	30
2.14 MultiLayer Perceptron (MLP).....	30
2.15 Rejection of Classification.....	33
2.15.1 Defining classifier dependent thresholds	34
Chapter 3 Preprocessing Techniques.....	38
3.1 Class-wise Preprocessing.....	40
3.2 Reducing Dimensionality.....	40
3.3 Principal Component Analysis (PCA).....	41
3.4 Independent Component Analysis (ICA).....	42
3.5 Improper Labelling.....	45
3.5.1 Robust Reclassification (RR)	45
3.5.2 Fuzzy Labelling (FST)	47
3.5.3 Interquartile Membership (IQ).....	48
3.6 Probabilistic Learning.....	50
3.7 Probabilistic Teacher - No a priori knowledge.....	50
3.8 Density Estimates.....	51
Chapter 4 Storm Cell Data.....	53
4.1 Radar Decision Support System.....	53
4.2 Radar Processing.....	53
4.3 Characteristics of Convective Cells.....	56
4.4 RDSS Data.....	58
4.4.1 Flags	58
4.4.2 Heuristics	58
4.5 RDSS Storm Cell Classification.....	59
4.6 Schedule Correlation.....	60
4.7 Alternate labelling strategies.....	60
4.8 Experimental Data Set.....	62
Chapter 5 Results.....	64
5.1 Generation of training and test sets.....	64
5.2 Robust Reclassification Matrices.....	65

5.3	Notes on preprocessing with different classifiers.....	66
5.4	Specific classifier parameters.....	66
5.4.1	Fuzzy decision tree	66
5.4.2	MLP parameters.....	67
5.4.3	FCM parameters.....	68
5.5	Fuzzy decision Tree - no gain.....	69
5.6	Fuzzy Decision Tree with gains ratio.....	71
5.7	Pseudo inverse.....	73
5.8	MLP results.....	75
5.9	KNN with K = 1.....	77
5.10	KNN K= 3.....	79
5.11	FCM results.....	81
5.12	Fuzzy Label Results.....	83
5.13	Rejection classes.....	85
5.13.1	FCM rejection	85
5.13.2	MLP rejection	86
5.14	Probabilistic Results.....	87
5.15	Overall comparisons - classifiers and preprocessing.....	88
Chapter 6 Conclusions and Recommendations.....		91
6.1	Conclusions.....	91
6.1.1	Comparison of classifiers.....	92
6.1.2	Comparison of preprocessing.....	92
6.2	Recommendations.....	92
6.3	Future research.....	93

List of Figures

Linear classifier with decision regions	4
Rejection regions	5
Taxonomy of classifier	6
Bayes decision region for 2 class problem	11
Bayes classifier for 2 class problem.....	13
Pseudo inverse solution to 2 class problem	15
Decision tree	16
K nearest neighbour discriminant function	21
Membership plot for 1-dimensional data	27
Clustered Data	28
Validity index for clusters	29
Cluster centers and region of influence for FCM clustering.....	29
Clustering: typicality vs membership	30
Multilayer perceptron architecture	31
ANN Processing element	33
Natural rejection region	35
Cluster entropy versus	37
Concentric distributions	39
Variance not directly related to discrimination	42
Independent and principal components for a distribution	44
Robust Reclassification $Z = 1.2$ and $Z=2$	47
Membership functions	48
Fuzzy encoding with 4 fuzzy sets.....	49
Thunderstorm evolution	57
Microburst cross-section (Caracena 1982; 1987).....	57
Convective cell cross section	59
A high precipitation (HP) storm	61
Xie-Beni Validity index (fuzzification factor =2)	68
Results for testing and training sets using FDT	69
Results for testing and training sets using	71
Results for testing and training sets using PINV	73
Results for testing and training sets using KNN1.....	77
Results for testing and training sets using KNN3	79
Results for testing and training sets using FCM, $c= 16$	81
Results for testing and training sets using MLP with fuzzy labels	83
Best PreProcessing per Classifier	88
Best Classifier per PreProcessing	89
Rejection versus Accuracy	90
Fuzzy Labelling Results	90

List of Tables

Table 1.	Kappa Score Confidence.....	8
Table 2.	Poor and moderate classifiers.....	8
Table 3.	Pascal's Wager Pay Schedule.....	12
Table 4.	Design vector for pseudo-inverse example.....	14
Table 5:	Evaluation of ID3 type decision trees.....	18
Table 6.	ID3 algorithm.....	19
Table 7.	Professions and characteristics.....	20
Table 8.	FCM Algorithm.....	24
Table 9.	Fuzzy C-Means Theorem.....	26
Table 10.	Entropy of clusters.....	36
Table 11.	FCM cluster rejection.....	36
Table 12:	Temporal and spatial characteristics of severe events.....	54
Table 13.	RDSS Derived Features.....	55
Table 14.	Number of Cells per event type.....	62
Table 15.	Most likely cells from matched groups.....	63
Table 16.	Design and test set membership.....	64
Table 17.	Preprocessing strategies.....	65
Table 18.	Robust reclassification $Z = 1$	65
Table 19.	Robust reclassification $Z = 2$	65
Table 20.	Robust reclassification $Z = 2.5$	66
Table 21.	MLP architecture.....	67
Table 22.	MLP parameters.....	67
Table 23.	FCM parameters.....	68
Table 24:	Training and test confusion matrices FDT no pre P1.....	69
Table 25:	Training and test confusion matrices FDT P2.....	69
Table 26:	Training and test confusion matrices FDT P3.....	70
Table 27:	Training and test confusion matrices FDT P4.....	70
Table 28:	Training and test confusion matrices FDT P5.....	70
Table 29:	Training and test confusion matrices FDT gain P1.....	71
Table 30:	Training and test confusion matrices FDT gain P2.....	71
Table 31:	Training and test confusion matrices FDT gain P3.....	72
Table 32:	Training and test confusion matrices FDT gain P4.....	72
Table 33:	Training and test confusion matrices FDT gain P5.....	72
Table 34:	Training and test confusion matrices PINV P1.....	73
Table 35:	Training and test confusion matrices PINV P2.....	73
Table 36:	Training and test confusion matrices PINV P3.....	74
Table 37:	Training and test confusion matrices PINV P4.....	74
Table 38:	Training and test confusion matrices PINV P5.....	74
Table 39:	Training and test confusion matrices MLP P1.....	75
Table 40:	Training and test confusion matrices MLP P2.....	75
Table 41:	Training and test confusion matrices MLP P3.....	76
Table 42:	Training and test confusion matrices MLP P4.....	76
Table 43:	Training and test confusion matrices MLP P5.....	76
Table 44:	Training and test confusion matrices KNN $k = 1$ P1.....	77
Table 45:	Training and test confusion matrices KNN $k = 1$ P2.....	77
Table 46:	Training and test confusion matrices KNN $k = 1$ P3.....	78
Table 47:	Training and test confusion matrices KNN $k = 1$ P4.....	78
Table 48:	Training and test confusion matrices KNN $k = 1$ P5.....	78

Table 49:	Training and test confusion matrices KNN $k = 1$ P1.....	79
Table 50:	Training and test confusion matrices KNN $k = 1$ P2.....	79
Table 51:	Training and test confusion matrices KNN $k = 1$ P3.....	80
Table 52:	Training and test confusion matrices KNN $k = 1$ P4.....	80
Table 53:	Training and test confusion matrices KNN $k = 1$ P5.....	80
Table 54:	Training and test confusion matrices FCM P1.....	81
Table 55:	Training and test confusion matrices FCM P2.....	81
Table 56:	Training and test confusion matrices FCM P3.....	82
Table 57:	Training and test confusion matrices FCM P4.....	82
Table 58:	Training and test confusion matrices FCM P5.....	82
Table 59:	Training and test confusion matrices MLP fuzzy labels P1.....	83
Table 60:	Training and test confusion matrices MLP fuzzy labels P2.....	83
Table 61:	Training and test confusion matrices MLP fuzzy labels P3.....	84
Table 62:	Training and test confusion matrices MLP fuzzy labels P4.....	84
Table 63:	Training and test confusion matrices MLP fuzzy labels P5.....	84
Table 64:	Comparison of rejection class with FCM.....	85
Table 65:	Test and reject confusion matrices FCM $c = 16$	85
Table 66:	Comparison of rejection class with MLP.....	86
Table 67:	Test and reject confusion matrices MLP.....	86
Table 68:	Class joint probability of mislabelled samples I.....	87
Table 69:	Class joint probability of mislabelled samples II.....	87

All the mathematical sciences are founded on relations between physical laws and laws of numbers, so that the aim of exact science is to reduce the problems of nature to the determination of quantities by operations with numbers.

- On Faraday's Lines of Force

James Clerk Maxwell

Science is built up with facts, as a house is with stones. But a collection of facts is no more a science than a heap of stones is a house.

Jules Henri Poincare

Life being all inclusion and confusion, and art being all discrimination and selection, the latter in search of the hard latent value with which it alone is concerned...

- The Spoils of Poynton

Henry James

Chapter 1 Overview

1.1 Introduction

Meteorological volumetric radar data are used to detect thunderstorms, storm events responsible for nearly all severe summer weather. Discriminating between different types of thunderstorms is a challenge due to the high dimensionality of the data, the paucity of labelled data, and the imprecision of the labels. Several classification strategies and preprocessing techniques are tested to facilitate the discrimination between four types of storm events: wind, heavy rain, tornado and hail.

1.2 Preamble

Pattern recognition is the study of the classification of physical objects by "determination of quantities"; it is the constructive collection of facts in order to best discriminate and select between objects that may superficially appear similar. To this end, many algorithms and criteria have been proposed to determine and measure the meaningful structure and organization of data. In addition, special techniques have been developed to cope with confounding circumstances under which classification must take place.

1.3 Scope

The purpose of this thesis is to explore the role of preprocessing, rejection classes, and robust labelling in classification systems where the data is subject to adverse influences. This thesis considers the case of high data dimensionality, a small data set, and imprecise data labelling.

We examine the role of explicit preprocessing in enhancing the performance of decision trees based on C4.5 [Mur94][Qui87] [Sos98], fuzzy clustering techniques [Bez81][Gat89][Gus79] and multilayer perceptron neural networks [Rip94][Koh88][Lip97]. Specifically, we demonstrate dimensionality reduction

via principal components[Kar1] and independent components [Bel95][Bel].

Robustness enhancement is effected via fuzzy set theory[Zad73], robust reclassification[Lau79][Piz97][Rou87][Zhu92a] and probabilistic learning [Agr70][Gim74][Gre80].

The remainder of this thesis conforms to the following structure.

Chapter 2 provides an overview of pattern recognition methods including Bayes classifiers, decision trees, cluster analysis, and multilayer perceptrons.

The choice of performance measures is discussed as well as the definition of rejection classes. Chapter 3 introduces standard preprocessing methods, namely principal and independent components, which maximize the variance and non-normality of dimensional sub-spaces respectively. The problem of improper labelling is addressed using fuzzy set theory [Zad65], robust

reclassification [Piz97] and probabilistic learning [Gre80]. Chapter 4

discusses radar data processing and introduces meteorological material

regarding convective storm cells. It also details the characteristics of the data used in the real-world experiments. Chapter 5 lists the results by method

and provides an overall summary of performance. Chapter 6 contains conclusions from the experiments and recommendations for future work. An appendix of the

Matlab code used in this thesis follows.

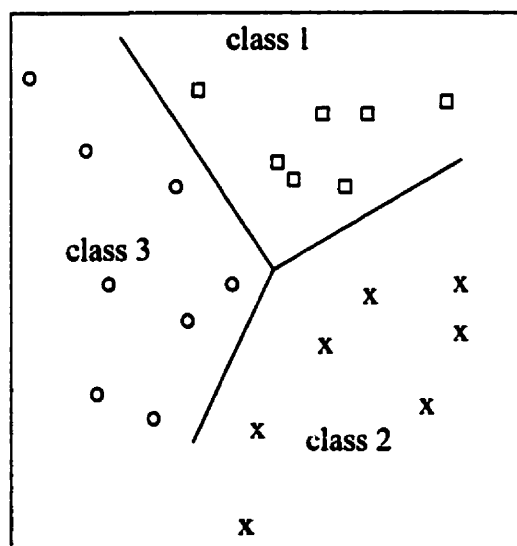
Chapter 2 Fundamentals of Pattern Recognition

Classification is the determination of a mapping ϕ from a feature space F to a classification space Ω . The feature space characterizes N objects in terms of D features, whether of quantity (empirical), quality (heuristic) or both. The classification space consists of K points, each point representing a separate class of objects. Thus,

$$\phi: F \rightarrow \omega_k, F \subset \mathcal{R}^D, k \in \{1, 2, \dots, K\} = \Omega. \quad (1)$$

(Throughout this thesis the following terms will be used interchangeably in order to highlight intuition: classifier and mapping, objects and vectors, dimensions and features.) Alternatively, classification may be viewed as the division of the feature space into K or more regions, each of which designates one of the K classes. These regions are disjoint and cover the whole feature space. For linear classification methods the boundaries of these regions are hyper planes and define decision rules (Fig.1).

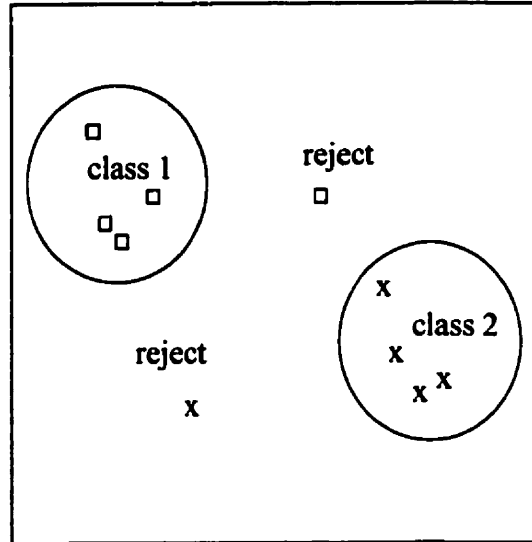
Figure 1. Linear classifier with decision regions



Some classifiers also define regions to which no class label is assigned. This may be due to lack of data from this region or to a dense, quasi-uniform

population of different classes; these regions are known as rejection regions. For example, if the distance from a sample to each cluster center exceeds some threshold, the classifier may decide to reject any label assignment (Fig.2).

Figure 2. Rejection regions



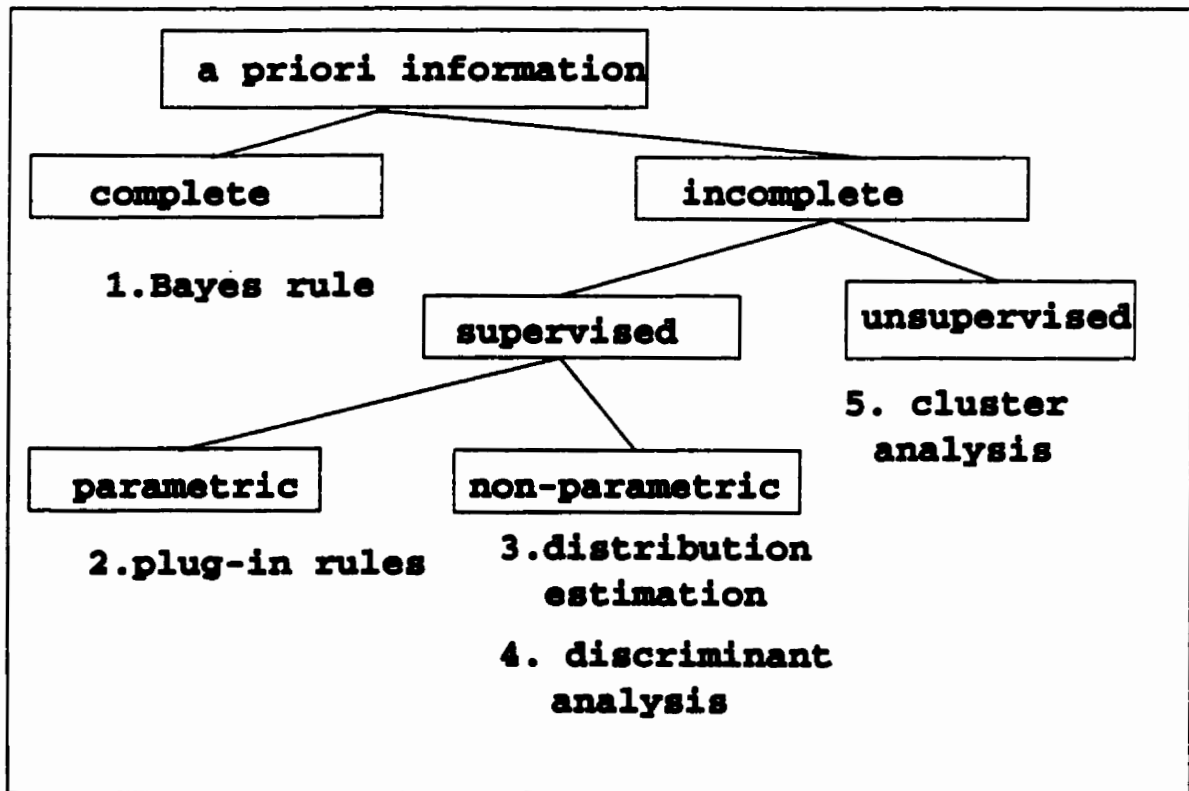
In order to construct the mapping ϕ , we must first determine a set of classifier dependent parameters that we deem to be sufficient. Typically, we divide our data set into two subsets, the design set which determines the classifier parameters, and the test set which evaluates the performance of the classifier in terms of a predictive error rate. The family of classifiers that we will use depends upon the kind of a priori knowledge that is known or justified (Fig.3).

Generally, the family of classifiers to be used is straightforward, though, for comparison, specific implementations from one or more families may be used. When comparing classifiers we should weigh the accuracy of results by the number of parameters that need to be estimated and the validity of our assumptions. The robustness of the system is sensitive to the number of parameter calculations.

Conservative design methods include v-fold cross validation (CV). Here, the

data set is decomposed into v subsets; v classifiers are designed using $v-1$ of these subsets with the remaining datum comprising the test set. Leave-one-out (LOO) cross validation is the extreme form of this method, designing N classifiers with $N-1$ design vectors with a test singleton. This is usually not used for iterative classification methods due to the high computational effort required. Often the results of several classifiers are combined into a democratic vote to assign a label to the test vectors. Classifier design may be complicated or protracted by various factors. High dimensionality of data requires a highly abstract visualization. Standard domain specific preprocessing may introduce more degrees of freedom. Mislabeled data (or outlier inclusion) may bias empirical statistics and distribution estimation.

Figure 3. Taxonomy of classifier



The following texts are recommended as excellent introductions to pattern recognition: [Dud73] [Cov91] [How90] [Pao89] [Krz88] [Kan82] [Kit86] [You86].

2.1 Performance Measures

Typically, the performance measure used for classifiers is the number of correctly identified vectors divided by the total number of vectors. Viewed as a function of the confusion matrix $CM = [cm_{ij}]$ (where diagonal entries $[cm_{ii}]$ represents correctly classed vectors, and off-diagonal entries $[cm_{ij}]$ denote class ω_i vectors mistaken for class ω_j vectors, $1 \leq i, j \leq K$)

$$P_o = \frac{1}{N} \sum_{i=1}^K cm_{ii}. \quad (2)$$

2.2 Kappa Score

However, since each class may have a different number of training vectors, we must differentiate between a poor classifier that assigns most of the vectors to the larger class and classifiers that assign most of the vectors in any one class to the correct class. In other words, we want to eliminate agreement due to chance. This agreement is

$$P_c = \frac{\sum_{i=1}^K \left(\sum_{j=1}^K cm_{ij} \sum_{j=1}^K cm_{ji} \right)}{N^2}. \quad (3)$$

The kappa score [Piz97] incorporates this chance agreement as

$$\kappa = \frac{P_o - P_c}{1 - P_c}. \quad (4)$$

Note: A kappa score of 0 denotes agreement strictly due to chance, while $\kappa > 0$, ($\kappa < 0$) denotes an agreement better (worse) than chance. A de facto standard measure of confidence is shown in Table 1. Alternate performance measures are possible by extending the binary classification performance (classed correctly or incorrectly) into a continuous degree of classification. Use of smooth kernel estimators [Paw88] quantifies the degree of mis-classification

and places less of a penalty on mis-classified objects that are near the class boundary.

Table 1. Kappa Score Confidence

Confidence	κ range
poor	$\kappa=0$
slight	$0 < \kappa \leq 0.2$
fair	$0.2 < \kappa \leq 0.4$
moderate	$0.4 < \kappa \leq 0.6$
substantial	$0.6 < \kappa \leq 0.8$
almost perfect	$0.8 < \kappa < 1$
perfect	$\kappa = 1$

Consider the confusion matrices in Table 2 generated by two hypothetical classifiers. Based upon traditional performance measures, P_1 and P_2 , the second classifier appears only marginally more accurate. However, when the kappa scores, κ_1 and κ_2 , are computed, a significant difference is seen.

Table 2. Poor and moderate classifiers

Classifier1	assigned ω_1	assigned ω_2	Classifier2	assigned ω_1	assigned ω_2
class ω_1	85	15	class ω_1	85	15
class ω_2	15	5	class ω_2	5	15

$$P_{01} = \frac{85 + 5}{85 + 15 + 15 + 5} = 0.75$$

$$P_{02} = \frac{85 + 15}{85 + 15 + 15 + 5} = 0.83$$

$$P_{C1} = \frac{(100)(100) + (20)(20)}{(120)^2} = 0.7222 \quad P_{C2} = \frac{(100)(90) + (20)(30)}{(120)^2} = 0.67$$

$$\kappa_1 = \frac{0.75 - 0.722}{1 - 0.722} = 0.1001$$

$$\kappa_2 = \frac{0.83 - 0.67}{1 - 0.67} = 0.5$$

2.3 Bayes classifier

The Bayes classifier [Dud73] takes advantage of probability density functions for each class

$$p(x|\omega_j) = \frac{p(\omega_j|x)}{p(\omega_j)} p(x) . \quad (5)$$

To classify an object we simply determine the conditional probability of the sample x for each class ω_k $1 \leq k \leq K$. The maximum conditional probability denotes the most likely class and determines the assigned label. Given an D -dimensional normal distribution, $x \sim N_D(m, \Sigma)$,

$$p(x) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-m)\Sigma^{-1}(x-m)\right), \quad (6)$$

where

$$m = E\{X\}, \quad (7)$$

and

$$\Sigma = E\{(X-m)(X-m)^T\} = \text{cov}(X). \quad (8)$$

The discriminant function for class ω_k in logarithmic form is

$$\begin{aligned} g_k(x) &= \ln(p(x|\omega_k)) + \ln(p(\omega_k)) \\ &= \left(-\frac{1}{2}\right)(x-m_k)\Sigma_k^{-1}(x-m_k) - \frac{1}{2}\ln(\Sigma_k) + \ln(p(\omega_k)). \end{aligned} \quad (9)$$

Thus, the Bayes Rule is:

Decide $x \in \omega_k$ if

$$\ln(p(x|\omega_k)) + \ln(p(\omega_k)) > \ln(p(x|\omega_j)) + \ln(p(\omega_j)), j \neq k. \quad (10)$$

For Gaussian distributions, second order assumptions define alternately a linear or quadratic classifier. For equal diagonal class covariance,

$\Sigma_k = \sigma^2 I$, $1 \leq k \leq K$, the linear classifier follows:

$$g_k(x) = \left(-\frac{1}{2\sigma^2}\right)(x-m_k)^T(x-m_k) + \ln\left(p(\omega_k)\right), \quad (11)$$

and we decide $x \in \omega_k$ if

$$\frac{\|x-m_k\|^2}{2\sigma^2} \leq \frac{\|x-m_i\|^2}{2\sigma^2} + \ln\left(\frac{p(\omega_k)}{p(\omega_i)}\right). \quad (12)$$

For equal a priori class probabilities, the Bayes rule reduces to a minimum distance classifier. We also have a linear classifier for equal class

covariance matrices. If $\Sigma_k = \Sigma$, $1 \leq k \leq K$, (correlated features)

$$g_k(x) = \left(-\frac{1}{2}\right)(x-m_k)^T \Sigma^{-1}(x-m_k) + \ln\left(p(\omega_k)\right) = w_k^T x + w_{k0}, \quad (13)$$

where

$$w_k = \Sigma m_k, \quad (14)$$

and

$$w_{k0} = \left(-\frac{1}{2}\right)m_k^T \Sigma^{-1} m_k + \ln\left(p(\omega_k)\right). \quad (15)$$

Note: The quantity $(a-b)^T \Sigma^{-1}(a-b)$ is the Mahalanobis distance between vectors a and b. If all classes have equal a priori probability, $p(\omega_k) = p(\omega_i) \forall i$ we have a minimum Mahalanobis distance classifier.

Unequal class covariance defines a quadratic classifier;

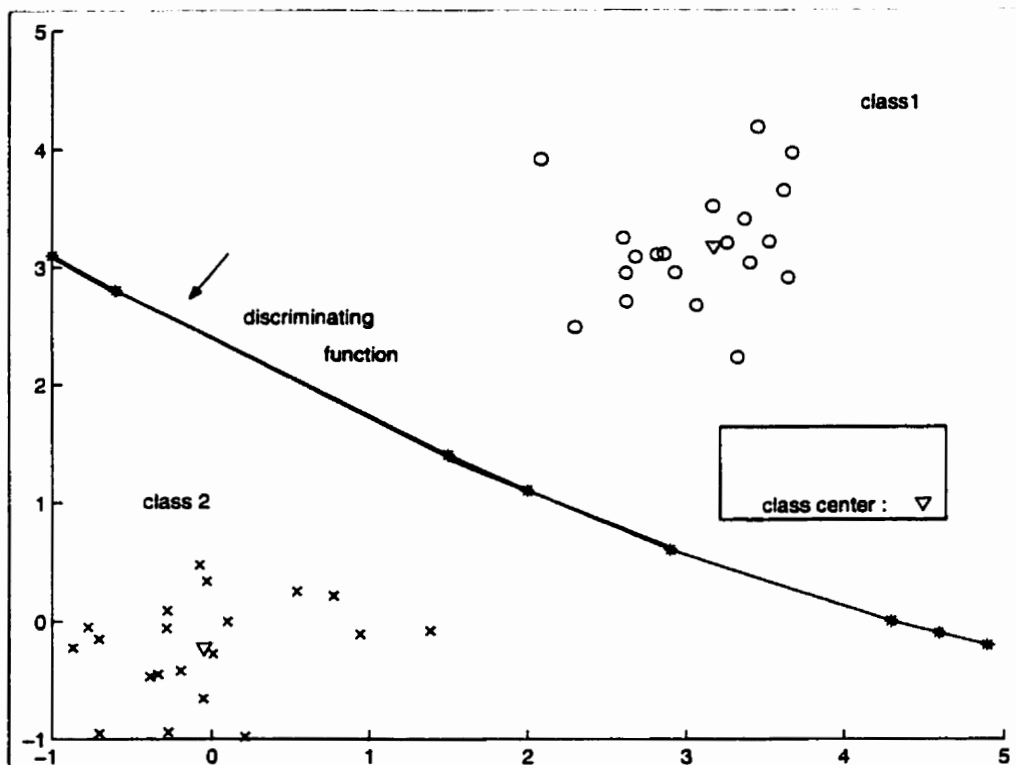
$$\begin{aligned} g_k(x) &= \left(-\frac{1}{2}\right)(x-m_k)^T \Sigma_k^{-1}(x-m_k) - \frac{1}{2} \ln\left(|\Sigma_k|\right) + \ln\left(p(\omega_k)\right) \\ &= x^T W_k x + w_k^T x + w_{k0} \end{aligned} \quad (16)$$

The added matrix is

$$W_k = \left(-\frac{1}{2}\right) \Sigma_k^{-1} . \quad (17)$$

Consider Figure 5 which shows the quadratic decision boundary for a two class problem. Note how the covariance affects the boundary, sweeping around the class with lower variance. If we suppose that the labels are switched for s samples near the boundary, the overall effect on the means and covariance matrix may be small. However, the classification rate will have dropped significantly. For any classifier, the classification performance (error rate) has a lower bound of the Bayes rate (risk). This theoretical limit is the optimal performance of any classifier for a given distribution.

Figure 4. Bayes decision region for 2 class problem



However, in order to implement Bayes type classifiers, the a priori probability density function (pdf) of the vectors must be known. Empirical estimation of the pdf may be estimated using the design set and kernel estimators e.g. Parzen windows.

The Parzen density estimate [Jeo94] is the sum of kernel functions ϕ placed at each design sample x . The kernel functions are usually chosen to have such mathematically tractable properties as differentiability or continuity. For a design set X_k with N_k samples of dimension D , we have the Parzen estimate at y of

$$f_k(y) = \frac{1}{N_k h^D} \sum_{x \in X_k} \phi\left[\frac{x-y}{h}\right]. \quad (18)$$

Here h is a smoothing parameter that relates to the window size, the space under consideration for the estimate of each sample.

Recall that the risk for a classification procedure is

$$R(\psi) = \sum_{k=1}^K p_k \int L(\psi(x)) f_k(x) d\mu(x) .. \quad (19)$$

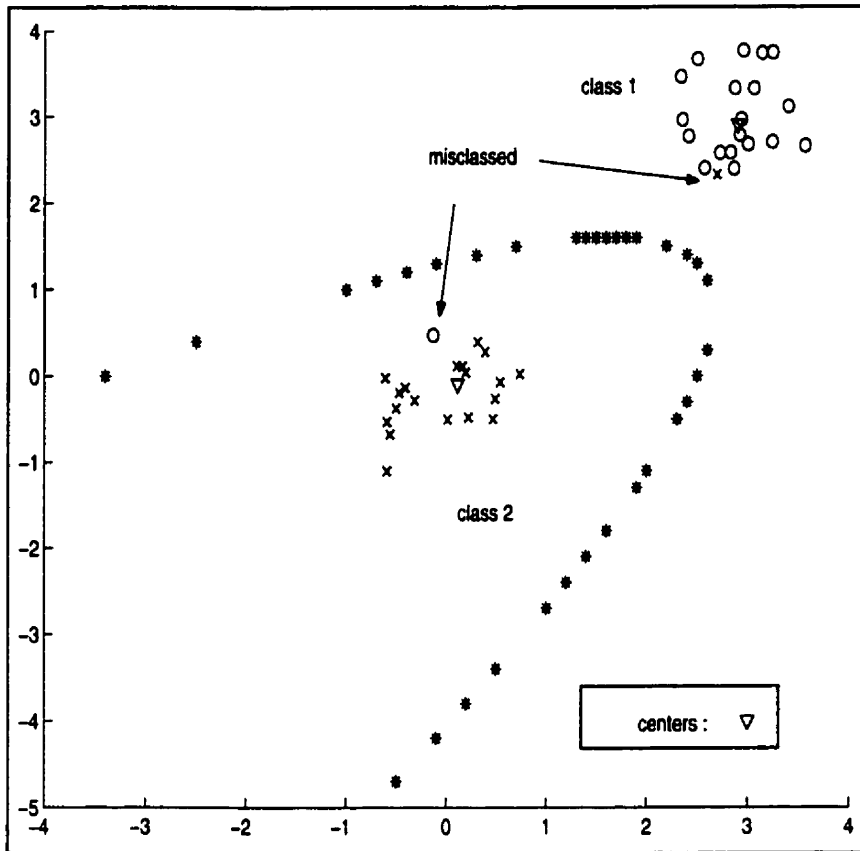
The cost or loss matrix defines the relative penalties for mis-classification and is produced by multiplying the probability of the event by the benefit of each event. This allows the effects of decisions to be weighed. We consider only the 0-1 loss function defined by $L_{ij} = 0 \quad i = j$

($L_{ij} = 1 \quad i \neq j$) and denote the Bayes risk as R_0 . Note however that some cost matrices are independent of probabilities. Recall Pascal's wager, where for any $p = \text{probability}(\text{Pascal's God exists}) > 0$ and the pay schedule of Table 3, his decision to live righteously is justified.

Table 3. Pascal's Wager Pay Schedule

	$\exists GOD$	$\exists \neg GOD$
live a righteous life	∞	-small
live a wicked life	$-\infty$	+small

Figure 5. Bayes classifier for 2 class problem



2.4 Pseudo-inverse

A classifier may be considered in terms of matrix operations where the N D -dimensional design vectors \mathbf{X} , arranged in a N by D matrix, are multiplied by a D by K matrix \mathbf{W} such that the resulting matrix is the N by K classification vector \mathbf{B} . The matrix \mathbf{W} is the matrix product of the inverse of the design matrix \mathbf{X} , a D by N matrix, and the N by K target matrix \mathbf{B} . The target vector of a sample is a row of zeros with a one in the k^{th} column to designate class ω_k .

$$\mathbf{XW} = \mathbf{B} \quad (20)$$

then

$$\mathbf{W} = \mathbf{X}^{-1}\mathbf{B}. \quad (21)$$

This results in an algebraically appealing solution that is computationally inexpensive. However, if \mathbf{X} is not square it is then overdetermined (more

equations than unknowns). The more general equation then applies;

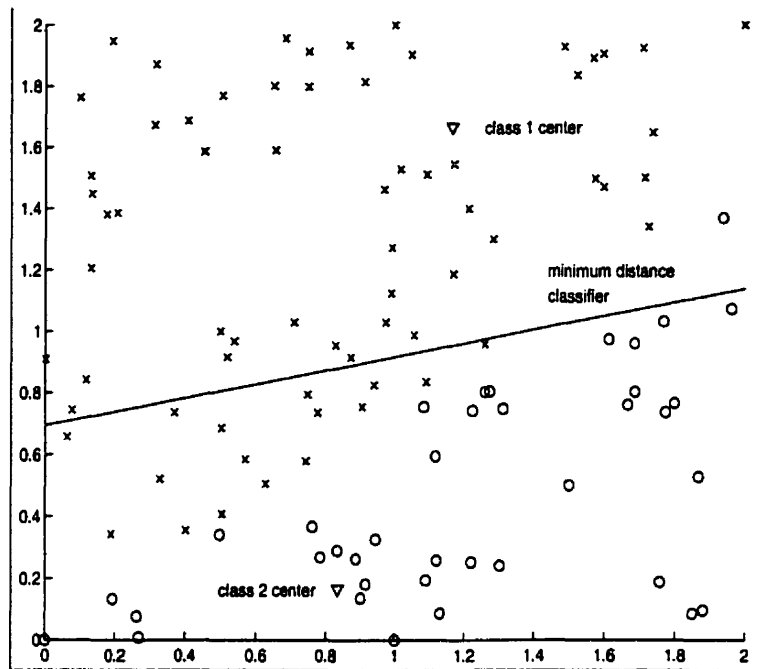
$$\mathbf{W} = [\mathbf{X}\mathbf{X}^T]^{-1}\mathbf{B} = \mathbf{X}\mathbf{B} \quad (22)$$

\mathbf{W} is known as the pseudo-inverse (PINV) of \mathbf{X} . The PINV is calculated using singular value decomposition. The pseudo-inverse takes into account only global information about the relationship between the feature vectors and the classification. Despite this we will use pseudo-inverse as a linear benchmark for the other classifiers. Figure 6 shows the pseudo-inverse classification for random points in $[0,2]^2$ given the design vectors in Table 4; it also shows the discriminating function obtained by taking the class means and assigning labels on a nearest center approach.

Table 4. Design vector for pseudo-inverse example

dimension 1	dimension 2	class
0	0	1
1	0	1
1.5	0.5	1
2	2	2
1	2	2
0.5	1	2

Figure 6. Pseudo inverse solution to 2 class problem



2.5 Decision Trees (ID3 / C4.5)

A decision tree is a set of inductive rules that discriminate objects based on a series of questions regarding feature values. ID3 and C4.5 are supervised learning algorithms introduced by Quinlan [Qui93][Qui87] for generating decision trees. ID3 concerns itself with non-numeric (categorical/qualitative) data while C4.5 extends ID3 to include continuous features.

Structure

A set of nodes is linked in a tree structure where each node is associated with a question or a label, depending on whether it has a successor or not. Denote nodes that have a successor as question nodes and terminal/leaf nodes as class nodes. Each question node determines its successor based on the answer to its question. In this fashion a unique class node in the tree is reached; not all questions are posed for each datum. The derivation of the structure of the decision tree algorithms is based upon the uncertainty of each class given a specific feature value. Each question partitions the space with a hyper plane; each path from the root node to a leaf node defines a

convex class region.

Figure 7. Decision tree

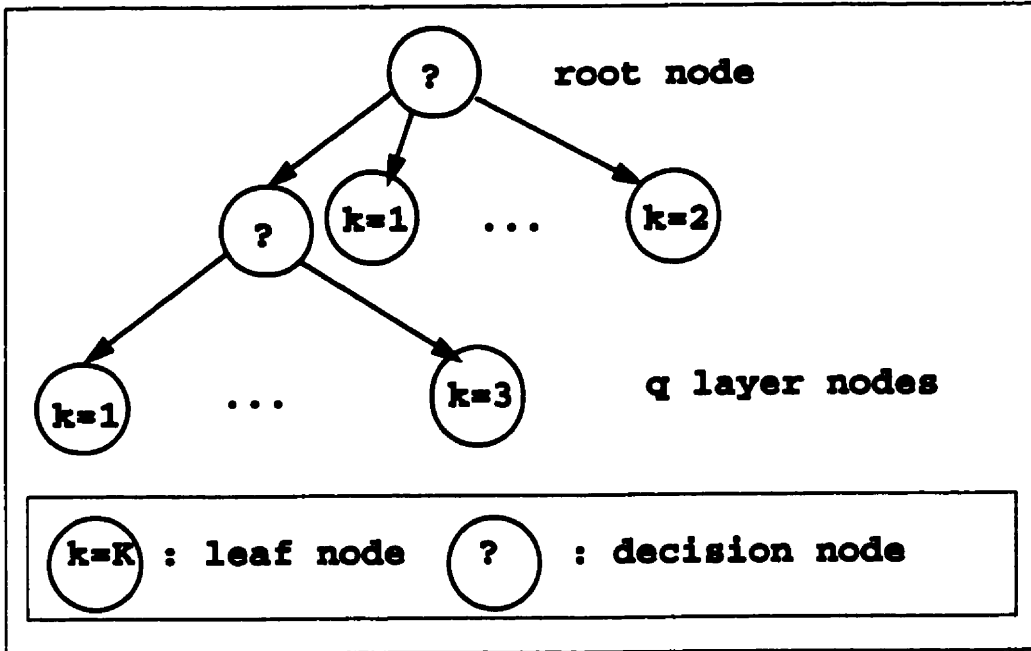


Figure 7 displays the process of classification used in a decision tree. Each node determines which side of its hyperplane the feature value is on; this ultimately decides the sample's assigned label.

The notion of entropy is essential for the development of the discrimination made at each node. We follow its definition with the decision tree generating algorithm. Throughout this section, \ln denotes the logarithm to base 2.

Definition: Entropy is the measure of uncertainty in a distribution

$P = [p_1, p_2, \dots, p_K]$ of K classes where p_k is the probability that the sample

is from class ω_k . Here P may be an empirical probability $p_k = \frac{N_k}{N}$ estimated

from the data. The entropy of P is

$$\eta(P) = - \sum_{k=1}^K p_k \ln(p_k) \quad . \quad (23)$$

(If we let $-\ln(p_k)$ define the information associated with the occurrence of

a class ω_k sample, then $\eta(P)$ is the average amount of information per class

occurrence.) For a uniform distribution, entropy is maximized; each possibility is equally likely and the outcome most uncertain. Entropy may also be thought of as the information conveyed by a distribution. For example, if there are K equally probable events, then the probability p of each is $1/K$ and the information conveyed by each event is $-\ln(p) = \ln(K)$. Then if there are 16 events, $\ln(16) = 4$ and we need 4 bits to identify each event.

Gain is often used to rank features for decision tree nodes and is defined as the difference between the uncertainty of the distribution P and the uncertainty of P given the knowledge of feature Y . Then

$$\Delta\eta(P, Y) = \eta(P) - \eta(P|Y) \quad . \quad (24)$$

The depth of the above tree is directly related to the number of hyper planes needed to define a class region and the number of operations needed to classify a test vector [Abr63].

2.6 Using Gain Ratios

Note that gain favours attributes that have a large number of values. For a distribution A that has a distinct value for each record, then $\eta(A|X)$ is 0, thus $\gamma(A, X)$ is maximal. We define the gain ratio γ as an extension of the entropy gain;

$$\gamma(X, A_k) = \frac{\Delta\eta(X, A_k)}{\zeta(X, A_k)} \quad , \quad (25)$$

$$\zeta(X, A) = I \left[\left[\frac{|A_1|}{|X|}, \frac{|A_2|}{|X|}, \dots, \frac{|A_Q|}{|X|} \right] \right] \quad . \quad (26)$$

$\zeta(X, A)$ is the information due to the split of X based on the value of the categorical attribute A .

2.7 Pruning Decision Trees

In the design of the decision tree, we usually continue until the nodes contain members from only one class. However, depending upon the distribution, this process may result in a tree with long and uneven paths. The tree may then be generalized by some form of pruning where two or more leaf nodes are collapsed making the higher level node a leaf node. Pruning increases the error in classification but also makes the decision tree more stable. Multiple trees may be designed using different pruning methods with the majority deciding the classification.

Table 5: Evaluation of ID3 type decision trees

Advantage	straightforward to automate
Disadvantage	for new data, the tree structure must essentially be recalculated

Continuous features are intractable with ID3 since each value may be unique. C4.5 incorporates continuous features by creating a partition based on relative magnitude to a specific feature value.

For the continuous dataset X , $x, u \in X$, C4.5 calculates the entropy between the sets:

$$A = \{u | \left(\frac{u}{x} > 1\right)\} \text{ and } B = \{u | \left(\frac{u}{x} < 1\right)\}, \forall x. \quad (27)$$

The partition with the lowest entropy is chosen.

Table 6. ID3 algorithm

Step 1. Calculate initial entropy of the training set, T , containing N_k vectors from each class ω_k , $1 \leq k \leq K$,

$$\eta(T) = \sum_{k=1}^K -\left(\frac{N_k}{N}\right) \log\left(\frac{N_k}{N}\right) = \sum_{k=1}^K -p_k \log(p_k). \quad (28)$$

Step 2. Select a feature to serve as the root node/leaf node of the tree.

i For each feature/attribute A_p , $p = 1 \dots P$, partition the data set relative to the J attribute values of a_{pj} . Denote the number of vectors in any one attribute value partition as n_{pj} .

ii Denote the number of vectors in attribute value partition a_{pj} in class k , $k = 1 \dots K$ as $n_{pj}^{(k)}$. Evaluate the entropy for this partition as

$$\eta\left(T\left(A_{p,j}\right)\right) = \sum_{k=1}^K -\left(\frac{n_{pj}^{(k)}}{n_{pj}}\right) \log\left(\frac{n_{pj}^{(k)}}{n_{pj}}\right). \quad (29)$$

The total entropy over all attribute values is:

$$\eta\left(T\left(A_p\right)\right) = \sum_{j=1}^J \sum_{k=1}^K -\left(\frac{n_{pj}}{\sum_j n_{pj}}\right) \left(\frac{n_{pj}^{(k)}}{n_{pj}}\right) \log\left(\frac{n_{pj}^{(k)}}{n_{pj}}\right). \quad (30)$$

iii Select the feature A_p that yields the greatest decrease in entropy:

$$\eta(T) - \eta\left(T|A_p\right) = \Delta\eta_p. \quad (31)$$

iv Feature A_p is now the root of the tree, if we are on first iteration, or one of J leaf nodes.

Step 3. Iterate Step 2 for each leaf node of the tree until all leaf partitions contain vectors from only one class (or, equivalently, until the entropy goes to zero).

2.8 Fuzzy Decision Tree (FDT)

The FDT algorithm [Sos98] has been modified to deal with continuous data

slightly differently from C4.5; the range is quantized into Q equal partitions and the data represented symbolically by the range interval it lies in. The partition with the highest gain, or gains ratio is then chosen. A shortcoming of the current version of FDT is that the intervals are uniform, and the number of intervals constant for all feature; intervals may have no entries for highly clustered data and the number of partitions will tend to be sub-optimal for some features. Consider the entropy of the following distribution.

Table 7. Professions and characteristics

	beard	glasses
doctors	1	5
lawyers	3	3

Note the conditional probabilities

$P(\text{profession}=\text{doctor}|\text{beard}) = 1/4$
 $P(\text{profession}=\text{doctor}|\text{glasses}) = 5/8$
 $P(\text{profession}=\text{lawyer}|\text{beard}) = 3/4$
 $P(\text{profession}=\text{lawyer}|\text{glasses}) = 3/8$

The uncertainty of a man's profession for both characteristics is:

$\eta(\text{profession}|\text{beard}) = -(1/4)\ln(1/4) - (5/8)\ln(5/8) = 0.9238$
 $\eta(\text{profession}|\text{glasses}) = -(3/4)\ln(3/4) - (3/8)\ln(3/8) = 0.8419.$

We are more likely to be correct in our guess of a man's profession if we know whether he wears glasses rather than a beard.

2.9 \mathcal{K} nearest neighbours (KNN)

A majority vote of the \mathcal{K} nearest data vectors may also be used as a classifier.

Each test vector is assigned a label based on the class most represented by

the \mathcal{K} nearest neighbours. That is, decide $x \in \omega_i$ iff

$$\sum_{\kappa=1}^{\mathcal{K}} \chi(\omega_{\kappa}=i) > \sum_{\kappa=1}^{\mathcal{K}} \chi(\omega_{\kappa}=j), \quad (32)$$

where ω_{κ} is the class of the κ^{th} neighbour and $\chi(x)$ is the characteristic function

$$\chi(x) = \begin{cases} 1, & (x = \text{TRUE}) \\ 0, & (x = \text{FALSE}) \end{cases} \quad (33)$$

The KNN method is a non-parametric density estimator and relates to the nature of the true pdf. This voting scheme may be modified with a weighting function such that the s consecutive samples of a class will dominate t samples from another class. Various weight families may be of interest. We then decide $x \in \omega_i$ iff

$$\sum_{k=1}^{\mathcal{K}} w_k \chi(\omega_k = i) > \sum_{k=1}^{\mathcal{K}} w_k \chi(\omega_k = j). \quad (34)$$

Usually \mathcal{K} is odd to prevent ties. This method achieves 100% training in the trivial case $\mathcal{K} = 1$. Although this classifier uses a linear function of data, it generates a complex decision region. It is also non-iterative and thus easy to update. For further reference see [Cov67].

Figure 8. K nearest neighbour discriminant function



2.10 Cluster Analysis

Cluster analysis refers to any of a large number of algorithms that identify clusters or modes in data sets[Eve93]. This includes both iterative algorithms for quantitative data and graph theoretic methods for qualitative data

(minimum spanning tree). Both forms assign d -dimensional unlabelled data vectors X ,

$$X = \{x_1, x_2 \dots x_N\}^T \subset \mathbb{R}^D, \quad (35)$$

to C different cluster centers V ,

$$V = \{v_1, v_2 \dots v_C\}^T \subset \mathbb{R}^D, \quad (36)$$

based on a metric defined for the space. We concern ourselves solely with the iterative algorithms. These algorithms begin by initializing C centers in space. Iteration follows as the centers are relocated based on neighbouring samples. The distance from each data point to every center defines weights which allow closer samples to have a larger impact on the relocation.

Iterative methods may be distinguished by the exact method of center relocation (dually, objective functionals determine the formulation of the equations and may be used to discriminate): using crisp membership, fuzzy membership [Bez81][Gat89][Gus79][Pal95] or possibilistic typicality [Geb][Kri93][Kri96]. Intuitively, there are some satisfying results from clustering algorithms. For one cluster center and the Euclidean metric, the hard C -means algorithm converges to the center of gravity of the dataset (minimizing the mean square error). Since the number of cluster centers to use is required for most algorithms validity measures to confirm the number of cluster centers is essential.

Generally, one hopes to learn correct data prototypes and identify natural subdivisions or modes of the probability density function. Clustering is also important for vector quantizers where an optimal many-to-one (region-to-point) map is sought for data compression. We now define some terms to elucidate the differences between three iterative clustering algorithms.

2.11 Data partition matrices

For N data vectors and C cluster centers, we define a data partition matrix to be the $N \times C$ matrix $U = [u_{nc}]$ where $[u_{nc}]$ is the membership of the n^{th} vector in the c^{th} cluster center [Pal95]. Note that there are three sets of partition matrices:

the crisp (hard) c partition matrix,

$$M_{hcn} = \left\{ U \in M_{fcn} \mid \left(u_{nc} \in \{0, 1\} \forall k \forall i \mid \left(\sum_{c=1}^C u_{nc} = 1 \forall k \right) \right) \right\}, \quad (37)$$

the fuzzy (or constrained possibilistic) partition matrix,

$$M_{fcn} = \left\{ U \in M_{pcn} \mid \left(\sum_{c=1}^C u_{nc} = 1 \forall k \right) \right\}, \quad (38)$$

and the possibilistic partition matrix,

$$M_{pcn} = \left\{ U \in \mathcal{R}^D \mid \left(\left(0 \leq u_{nc} \leq 1 \forall i \forall k \right); \left(u_{nc} > 0 \forall c \forall n \right); \left(0 < \sum_{n=1}^N u_{nc} < N \forall c \right) \right) \right\}. \quad (39)$$

Note that HCM assigns each datum exclusively to one cluster.

FCM generalizes HCM in that it assigns a degree of membership

in each cluster for all data points. While both HCM and FCM may be viewed as "pure" data partitioning matrices in that the membership values of each datum must sum to one, PCM is a mode-seeking algorithm that measures the typicality (possibility) of a vector being assigned to each cluster. The notion of typicality is concerned with a slightly different functional and attempts to maximize membership while minimizing the unconstrained FCM functional.

Table 8. FCM Algorithm

given: unlabelled data X , $|X| = N$

number of clusters c where $1 \leq c \leq N$

fuzzification exponent $m > 1$

iteration limit T

error termination criterion $\epsilon > 0$

objective function J_m e.g. inter-cluster MSE

objective function norm e.g. Euclidean distance

error norm e.g. $E_t = \|V_t - V_{t-1}\|$

Step 1 Initialization: choose C initial cluster centers $V_0 = [v_1 \dots v_C]^T$.

Step 2. Iterate: for $t = 1$ to T

Update memberships U_t using V_{t-1} with (45)

Update cluster centers V_t using U_t with (46)

if $E_t = \|V_t - V_{t-1}\| < \epsilon$

set $U_{final} = U_t$, $V_{final} = V_t$, $t = T$

end

Step 3. Output final centers V_{final} with memberships U_{final} .

It should be noted that PCM has the potential to discover coincident cluster centers. This is not necessarily a defect [Kri96] and may serve as a validity measure on the number of cluster centers. One may use PCM to discover one mode at a time and remove each mode from the data set as the PCM algorithm converges.

The possibilistic objective function is

$$\min \left\{ \left(J_{m, \beta} (U, V; X) = \sum_{c=1}^C \sum_{n=1}^N (u_{nc})^m \|x_n - v_c\|_A^2 \right) + \sum_{c=1}^C \beta_c \sum_{n=1}^N (1 - u_{nc})^m \right\}, \quad (40)$$

where $\beta_i > 0$ for all i is the bandwidth (resolution) parameter and is estimated as

$$\beta_c = Q^c \frac{\sum_n u_{nc}^m \|x_n - v_c\|}{\sum_n u_{nc}^m} \quad (41)$$

with the interpretation that β_c is proportional to the average fuzzy intra-cluster distance of cluster i . Generally $Q = 1$.

The fuzzy objective function,

$$\left(\begin{matrix} \min \\ (U, V) \end{matrix} \right) \left\{ J_m (U, V; X) = \sum_{n=1}^N \sum_{c=1}^C (u_{nc})^m \|x_n - v_c\|_A^2 \right\}, \quad (42)$$

is minimized subject to

$$\sum_{c=1}^C u_{nc} = 1, \quad \forall c, 1 \leq c \leq C. \quad (43)$$

(The sum of weighted distances from vectors to cluster centers is minimized with the memberships summing to 1.)

The hard C-means objective function is

$$\left(\begin{matrix} \min \\ (A, V) \end{matrix} \right) \left\{ J(A, V; X) = \sum_{n=1}^N \sum_{c=1}^C a_n \|x_n - v_c\|_A^2 \right\}, \quad (44)$$

where $A = [a_1, a_2, \dots, a_N]$ and $a_n \in \{0, 1\}$.

Note that initialization of cluster centers may be achieved by choosing random points in the feature space or selecting random design points.

Table 9. Fuzzy C-Means Theorem

If $D_{ncA} = \|x_n - v_c\|_A > 0$ for all i and k , then $(U, V) \in M_{fcn} \times \mathcal{R}^D$ may

minimize $J_m(U, V; X)$ only if, when $m > 1$,

$$u_{ni} = \left[\sum_{c=1}^C \left(\frac{D_{niA}}{D_{ncA}} \right)^{\frac{2}{m-1}} \right]^{-1}, \quad (45)$$

and

$$v_c = \frac{\sum_{n=1}^N (u_{nc})^m (x_n)}{\sum_{n=1}^N (u_{nc})^m}. \quad (46)$$

Analysis of the limiting cases of the fuzzy exponent show that as the fuzzy

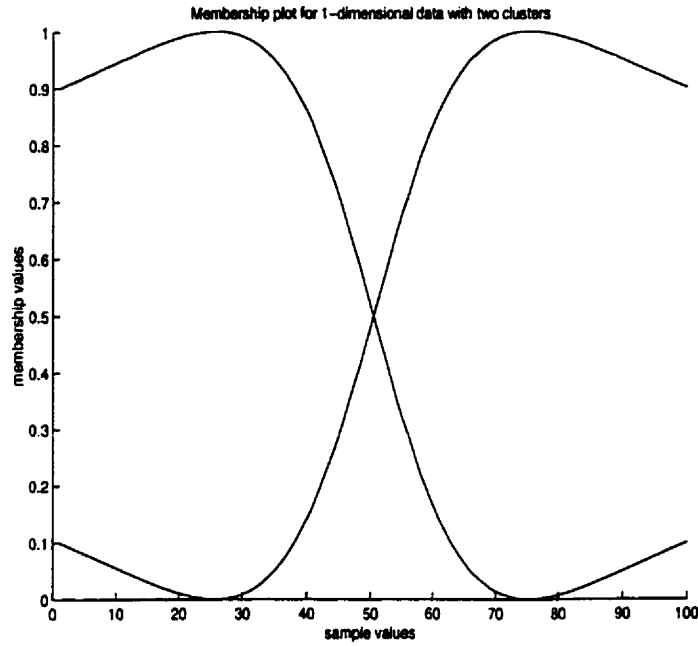
exponent approaches 1 FCM converges to HCM ($\lim_{m \rightarrow 1} \left| \{u_{nc} | (u_{nc} > 0)\} \right| = 1$,

$\lim_{m \rightarrow 1} \left| \{u_{nc} | (u_{nc} = 0)\} \right| = N-1$) and that an infinite fuzzy exponent leads to

a uniform membership value for each data point $\lim_{m \rightarrow \infty} u_{nc} = \frac{1}{N} \forall n \forall c$.

See [Gat89] and [Gus79] for modified FCM algorithms that allow elliptic clusters (a modified Euclidean distance with a inverse covariance matrix and its determinant enter as additional factors in distance equation) and [Kla] for fuzzy rule base derivation.

Figure 9. Membership plot for 1-dimensional data



2.12 Cluster validity

Various indices exist to confirm the number of clusters inherent in the data.

The Xie-Beni validity index,

$$Val_{XB}(U, V; X) = \frac{\sum_{c=1}^C \left(\sum_{n=1}^N u_{nc}^2 \|x_n - v_c\|^2 \right)}{N \left(\min \{ \|v_c - v_i\|^2 \} \right)} = \left[\frac{\left(\frac{\sigma}{N} \right)}{sep(V)} \right], \quad i \neq j, \quad (47)$$

appears to be more robust than other indices [Pa195]. The terms are interpreted as the minimum separation (sep) between cluster centers,

$$sep(V) = \min \{ \|v_c - v_i\|^2 \}, \quad (48)$$

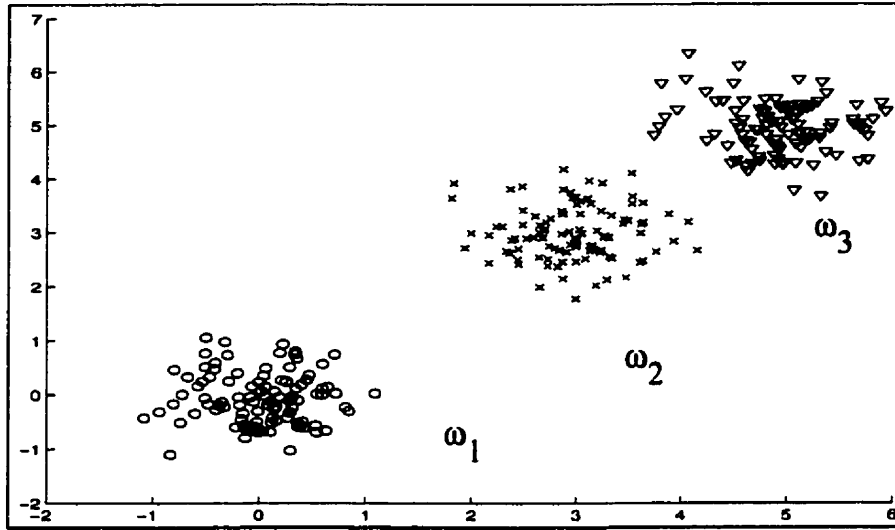
and the membership weighted covariance from the centers,

$$\sigma(U, V; X) = \sum_{c=1}^C \left(\sum_{n=1}^N u_{nc}^2 \|x_n - v_c\|^2 \right). \quad (49)$$

The following figures, (Fig.10-12), show the Xie-Beni validity index applied to determining the number of clusters in a synthetic data set. Note that the

index value is relative to each data set; the lowest index indicates the minimization of the above functional (47).

Figure 10. Clustered Data



It is shown that 3 clusters best represent the data and that 2 clusters represent the data 'better' (in the Xie-Beni sense) than 4 clusters. Figure 12 shows areas of influence for each cluster - samples falling within these regions contribute most to the cluster center location. When the number of clusters involved in FCM exceeds the number of inherent clusters (based on some index or design) more than one cluster center will be associated with one or more classes. The distribution of centers amongst the classes is a function of the random initialization. For this example, further iterations would produce different center distributions with a relatively same index.

Figure 11. Validity index for clusters

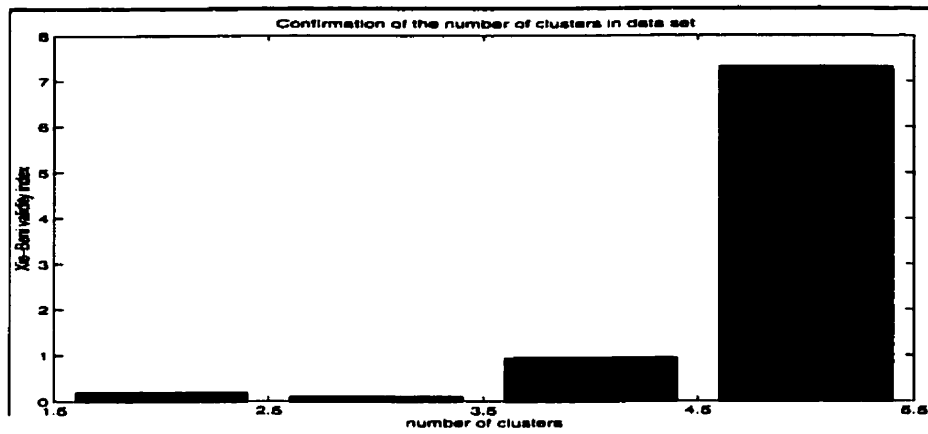
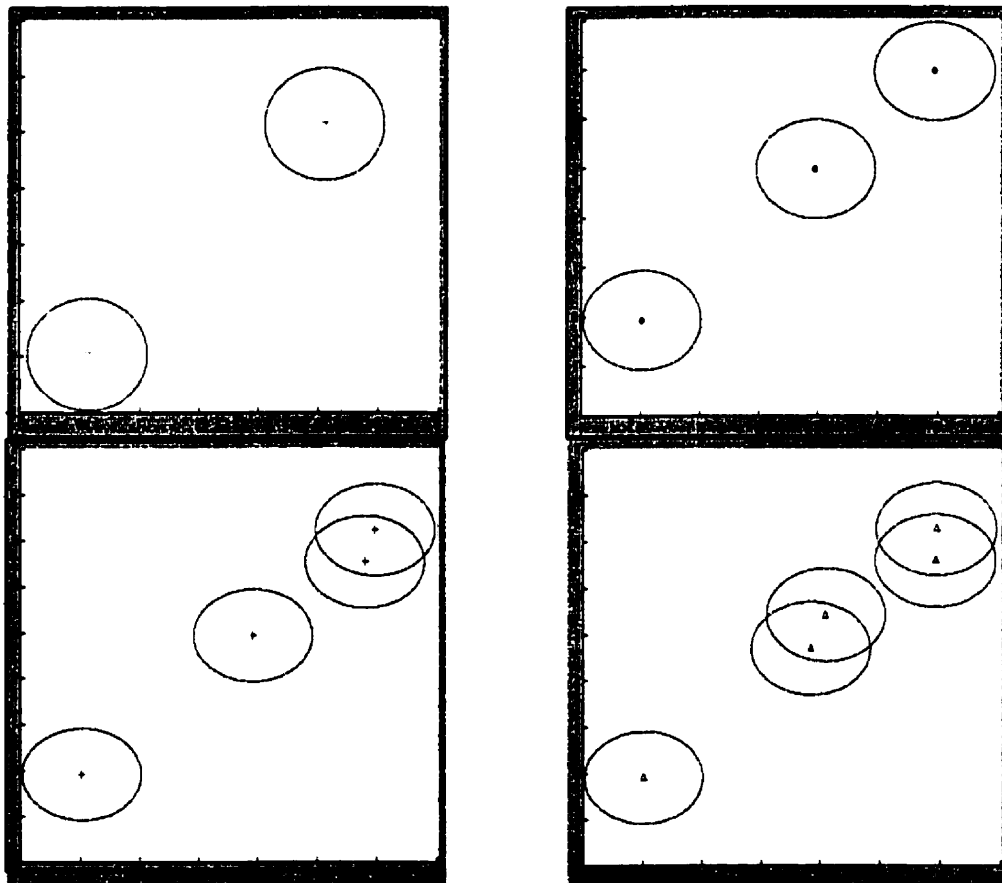


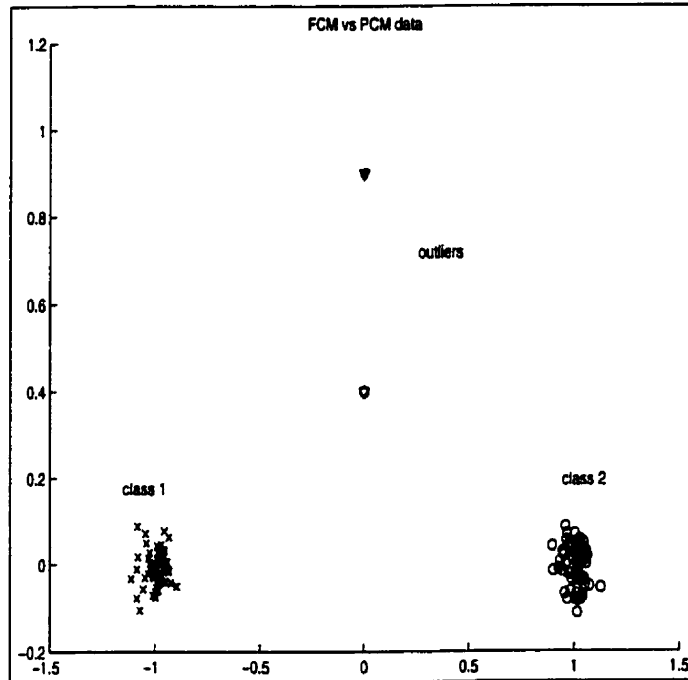
Figure 12. Cluster centers and region of influence for FCM clustering



The differences between pure data partitioning and mode seeking algorithms may be seen in Figure 13. A fuzzy data partition will assign equal memberships to the two inter cluster points since they are at the same distance from both centers. A possibilistic partition assigns a lower typicality value to the

intercluster sample furthest away.

Figure 13. Clustering: typicality vs membership



2.13 Artificial Neural Network (ANN)

An ANN is a massively parallel array of simple non-linear processing elements (PE) that effect supervised machine learning. Introductions and reviews may be found in [Bis94] [Bou96] [Rip94] [Cze94] [Koh88] [Lip97]. Note that the ANN conforms to the operational paradigm of biological neurons viz., information content is stored as weighted connections between processing elements. The ANN is initialized with a set of random weights and presented with each of the design vectors. Each PE operates on a weighted sum of inputs from the previous layer and passes its output to the next. The overall error, that is, the difference between the desired and actual network responses, is used to correct the weights. The vectors are presented repeatedly until the error stabilizes or until a maximum number of presentations is reached.

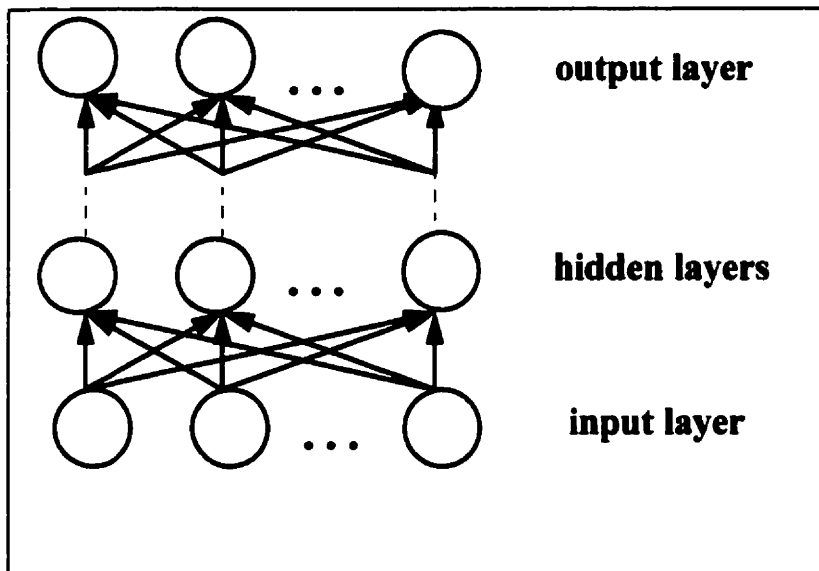
2.14 MultiLayer Perceptron (MLP)

The MLP is a supervised fully-connected feedforward ANN; the output from each

layer progresses forward through the network (Fig.14). The first layer is known as the input layer, the last as the output layer, while the remaining layers are the hidden layers.

MLPs have been applied successfully to virtually every area of pattern recognition: character recognition [Gos96], speech recognition [McC88], satellite imagery [Bou] and medical applications [Due96].

Figure 14. Multilayer perceptron architecture



ANNs are a non-linear classifier due to the non-linearity introduced by the PEs (Fig.15). This non-linearity entails a highly complex error surface and the profusion of local minima. A gradient descent technique is used to explore this space in search of a global minima for the error function. A common error function is the sum of squared errors,

$$\epsilon = \frac{1}{2} \sum_C \sum_N (d - o)^2, \quad (50)$$

where d is the desired response and o the network output.

For a differentiable transfer function backpropagation is used to adjust the inter-neuron weights to decrease the network error. The chain rule is used to calculate the partial derivative of the error with respect to each weight and adjusted as follows;

$$w_{t+1} = w_t + \Delta w_t = w_t - \alpha \frac{\delta \epsilon}{\delta w_t} + \beta \Delta w_{t-1} \quad \left(\begin{array}{l} \alpha \in [0, 1] \\ \beta \in [0, 1] \end{array} \right) \quad (51)$$

Here, α is known as the learning rate; β is a momentum term.

The output of each PE is

$$o = f \left(\sum_{i=1}^N x_i w_i - \theta \right) \quad (52)$$

Common transfer functions include a step function/ramp, or belong to the class of S-shaped sigmoid functions. N is the number of neurons in the previous layer and θ is a bias term. One such sigmoid function is the logistic function,

$$f(x) = \frac{1}{1 + e^{-\gamma x}} \quad (53)$$

Many variations to this general outline exist. A momentum term may be added to speed learning, a gain term, γ , in the logistic function can modify the transfer function slope, and an alternate distance metric, like the L_1 norm or Manhattan distance, may add robustness to the network. The use of bipolar transfer functions generally improves network performance since logistic functions have small outputs for sample values in the lower percentiles. This may bias results.

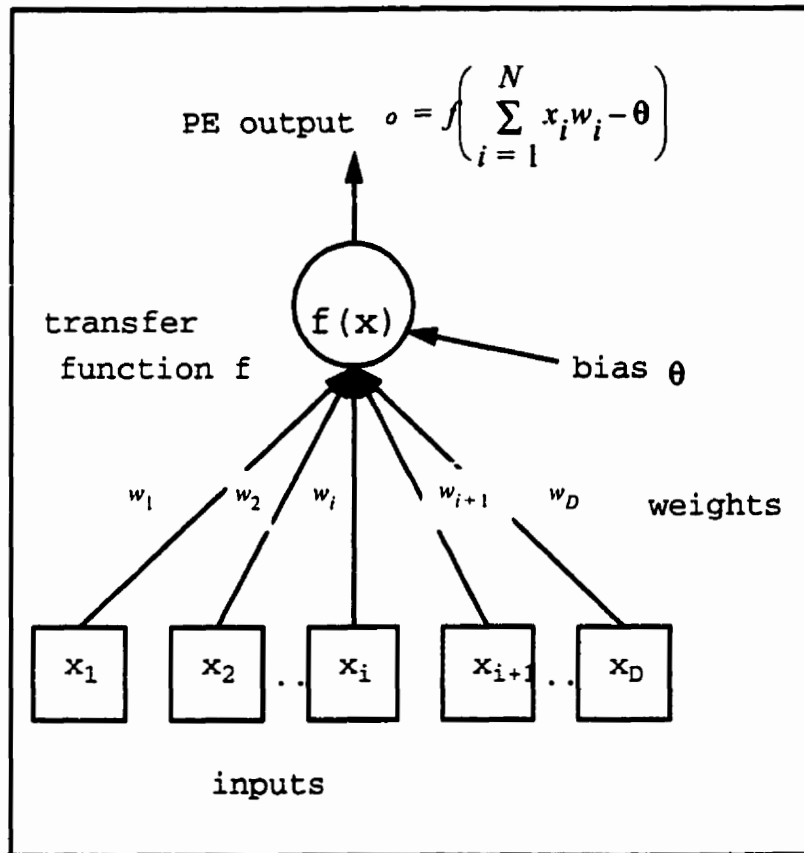
Overtraining should be avoided when presenting inputs to the network.

Essentially the network memorizes the training vectors exactly and has poor generalization to any other vector. This may be remedied by reducing the number of neurons and training for a longer period.

The relation between network architecture and convex regions in the solution space is well known. A one layer network effects a hyper plane solution, a two layer has disjoint convex regions while three may form arbitrarily complex regions limited only by the number of nodes. Thus a three layer neural network

is a universal function approximator. We follow the expedient principle of designing the parameters for the MLP on a single dataset and applying the set-optimal solution applied without further analysis.

Figure 15. ANN Processing element



2.15 Rejection of Classification

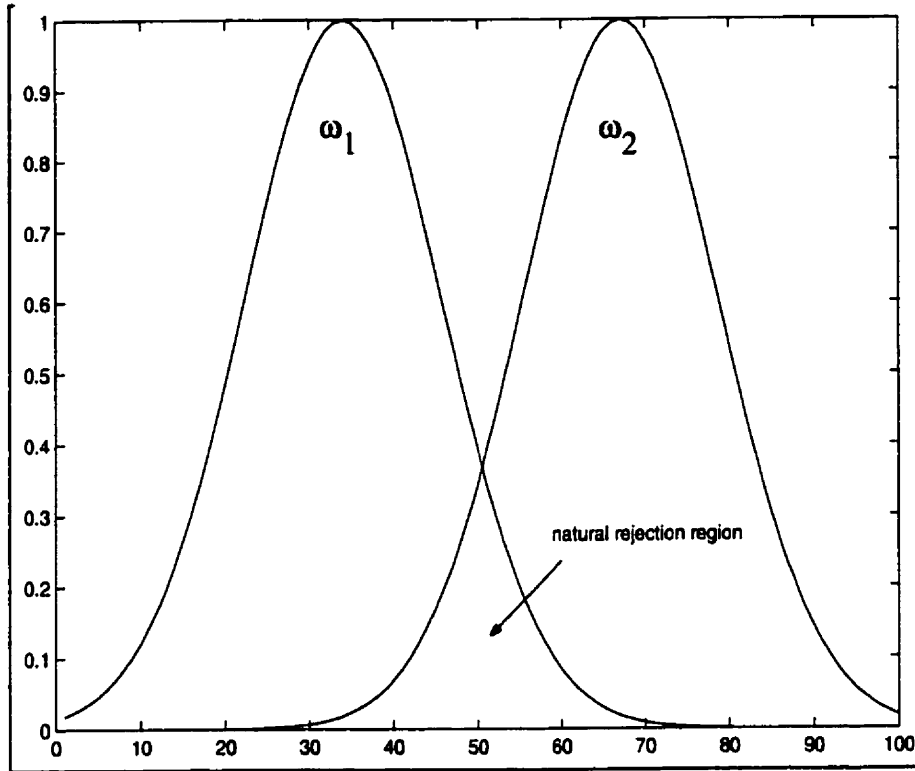
A study of rejection regions may be found in [Cho70][Ha97] while the somewhat related topic of incomplete pattern vectors is discussed in [Kit78][Paw93][Zhu90]. Depending upon the difference in magnitude of two or more discriminant functions a reject class may be implemented whenever the difference falls below some threshold [Chow]. This may be extended to a class selective rejection scheme [Ha97] where the classification is excluded from a subset of the total number of classes. Alternate rejection criteria exist after ranking the classes by a posteriori probabilities [Cho70] such as: top-R ranking where the R most probable classes are retained, $1 \leq R \leq K$, or the

constant-risk (set of alternatives) approach where the probabilities of the R best classes retained accumulate to some threshold. Reject classes minimize the error rate given a rejection rate, or vice versa. Class selective rejection attempts to minimize the error rate for a given average number of retained classes.

2.15.1 Defining classifier dependent thresholds

The level of uncertainty in any decision of a classifier may be measured by the relative magnitudes in its discriminating functions. In a Bayes classifier, we would be dubious of any label assigned when the difference between two or more discriminant functions is small and the pdfs are empirically based (Fig.16). This idea is implicit in the membership functions of FCM but may be extended by omitting from consideration samples with low membership values; or accepting the labels of test samples closest to clusters with low entropy where the entropy is based on membership values. For ANNs, one may consider the use of one output neuron per class and then threshold the difference between the output of the largest two largest output neurons. One advantage of this rejection criteria is its flexibility and applicability to any classifier.

Figure 16. Natural rejection region



We define as cluster-entropy the entropy of each cluster center based on its members (for hard C-means) and a subset of its members (FCM). Thus, our empirical estimates of probability density are the relative proportions of design vectors in each class associated with a particular cluster. Overall, this has the flavour of a Parzen window approach - probabilities for the conditional association of vectors in clusters may also be derived. Based on statistics of the cluster-entropies, we may define the reject rule: if x belongs to a cluster whose entropy is higher than a threshold T , we will reject making a classification decision. Thus, we avoid making high risk decisions and consider the true probability distribution (in a pointwise approach (it is discrete in the number of clusters which are the number of samples of the pdf)). By these means we hope to achieve a higher rate of correct classification on a smaller subset. Use of the mean and median cluster entropies as rejection thresholds is considered in [Ale99] while we consider labelling a specified percentage of test samples.

Consider two hard C-means clusters that contain the following design vectors. Let the tuple $[x_1, x_2, \dots, x_n]$ denote the number of vectors associated with a specific cluster with x_i from class ω_i , $1 \leq i \leq n$.

Table 10. Entropy of clusters

	cluster c_1	cluster c_2
members (ω_1, ω_2)	[4,7]	[3,10]
cluster entropy	0.9457	0.7793

Suppose also that the rejection threshold is 0.8. Then we would refuse to assign the label ω_1 to any test vector that is closest to (has maximal membership in) cluster c_1 , since it exceeds the entropy threshold, and classify as ω_2 any vector that is closest to cluster c_2 . We may also consider the entropy of various subsets of samples assigned to a cluster. The cardinality of samples per class exceeding a specific membership (FCM) may be used as well as weighting this cardinality by class a priori probabilities. Indeed, the a priori probabilities may define for each class a unique membership threshold for cardinality calculations. Consider the following FCM cluster with membership values for classes ω_1 and ω_2 respectively. Let μ_{min} be the minimum membership value considered for entropy

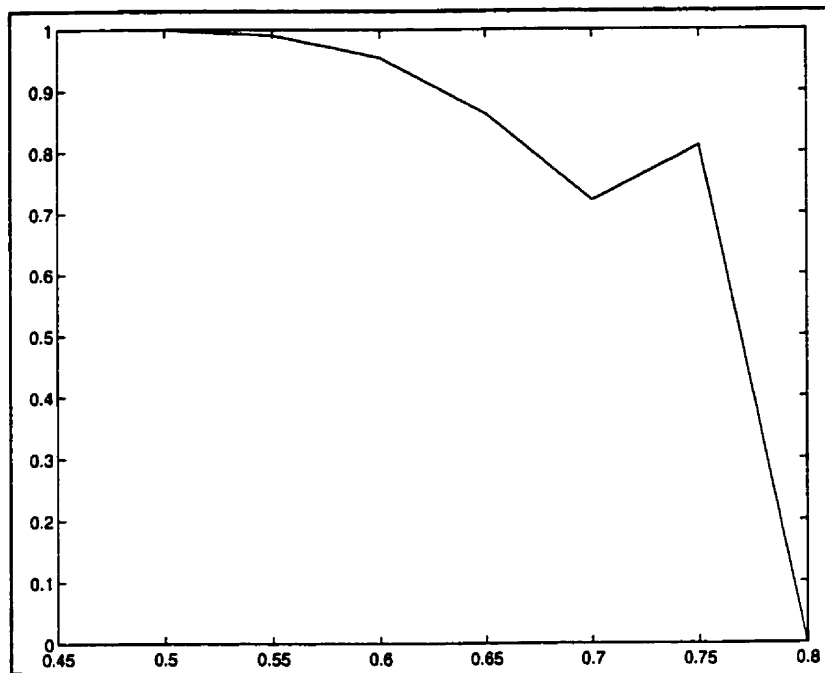
Table 11. FCM cluster rejection

	membership values
ω_1 samples	0.64 0.69 0.77 0.81 0.98
ω_2 samples	0.49 0.54 0.56 0.61 0.71

calculations.

Table 17 plots the cluster entropy versus μ_{min} .

Figure 17. Cluster entropy versus μ_{min}



Chapter 3 Preprocessing Techniques

Preprocessing in the context of pattern recognition is data manipulation that enables a less complex classifier to be used. Functions or transformations of the original data are often sought that maximize some measure of separability e.g. interclass variance while minimizing intra-class variance or maximizing the variance in a subspace. Alternatively, the data may be decomposed into various bases. Standard preprocessing techniques include the transforms of Hough [Ben97], Gabor [Por88], Fourier [Krz88] and normalization [Les83]. Feature extraction and selection [Kit86] [Far84] may also be included at this stage where not only the original data are considered, but all the forms generated by the above transforms and functions. Preprocessing is done implicitly in some complex classifiers (eg. the output from each layer in an ANN) though explicit manipulation may lead to new insight regarding the nature of the phenomenon (this could also be accomplished by analysing the neural weights of each layer).

Figure 18. Concentric distributions

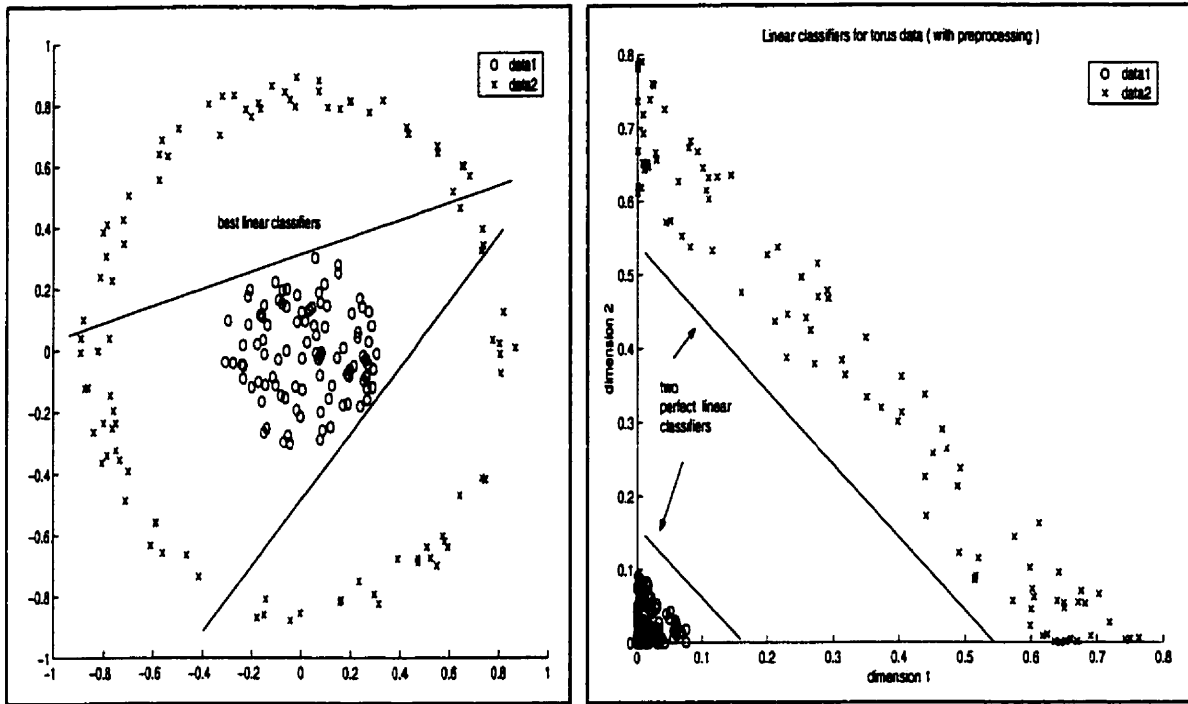


Figure 18 displays the exchange of data manipulation for a less complex classifier. The original data forms a convex region (class1) and a torus(class2). The problem is not linearly solvable. However, by taking the square of each original variable, the problem becomes solvable by a classifier of reduced complexity.

Alternate Measures of Separability that may be maximized (minimized) are: the scatter matrix,

$$S_i = \left[\sum_{x_k \in C_i} (x_k - v_i)(x_k - v_i)^T \right], \quad (54)$$

the within cluster scatter matrix,

$$S_W = \sum_{i=1}^C S_i = \sum_{i=1}^C \left[\sum_{x_k \in C_i} (x_k - v_i)(x_k - v_i)^T \right], \quad (55)$$

the between cluster scatter matrix,

$$S_B = \sum_{i=1}^C (v_i - \bar{v})(v_i - \bar{v})^T, \quad (56)$$

and total scatter matrix,

$$S_T = S_W + S_B = \sum_{k=1}^N (x_k - \bar{v})(x_k - \bar{v})^T. \quad (57)$$

3.1 Class-wise Preprocessing

Preprocessing may be done on the total data set (before designating design and test set), class wise on the design set, or on distinct clusters in each of the classes. For example, one could calculate separate cluster means, standard deviations, independent and principal components. This leads to the design of L local classifiers; the domain is partitioned into L regions and a unique classifier designed for each region. This may exploit different distributions densities and allow low complexity classifiers to be used for most of the sample space.

3.2 Reducing Dimensionality

Dimensionality reduction has often been motivated by psychological and visual limitations. The human mind is most adept when dealing with 5 to 10 features, while our vision is practically restricted to two dimensions (in the sense of viewing exhaustive dimensional information). There is also a rule of thumb for inducing meaningful relationships: have at least 5 samples/dimension [Paw99]. Some attempts have been made to represent high dimensional objects in two space, namely Andrew's plots [Krz88] and Chernoff faces [Che73]. Therefore an intuitive understanding is sought with a relatively small number of features. Leaving graphical approaches, we focus here on the uncorrelation and the independence of variance. This processing results in new features which are a weighted combination of original features, e.g. principal and independent components.

3.3 Principal Component Analysis (PCA)

Principal component analysis [Krz88] calculates the distribution of the sample variance among the D features. The D principal components are projections of the original features onto orthogonal axes. The components are ordered by variance such that the first principal component has the largest possible variance. Further components maximize variance subject to the condition that they remain uncorrelated to all previous projections. Mathematically, principal components are the eigenvectors of the covariance matrix; their ordering corresponds to the magnitude of their respective eigenvalues. For X , an N by D data matrix, with D by D sample covariance matrix S , the principal components

$y = [y_1, y_2 \dots y_D]$ are linear transformations of the original features, $x = [x_1, x_2 \dots x_D]$.

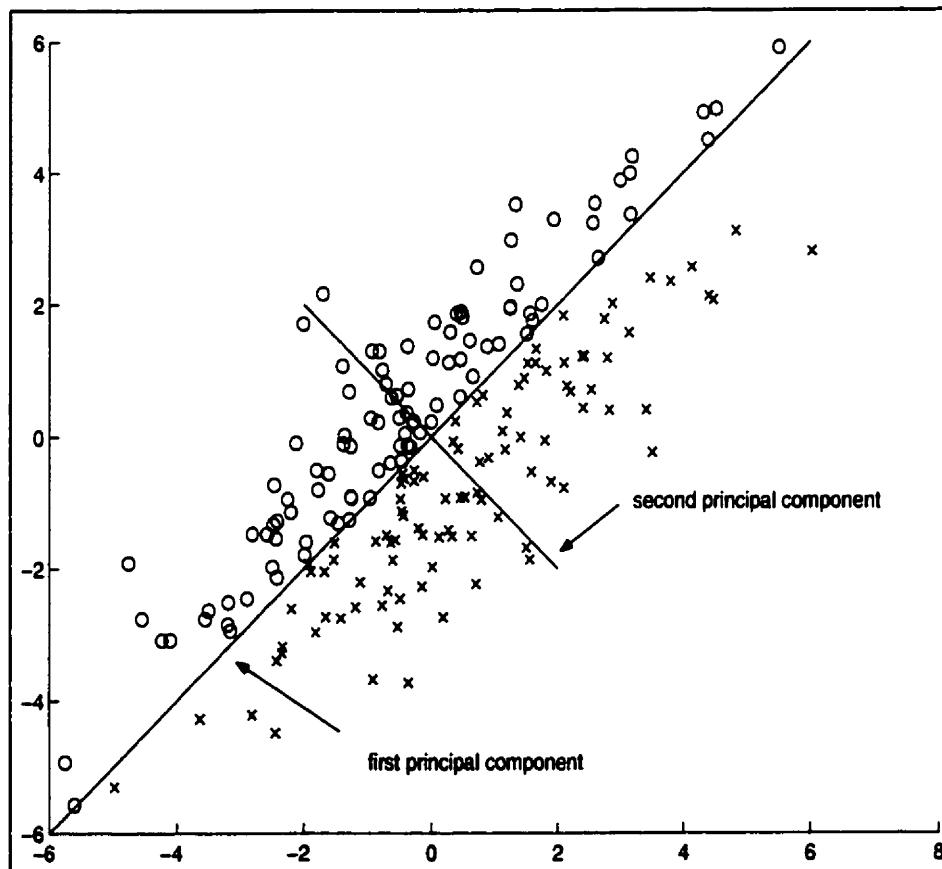
I.e.

$$y_i = \sum_{j=1}^D a_{ij} x_{ij}, \quad (58)$$

where $a_i = [a_{i1}, a_{i2}, \dots a_{iD}]$ is the eigenvector corresponding to the i^{th} largest eigenvalue of S . Analysis of the magnitudes of the coefficients in the eigenvectors of principal components often isolates meaningful relationships between physical features. Note that all the original features should have comparable variance. Therefore, we presuppose some scaling, standardization or normalization. A scree plot may be used to determine the number of principal components to consider; it plots the number of components versus the total percentage of accumulated variance. A point of inflection denotes the point of diminishing returns (optimum number of components) vis-a-vis total variance. However, the discriminatory power of the features may not be

related to its variance.

Figure 19. Variance not directly related to discrimination



3.4 Independent Component Analysis (ICA)

A stronger constraint on the feature variance is statistical independence: the joint density must be the product of the marginal densities. ICA (robust or non-linear PCA) determines a basis of dimension $d < D$

[Bel95][Bel][Van98][Lee99a][Hur96]. ICA is used in blind source separation and blind deconvolution [Bel95], and feature extraction [Bel97][Hur96][Kar97]. Often neural networks are used to compute independent components [Hyv96][Hyv97][Kar97][Kar1]. For our purposes, we consider the following problem: we observe samples from a linear combination of independent signals and want to estimate the independent signals. Let $s = [s_1, s_2 \dots s_q]$ be q statistically independent sources. The observed samples are denoted $x = As$ where A is a mixing matrix. The columns A are known as the independent component basis vectors (for feature extraction the columns are also features). Using higher order assumptions, ICA estimates the independent sources s given a required number of independent components. Usually $q \leq D$. Since uncorrelatedness is a prerequisite for statistical independence, PCA is often used to sphere (uncorrelate/whiten) the observed samples x ; it is also used to estimate the number of independent components. For a low noise level, the energy of x is concentrated in the first few principal components. For independent components, a linear (sphering) transformation V is determined such that for $y = Vx$, $E\{yy'\} = I$.

This is accomplished by setting $V = Q^{-\frac{1}{2}}$ where $E\{xx\} = Q$ is the correlation matrix for the original data x .

Then,

$$E\{yy'\} = E\{VxxV\} = Q^{-\frac{1}{2}}QQQ^{-\frac{1}{2}} = I. \quad (59)$$

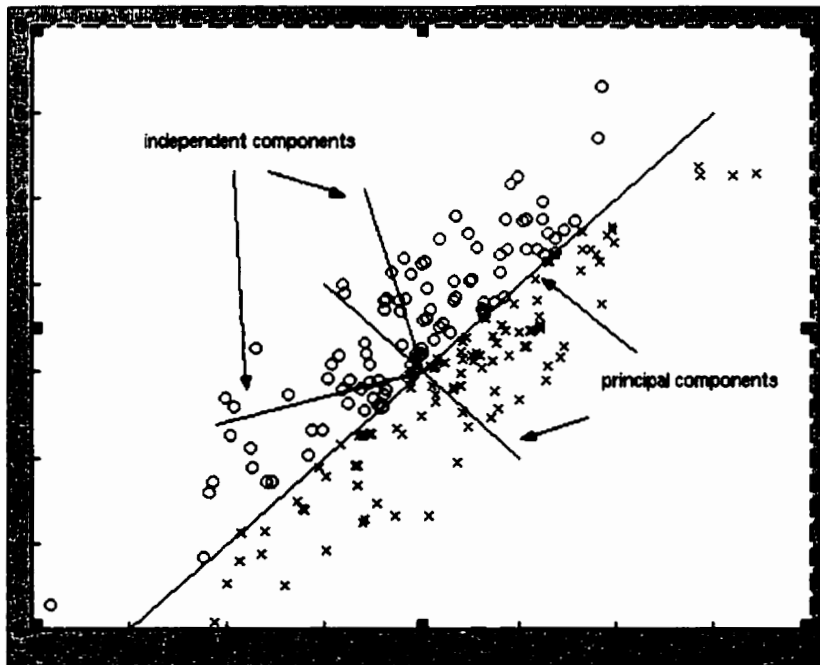
We now need only determine the orthogonal matrix V instead of the general matrix A . An orthogonal transformation (a joint density rotation) is now sought to maximize the non-normality of the marginal densities. Generally, a kurtosis maxima or minima is used to used as a criteria of independence. The

kurtosis (fourth order cumulant) for a zero-mean random feature x is

$$kurt(x) = E(x^4) - 3(E(x^2))^2 . \quad (60)$$

Note: the kurtosis of gaussian distributions is zero, while sub-gaussian (flatter) distributions, e.g. bimodal, have negative kurtosis while super-gaussian (peaked at zero with heavy tail) distributions have positive kurtosis. Recall that non-gaussian distributions contain significant information in higher order statistics. For sub(super)-gaussian distributions the kurtosis is maximized (minimized) and the solution may be found by standard gradient descent techniques. (The non-normality of the distribution is always maximized.) Note that only one independent gaussian component may be estimated; by the central limit theorem a linear mixture of independent random features is more Gaussian than the original features. Compared to the components determined by PCA, independent components are statistically independent and not necessarily orthogonal (Fig.20).

Figure 20. Independent and principal components for a distribution



3.5 Improper Labelling

There are many ways that errors may contaminate data. To increase the accuracy of performance in the face of inaccurate labels, we may, of course, revert back to an unsupervised recognition method. However, this may be extreme for only a partial lack of knowledge. We assume that most of the labels are correct. This leads to a robust reclassification of the samples using the median average deviation, fuzzy labelling (modelling the uncertainty of the labels), and probabilistic learning. Literature references for fuzzy logic include [Bez87] [Kau75] [Ped] [Zad] while robust reclassification is discussed in [Lau79] [Piz97] [Rou87] [Zhu92a].

3.5.1 Robust Reclassification (RR)

RR deals with outliers, noisy data and the effects they have on the estimation of classification parameters, particularly standard deviation and range. It exploits the robustness (and simplicity) of the median to decide whether samples are likely to belong to a given distribution. This confirmation of the expert classification of the training data is known as burnishing a tarnished gold standard. Rather than cull the deviant sample from the training data set, we reassign it. The median of the absolute deviation (MAD),

$$\tau(x) = \frac{\text{median}|x - \text{median}(x)|}{0.6745}, \quad (61)$$

is a robust estimator of the standard deviation. (Note: the constant 0.6745 is a scaling factor chosen such that $\tau(x)$ approaches the standard deviation as the error distribution becomes more gaussian.) Extending this concept for D-dimensional samples, $D > 1$, $x_j = [x_{j1}, x_{j2}, \dots, x_{jd}]$, $j = 1 \dots N_k$ of class ω_k , we define each component of τ_k as

$$\tau_{ki} = \frac{\text{median}|x_{ji} - \text{median}(x_{ji})|}{0.6745}. \quad (62)$$

A feature of x_j is declared a class ω_k feature i outlier if

$$|x_{ji} - \text{median}_{ki}| > Z\tau_{ki} . \quad (63)$$

Here, Z is a constant, $Z \geq 1$, and median_{ki} is feature i in the class ω_k

medoid. The membership of sample x_j in the class ω_k medoid is

$$D_{\omega_k}^{(j)} = \sum_i d_{ji}^{(k)} , \quad (64)$$

where

$$d_{ji}^{(k)}(x_j) = \begin{cases} 0, & \text{if } (|x_{ji} - \text{median}_{ki}| > Z\tau_{ki}) \\ \left(1 + |x_{ji} - \text{median}_{ki}|\right)^{-1}, & \text{otherwise} \end{cases} . \quad (65)$$

Class re-assignment of training vector x_j from class ω_i to class ω_j will

occur whenever

$$D_{\omega_i}^{(j)} < D_{\omega_j}^{(j)} . \quad (66)$$

Alternate formulations exist for the reclassification criterion. For multivariate samples, we may form a reject class for any sample that has T dimensions exceeding the product $Z\tau$. This may be done both class-wise or overall. Note: 99% of normal samples lie within 2.5 MAD of their median.

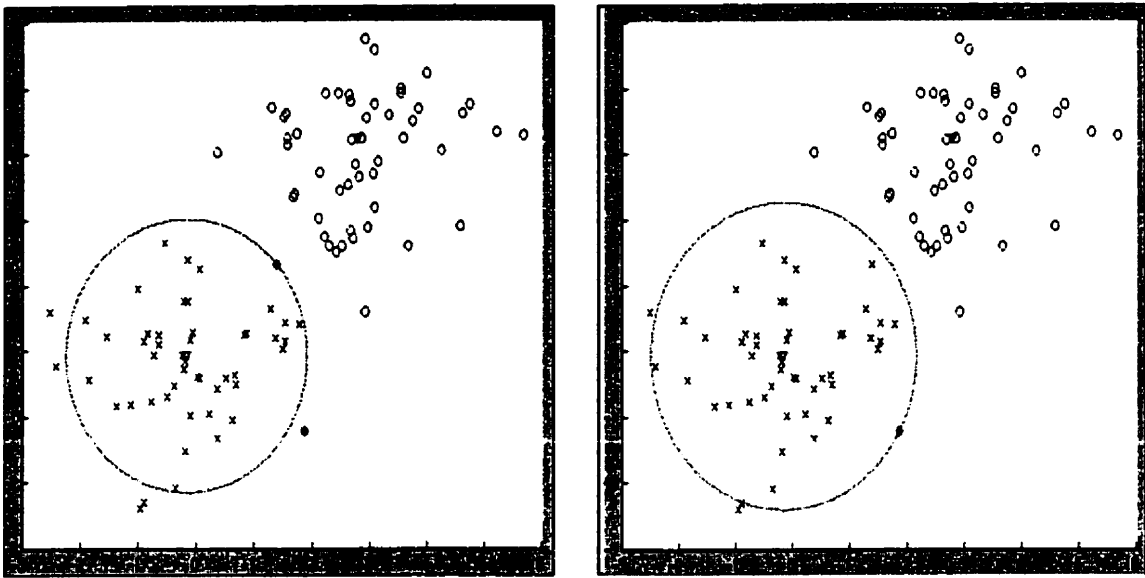
Figure 21 shows the reclassification of samples given different thresholds Z ($=1.2$ and 2). The circle denotes the closest radius where reclassification will occur. In this example, only samples from class ω_1 are reclassified. Note

also that samples from ω_1 outside the circle perimeter and on the far side

from the class ω_2 will not be reclassified. As the Z increases, we expect the

number of gaussian samples reclassified to decrease.

Figure 21. Robust Reclassification $Z = 1.2$ and $Z=2$



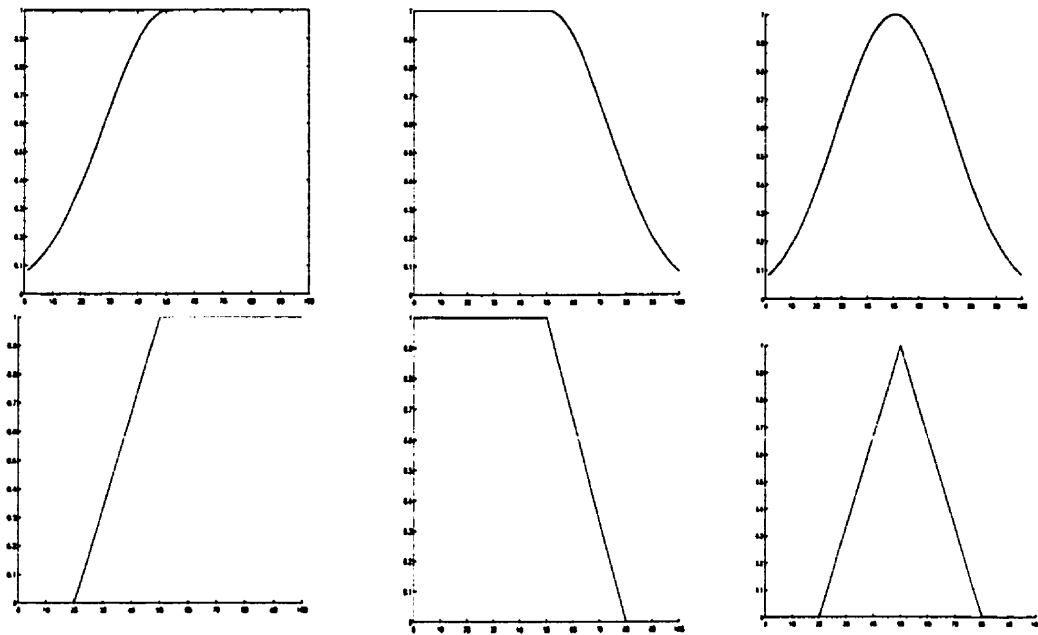
3.5.2 Fuzzy Labelling (FST)

FST [Zad65] incorporates uncertainty into mathematical models. The crisp (hard) degree of membership ($\{0,1\}$) in a set is extended by FST to include all membership values in $[0,1]$. In this way linguistic terms with implicit uncertainty e.g. 'around' and 'close to', are easily modelled. A fuzzy set is defined by its membership function, a non-negative value over a certain domain; typical mathematic expressions of this fuzziness involve gaussian or triangular fuzzy sets (Fig.22). Contrast intensification,

$$CON(\mu) = \begin{cases} 2\mu^2, & \mu < 0.5 \\ 1 - 2(1-\mu)^2, & \mu \geq 0.5 \end{cases} \quad (67)$$

is often used to accentuate the difference between membership values below and above 0.5. Here $\mu = \mu(x)$ is the membership value of x . Subsets of elements which all have a membership value of at least μ_{min} are also used to differentiate objects in a fuzzy set. Consider the fuzzy set $A = \{0.6, 0.65, 0.75, 0.9\}$. Denote the subset of A with minimal membership μ_{min} as $A_{\mu_{min}}$. Then $A_{0.6} = [A]$, $A_{0.7} = [0.75, 0.9]$ and $A_{0.8} = [0.9]$.

Figure 22. Membership functions



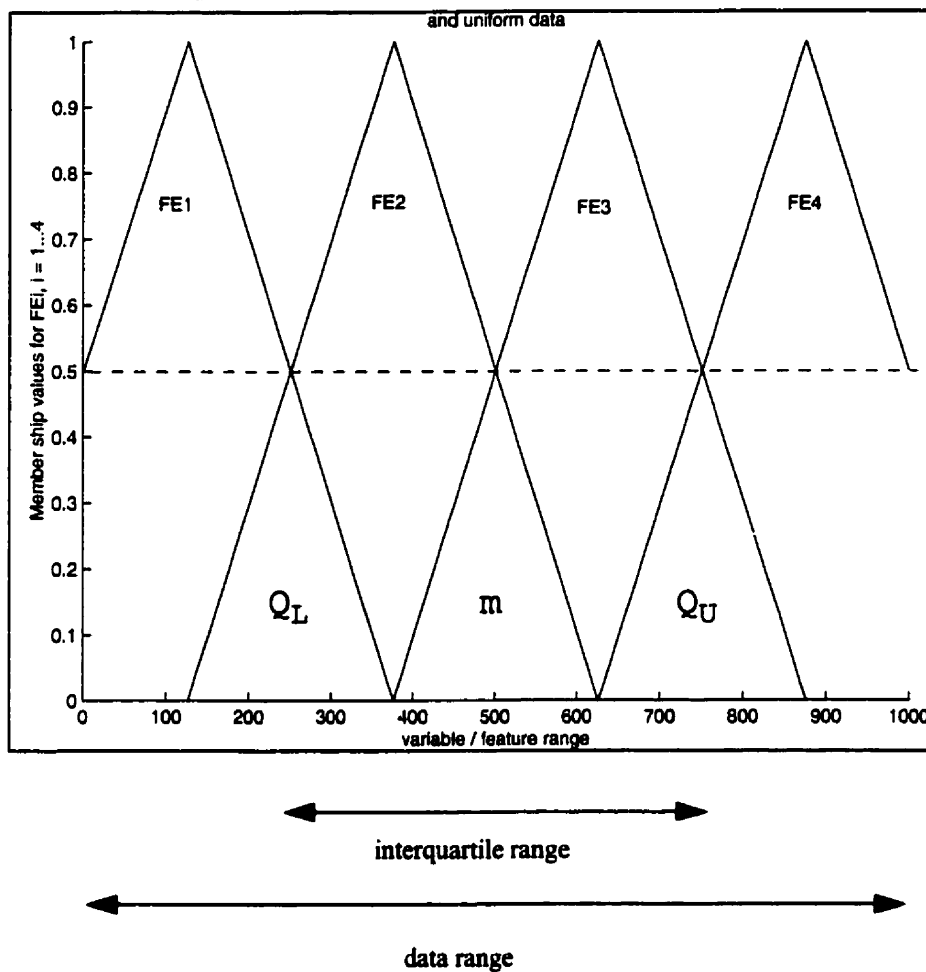
3.5.3 Interquartile Membership (IQ)

IQ partitions the data domain into fuzzy sets using overlapping membership functions to reflect the quartile distribution of a feature [Piz97]. The original data value is encoded depending upon its degree of membership in each of four fuzzy sets; the fuzzy encoding is utilized for each feature with feature specific parameters i.e. fuzzy sets (Fig.23). IQ is robust against outliers and transforms the data into a more tractable classification space (a space which conforms to certain outlier assumptions). The number of partitions is decided ad hoc in the form of a 1 of p intervalization. There are also various methods to determine what intervals to use. A general fuzzy membership function f_i may be considered as the trapezoid characterized by its width and end points (68). For $w=0$ we have the familiar triangular fuzzy set. The membership value at the intersection of two fuzzy sets is usually the same for all fuzzy sets in an encoding. Let x be the unencoded feature and a (b) its minimum (maximum) value. Denote as w the width of the fuzzy membership function and

\tilde{r}_i (\tilde{l}_i) the right (left) boundary of the fuzzy set. The equation for a fuzzy set is then

$$f_i(x) = \begin{cases} 1 \wedge \left(0 \vee \left(1 + w - 2 \left(\frac{1 + w - b}{r_i - l_i} \right) \left| x - \frac{r_i + l_i}{2} \right| \right) \right), & \text{if } (l_i \leq x < r_i) \\ 1, & \text{if } (l_i = x = r_i) \\ 0, & \text{if } (l_i = r_i \neq x) \end{cases} \quad (68)$$

Figure 23. Fuzzy encoding with 4 fuzzy sets



Note: when $b > 0.5$ there is a 1-1 mapping between the encoded and unencoded value (no extra degrees of freedom are introduced).

One method of determining appropriate values for l_i and r_i is to use percentile values of the sample distribution. Percentile values refer to the relative frequency distribution and the proportion of the number of features that lie

to the left (right) of some point. For example, the P^{th} percentile is characterized by having $P\%$ of the total number of samples to its left and $(100-P)\%$ of the samples on its right. For small data sets, this may be simplified by ranking the feature values by magnitude and selecting for $i = 1 \dots P$, the element with rank

$$\text{floor}\left(\frac{100i}{P}(n+1)\right) \quad (69)$$

as the i^{th} percentile value. Calculating boundaries of the fuzzy sets:

$$\tilde{l}_i = \frac{(2w-b+2)l_i - br_i}{2(w-b+1)}, \quad (70)$$

and

$$\tilde{r}_i = \frac{(2w-b+2)r_i - bl_i}{2(w-b+1)}. \quad (71)$$

3.6 Probabilistic Learning

Probabilistic learning refers to learning where the class labels have been assigned with some degree of error. This problem, known also as the probabilistic teacher or stochastic labelling, is discussed in [Agr70], [Gim74], [Gre80], [Kri90], [Sha72] and [Tit89]. This approach may be used with unlabelled data where relative frequencies are known or to compensate for tarnished gold standards (domain expert error). The first step in probabilistic applications should be to estimate the nature of this mislabelling and then estimate parameters for classification.

3.7 Probabilistic Teacher - No a priori knowledge

For probabilistically labelled data with no a priori knowledge of functional forms we refer to [Gre80]. We denote the data set of N vectors of D dimensions by X , the teacher label as θ , and the true label as ω . Let the probability that the teacher assigns label θ_k to a member of class ω_j as p_{kj} , the teacher class probabilities be pt_k and the teacher class densities be ft_k . The

conditional probability p_{kj} is assumed known. Recall that the class probabilities p_k and class densities f_k are unknown and will be estimated from the data.

3.8 Density Estimates

We assume the joint probability matrix $P = [p_{kj}]$ is non-singular and has as its inverse $Q = [q_{kj}]$. Then the teacher density for class i is

$$f_k^t(x) = \left(\sum_j f_j(x) p_{kj} p_j \right) / \left(\sum_j p_{kj} p_j \right) \quad (72)$$

with teacher probability

$$p_k^t = \sum_{j=1} p_{kj} p_j. \quad (73)$$

The correct density is

$$f_k(x) = \left(\sum_{j=1} f_j(x) (q_{kj} p_j^t) \right) / \left(\sum_j q_{kj} p_j^t \right), \quad (74)$$

with correct probability

$$p_k = \sum_j q_{kj} p_j^t. \quad (75)$$

Now the product of the correct density and probability yield

$$p_k f_k(x) = \sum_j q_{kj} p_j^t f_j(x). \quad (76)$$

An estimate of the teacher class probability, \hat{p}_k^t , is the ratio of vectors that the teacher ascribes to class i to total vectors. We then estimate the true class probability as

$$\hat{p}_k = \sum_j q_{kj} \hat{p}_j^t. \quad (77)$$

To estimate the teacher probability density, we partition the data into K subsets where each subset is declared by the teacher to come from a single class. We denote the estimate of the teacher density based on the k^{th} subset as \hat{f}_k^t and estimate the true density for class k as

$$\hat{f}_k(x) = \left(\sum_j q_{kj} \hat{p}_j t_j(x) \right) / \left(\sum_j q_{kj} \hat{p}_j \right). \quad (78)$$

Thus, the pattern recognition procedure is to decide class ω_k such that k maximizes

$$\sum_j q_{kj} \hat{p}_j t_j(x). \quad (79)$$

Chapter 4 Storm Cell Data

4.1 Radar Decision Support System

A prerequisite to short term weather prediction is the ability to classify severe weather cells at different stages in their life cycles. The Radar Decision Support System (RDSS), developed by InfoMagentics Technologies Incorporated, is used by Environment Canada to process wide beam radar returns across Canada. RDSS maintains a database of weather cells that meet a minimum reflectivity threshold (47dBZ) which is an indicator of storm severity [Kec94]. For an overview of fuzzy and neural network techniques applied to meteorology see [Piz95] [Lem80] [Lak97] [Imy92] [Eag83] [Dos94] [Den95] [Bla] [Ale99] [Fed].

4.2 Radar Processing

A radar data processing system conducts a volume scan by stepping a continuously rotating antenna through a series of elevation angles at regular intervals. Subsystems exist that allow operational meteorologists to focus their attention on regions of interest, known as storm cells, within the volumetric radar scan. Reflectivity averages per square kilometer are used to form the Adjusted Equivalent Reflectivity (AER) and Constant Altitude Plan Position Indicators (CAPPIs). The AER is used to prepare precipitation accumulations by smearing the radar return values by motion vector. This interpolates the cell sampling for integrated liquid calculations. A CAPPI is a horizontal slice of raw radar data. Other important measurements for storm cell characterization are the horizontal (vertical) reflectivity gradients, absolute value of spatial rate of change between neighbouring reflectivity values, the height of the absolute maximum reflectivity, and the largest reflectivity value found anywhere in vertical column. Horizontal (vertical) profiles are also used.

It is difficult to classify detected storm cells into specific types of storm events due to a number of confounding factors. Storm cells have an amorphous

and complex three-dimensional structure as well as a vague evolution. The nature of the rotating antenna causes the acquisition of the data to be incomplete. Also, the verification of the ground truth of the severe weather event is uncertain. The data itself may be suspect; high reflectivity values may be caused not by events but by the beam being bent until it hits the ground (which can give a high return) or through an error in the data collection.

The sampling rate of these cells is once every five minutes, with the reflectivity values spread out in time over the five minute interval as the radar adjusts its azimuth. This is sufficient to capture the dominant features in most heavy rains and winds; hail and tornadoes have substantially shorter life cycles and features may not be most clearly depicted (cf. Table 6).

Table 12: Temporal and spatial characteristics of severe events

Meteorological event	temporal range [km]	spatial range [km]
Heavy rain	5-60	10
Hail	5-10	5
Tornadoes	5-15	0.5 x 0.5
Gust Front	5-30	1 x 30
Shear Zone	10-30	1 x 10
Mesocyclone	15-60	4
Convergence/Divergence	10-60	5 x 20

When a storm cell is found, a number of parameters known as products are derived from the volumetric data. See Table 7 for a complete listing. Each cell is characterized by a 60 by 60 by 21 volume of reflectivity values (dBZ),

derived products, and the affirmation of heuristic criteria. (Note that the actual cell is sometimes a small portion of this volumetric cube and that a high percentage of the reflectivity values may be null due to radar blind zones. This occurs whenever the storm cell is near the radar and extends over the 19 degree azimuth.) Each cell is associated with other cells in the same vicinity and time. These groupings of similar cells are denoted as a matched group. Past work on the raw volume scans used histograms based on the median reflectivity value (dBZ) of a radar slice of the cell [Piz95]; features included for a 1995 Vivian of (18 hail/7phail/10 torn/24 ptorn) mean, median, variance, skewness, Pearson's second coefficient of skewness, quartile coeff of skewness (=measure of asymmetry within first and third quarters), kurtosis. Results indicated an 80% classification rate.

Table 13. RDSS Derived Features

Feature	Range
location	{ \mathcal{R} , \mathcal{R} , \mathcal{R} }
extent	{ \mathcal{N} , \mathcal{N} }
core size	{ \mathcal{N} , \mathcal{N} }
severity heuristics [SC ¹ , WG ¹ , HP ²]	{0-3} ¹ , {0-2} ²
core tilt angle, orientation	{ \mathcal{R} , \mathcal{R} }
flags (SCf, J, S, Vel)	{0, 1}
core tilt vector	{ \mathcal{R} , \mathcal{R} , \mathcal{R} }
velocity	{ \mathcal{R} , \mathcal{R} , \mathcal{R} }
core size	{ \mathcal{N} , \mathcal{N} }

Note that \mathfrak{R} and \mathfrak{N} denote the positive real and integer numbers respectively. Heuristic outputs are discrete numbers indicated by a range and $\{0,1\}$ denotes a binary feature.

4.3 Characteristics of Convective Cells

One of the dominant features of convective cells is a bounded weak echo region (BWER); this cavity in a field of high returns

indicates a strong updraft. An updraft may collapse to form a microburst (wind), grow to a supercell, or dissipate gradually.

Vertically Integrated Liquid (VIL) is also an important factor.

Vertically integrated reflectivity (VIZ) is calculated by integrating columns of returns and converted to a liquid density estimate (VIL). The familiar anvil shape of the cumulus nimbus storm is defined as overhang; a comparison of radar reflectivity values is taken at two given heights. Both must meet separate thresholds. Supercells are associated with a variety of severe weather and are distinguished by size and higher reflectivity from ordinary convective cells. Severe events are distinguished by relative cell motion orthogonal to dominant wind patterns (tornadoes), extreme reflectivity values (hail) and general magnitude.

Figure 24. Thunderstorm evolution

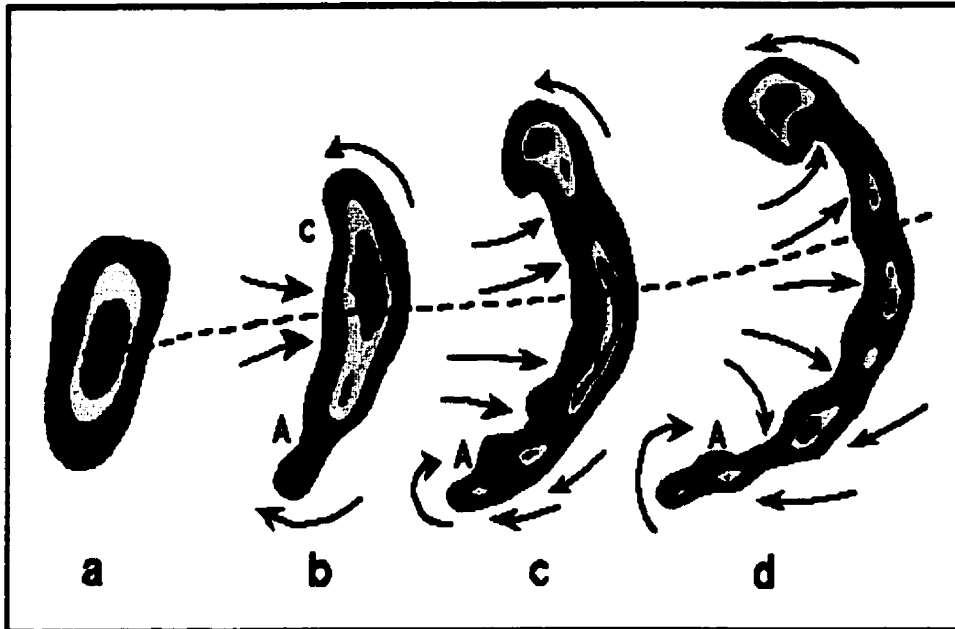


Figure 24 shows the typical evolution of a thunderstorm radar echo (a) into a bow echo (b,c) and into a comma echo (d). Dashed lines indicate the axis of greatest potential for downbursts. Arrows indicate wind flow relative to the storm. Note regions of cyclonic rotation (C) and anti-cyclonic rotation (A); both regions, especially C, are capable of supporting tornado development in some cases.

Figure 25. Microburst cross-section (Caracena 1982; 1987)

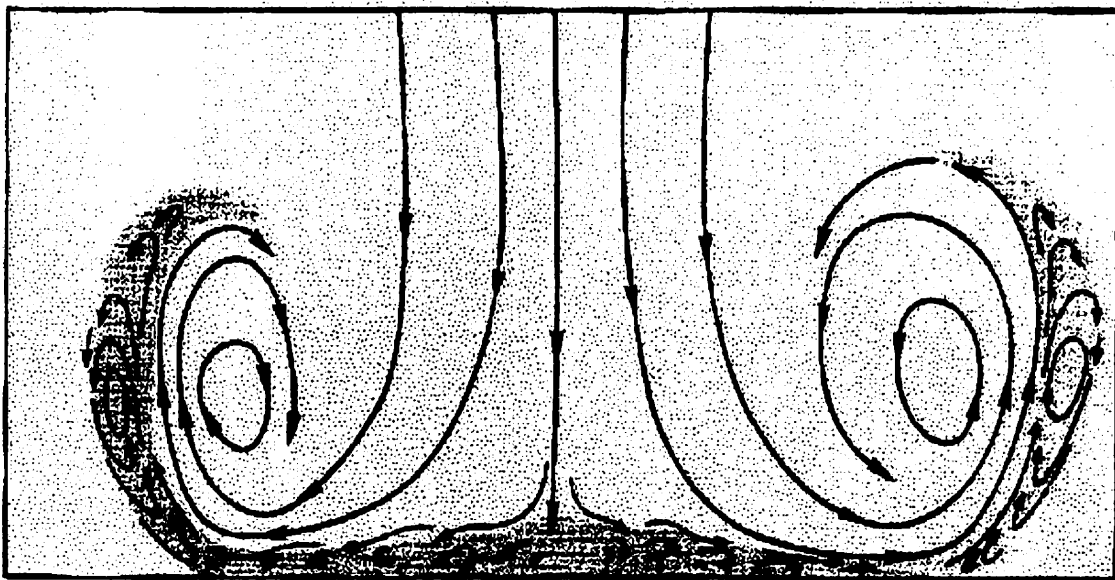


Figure 25 shows a cross section of the conceptual vortex ring model of a microburst. The hazard to small aircraft is clear. Wind gust potential (WGP) identifies locations where strong down drafts are or may occur through calculations involving VIL and echo top.

4.4 RDSS Data

The data used by RDSS includes both static and dynamic measurements, raw reflectivity values and inferred characteristics. This includes cell parameters and heuristic flags.

4.4.1 Flags

Flags are binary values that indicate whether an event has occurred or whether another field is valid. Velocity set Flag - indicates that rate of change calculations are valid since more than one cell belongs to the matched cell file. Supercell Flag - Cell meets the heuristic criteria defined for a supercell. See the supercell severity rating. Join (Split) Flag - have two distinct cells joined (split) from the last sampling?

4.4.2 Heuristics

Heuristic criteria are functions of cell size, volume, vertically integrated liquid (VIL); thresholds used are discussed in the literature [Wes] for both supercell and microbursts. Both supercell analysis and wind gust analysis calculate separate echo top profiles. An echo top shows the height of storm formations; a column of relectivity is scanned down until a threshold is exceeded or the bottom of column reached. The height of the first threshold exception is recorded.

Figure 26. Convective cell cross section

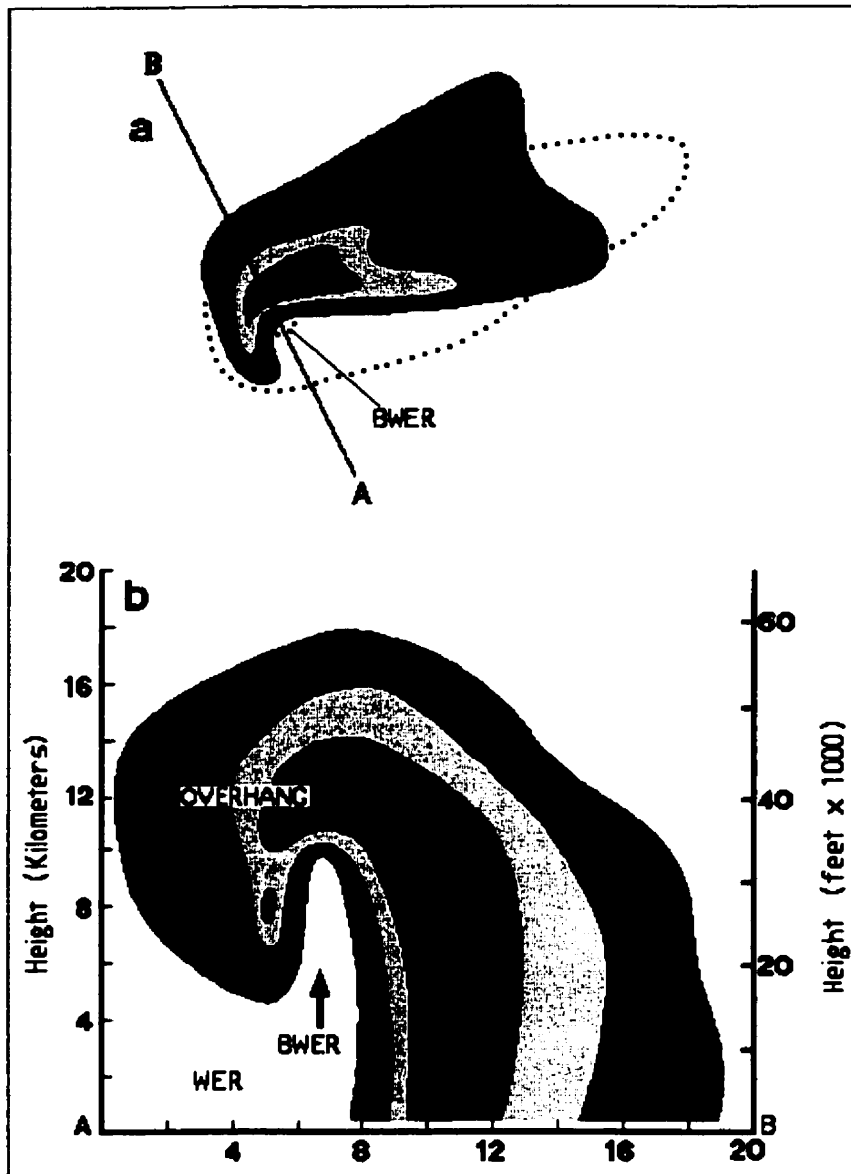


Figure 26 shows a cross section of a convective cell exhibiting a BWER and overhang.

4.5 RDSS Storm Cell Classification

From the static and dynamic measurements of the cell, RDSS compares each cell to heuristic criteria that define supercells, microburst conditions, and hail probability. Based upon which event severity level the cell meets the cell is stored with any associated series of cells in an overall severity directory. The overall severity is composed of a logical combination of individual event severities and is used to advise the

public. Unfortunately, some weather manifestations thought severe are not given an RDSS severity rating because they do not meet the minimum reflectivity value for RDSS archiving. For post event analysis exact times are not known. Additionally, several cells or series of cells may be found in the same locale and time interval.

4.6 Schedule Correlation

Class labels are assigned by a labelling algorithm. This includes correlating a schedule of observed storm events (including geographic location, duration and time of origin, and meteorological characteristics) with storm cells found in the radar scan within a time and distance radii. As this schedule will contain events observed by the general public (apart from meteorologists and climate observers), the descriptions may not be completely accurate and hence the class labels may be imprecise. Also, in a series of cells, the actual event may not be distinguishable from non-severe radar images.

4.7 Alternate labelling strategies

Methods to deal with the unverifiability of ground truth include the following:

- i) Label all files in each matched cell with the event type (hail, tornado, wind, rain) of the nearest scheduled event.
- ii) Use only the cell chosen as the single event cell to train the classifier as a prototype (with or without random noise).
- iii) The cells can be labelled with a fuzzy label based on their distance in time and/or location to the cell chosen as the most likely event cell. The use multiple fuzzy labels may be implemented by forming a conditional probability matrix of events. The probability that events follow each other or are simultaneously manifested would be invaluable for a fuzzy labelling

scheme and would allow one to label a cell as, e.g. 0.7 tornado and 0.6 hail.

Figure 27. A high precipitation (HP) storm

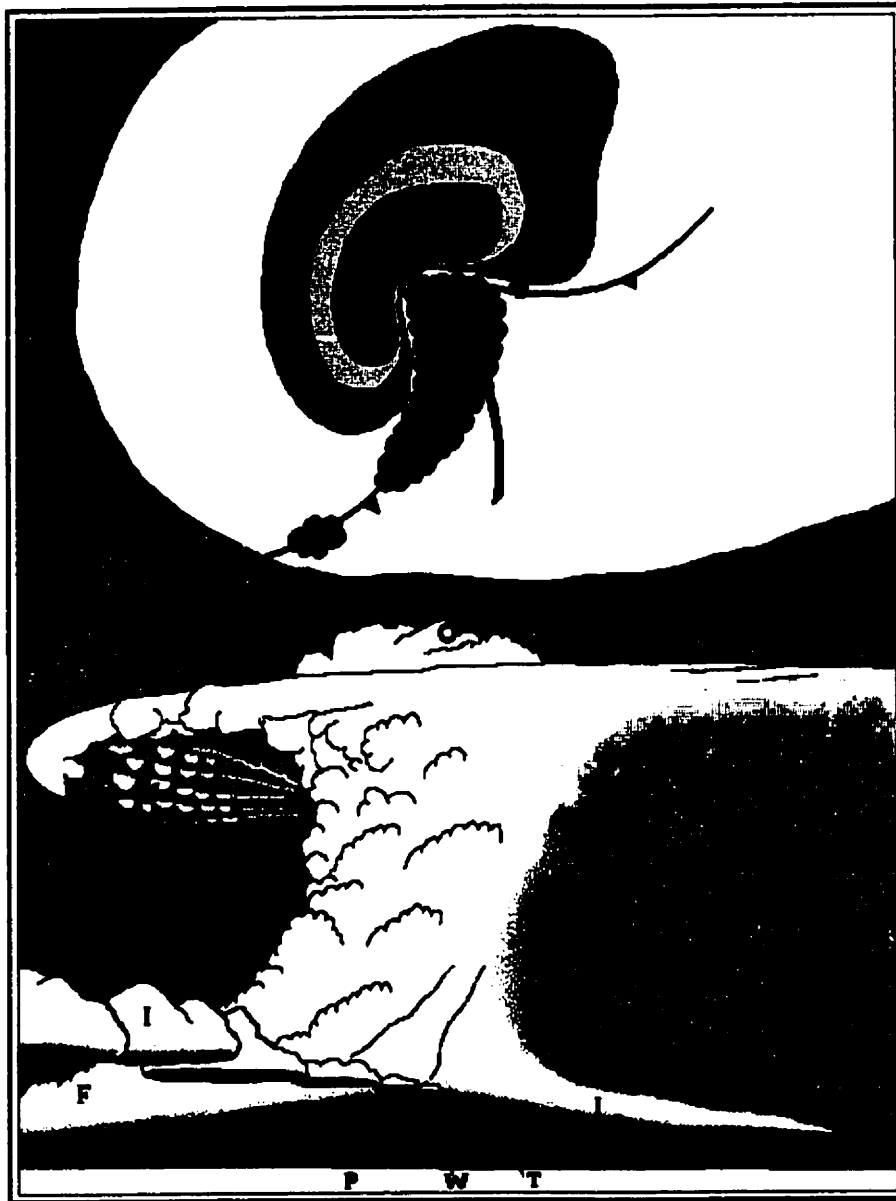


Fig. 3 HP storm (HP supercell). Schematic view from above of an HP storm (a), and visual features of the same storm as might be seen from the east or southeast (b). In (a), regions of radar reflectivity are colored according to intensity. Scalloped lines and violet shading indicate the region of main updraft (U). Surface inflow is indicated by arrow. Frontal symbols indicate location of gust front. Features in (b) include overshooting top (O), back-sheared anvil (B), wall cloud (W; often obscured by precipitation), tail cloud (T), inflow band (I), and regions of heavy precipitation (P). Compare with Figs. 5 and 7.

Figure 27 shows a high precipitation storm. A schematic view is seen in (a) with (b) showing the visual characteristics of such a storm viewed from the east to southeast. In (a) regions of radar reflectivity are coloured according to intensity. Scalloped lines and violet shading indicate the region of main updraft (U). Surface inflow is indicated by an arrow. Frontal symbols indicate location of gust front. Features in (b) include overshooting (overhang) top (O), back-sheared anvil (B), wall cloud (W) (often obscured by precipitation), tail cloud (T), inflow band (I), and regions of heavy precipitation (P).

For simplicity and due to lack of information on such a conditional probability matrix, we use only labelling schemes i) and iii). When comparing scheduled event locations and times to those of the RDSS matched cells, a time radius of 40 minutes was used and a location radius of fifty kilometres. The rather large location radius (RDSS locates the event to within seconds of a degree) are due to the notorious imprecision in the scheduled time of the events. An increase in both the considered location area and time interval is meant to include events both before and after their peak severity and to register valid events that have error in both scheduled fields.

4.8 Experimental Data Set

This thesis will concern itself solely with the data set generated by the Environment Canada wide beam radar in Vivian, Manitoba for the months May 1997 through September 1997. RDSS collected approximately 200 files with an average of 6 cells per file for the months July through September in 1997. It should be noted that the majority of these cells met the RDSS heuristic for its lowest severity rating, S0. Approximately 5% of the matched cells were rated S3. The corresponding schedule of events listed less than 100 confirmed events in that interval. Correlation between the schedule of events and the RDSS matched cells reduced this data set to the cells corresponding to 43 severe and non-severe weather events. These matched cells contain 185 associated cells that may or may not have been verified as one of the four event types (cf. Table 14).

Table 14. Number of Cells per event type

Tornado	25
Wind	27
Hail	99
Heavy Rain	34

Table 15 shows the position of the most likely event-cell from each matched cell sequence. When this position differs from the number of cells, the most likely position is indicated first, followed by the number of cells in its matched group in parentheses.

Table 15. Most likely cells from matched groups

Event Type	Most likely cell in matched cell sequence
Tornado	1 2 1 2(3) 3(6) 1(2) 4(5) 2(5)
Wind	1 2(5) 1 5(7) 2 9(11)
Hail	1(2) 1 1 1 2 6(7) 1 1 2 3(8) 1 2(6) 2 7(11) 3 1 1(7) 1 2 3(4) 9(11) 1 12(18) 3(5)
Heavy Rain	1(2) 1 1 1 2 6(7) 1 1 2 3(8) 1 2(6) 2 7(11) 3 1 1(7) 1 2 3(4) 9(11) 1 12(18) 3(5)

Chapter 5 Results

We begin by considering the data set composition according to event type. Next we consider the effect on the data set of robust reclassification. Classifier specific parameters are mentioned and results listed. The results from the experiments are organized by method (FDT, FDTg, PINV, MLP, KNN, FCM, MLP with fuzzy labels, FCM with rejection). Each section displays the kappa score for classification on the test and design sets using the preprocessing strategies listed in Table 17. Overall comparisons of classifiers and preprocessing follow.

5.1 Generation of training and test sets

The design set was composed of 137 samples chosen randomly from the data set. Because of the variance in the number of samples per class, approximately 75% of the samples from each class were included in the design set. The artifice of equal samples per class may be achieved by resampling, adding the multiple samples from the smaller classes with additive white gaussian noise, was considered but the noise characteristics are not justified.

Table 16. Design and test set membership

	Tornado	Wind	Hail	Heavy Rain
Design set	18	20	74	25
Test set	7	7	25	9

Each classifier was presented with 10 unique design sets and the overall classification rate recorded. Both the design and test confusion matrices are shown as are the traditional and kappa scores for design and test accuracy.

Table 17. Preprocessing strategies

P_1	no preprocessing
P_2	principal component
P_3	independent components
P_4	robust reclassification
P_5	interquartile membership

5.2 Robust Reclassification Matrices

For a given threshold Z , we list all samples that would be relabelled by robust reclassification where the parameters are calculated on the whole data set. For experimental purposes, τ was calculated with the design set only and a threshold of $Z=2.5$ used.

Table 18. Robust reclassification $Z = 1$

reclassified \ original	tornadoes	wind	hail	rain
tornadoes	0	8	4	2
wind	8	0	5	0
hail	28	22	0	22
rain	12	4	4	0

Table 19. Robust reclassification $Z = 2$

reclassified \ original	tornadoes	wind	hail	rain
tornadoes	0	7	3	1
wind	8	0	3	1
hail	30	24	0	28
rain	10	2	2	0

Table 20. Robust reclassification $Z = 2.5$

reclassified \ original	tornadoes	wind	hail	rain
tornadoes	0	8	4	1
wind	8	0	3	1
hail	33	23	0	26
rain	10	1	5	0

Use of rejection classes was done only with FCM and MLP.

Thresholds for rejecting classification are discussed in their respective sections. Fuzzy labels were used only with the multilayer perceptron.

Due to time constraints, probabilistic learning was not implemented although other probabilistic methods [Gre80] were tested and performed poorly.

5.3 Notes on preprocessing with different classifiers

For most classifiers (MLP, PINV, FCM, KNN), accumulated variance determines the number of components to keep; however, this criterion would restrict our decision tree to a depth of one leaf. This would compare the methods poorly as the decision tree is not iterative (extra dimensions do not slow it down to the same extent as FCM and MLP). One possibility is to use an oblique decision tree that would simultaneously consider several features. Thus, for FDT (with and without gain) we use 4 components for both PCA and ICA. Due to time constraints, some methods were evaluated less than 10 times. This is obvious from the confusion matrices but is not noted in the overall comparison. This is done because the results are typical (in the information-theoretic sense).

5.4 Specific classifier parameters

5.4.1 Fuzzy decision tree

Each feature was partitioned into 6 sets [Sos98]. An obvious improvement that

should be made is to compose partitions based upon the discrete nature of the feature; n-ary features may have at most n partitions.

5.4.2 MLP parameters

The following architecture was used for the MLP.

Table 21. MLP architecture

Architecture Layer	number of neurons
Input layer	22 (88)
Hidden layer 1	30
Hidden layer 2	20
Hidden layer 3	15
Output layer	4

This architecture was chosen by trial and error. The error of the network was observed for convergence to a zero error. No network achieved 100% on the design set; overtraining can not be the reason behind poor generalization. Although we could theoretically achieve the same results with two hidden layers, the speed of training would suffer. The network was presented with the design set in the same order for 500 epochs. At this time, if the confusion matrix associated with the design set exhibited diagonalization, training terminated and the test set was evaluated. This happened infrequently and the number of training sessions (each of 500 epochs) was limited to 20 sessions.

Table 22. MLP parameters

training function	gradient descent
performance function	mean square error regularized
distance function	Manhattan distance
transfer function	hyperbolic tangent sigmoid
training sessions	max 20 500 epoch sessions

Note: mean squared error regularized considers both the mean square error and the mean squared weights and biases of the net layers.

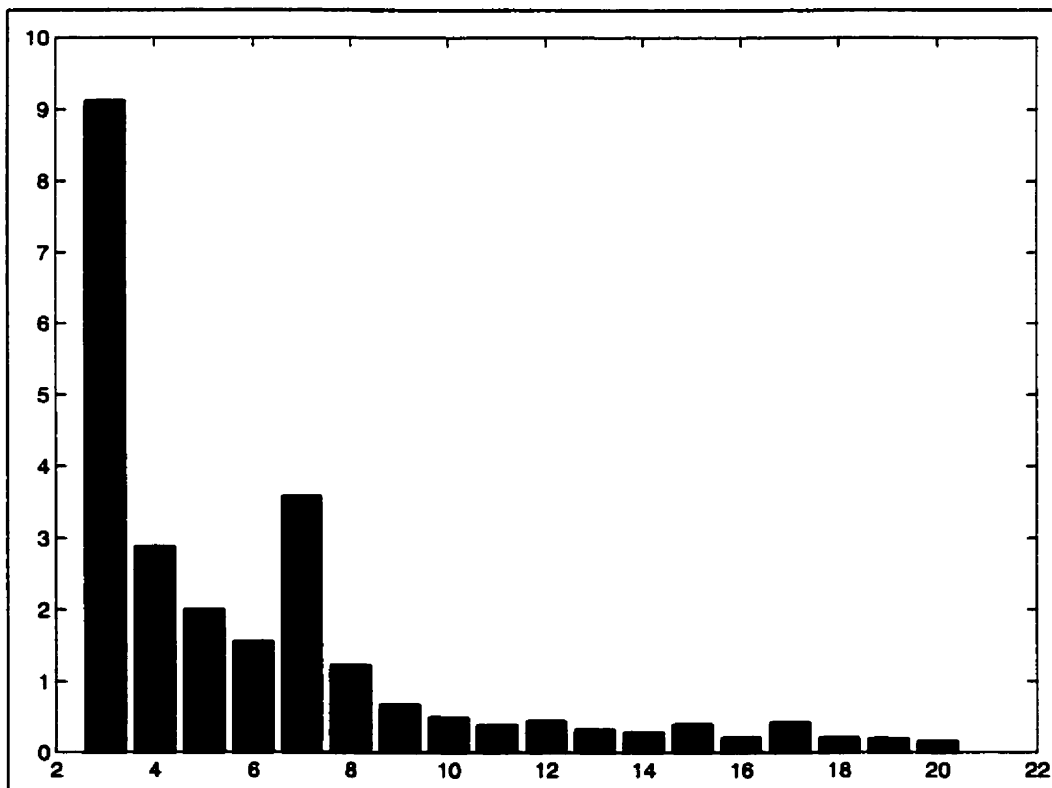
5.4.3 FCM parameters

The Xie-Beni validation index used to determine the number of clusters. For the initial design set, the data was clustered with FCM and the validity index calculated for 10 through 20 clusters. The maximum validity value determined the number of clusters used in all further clustering.

Table 23. FCM parameters

fuzzification factor	m = 2
maximum iterations to converge	40
error threshold to terminate	0.000001
initial centers	random training samples

Figure 28. Xie-Beni Validity index (fuzzification factor =2)



5.5 Fuzzy decision Tree - no gain

Figure 29. Results for testing and training sets using FDT

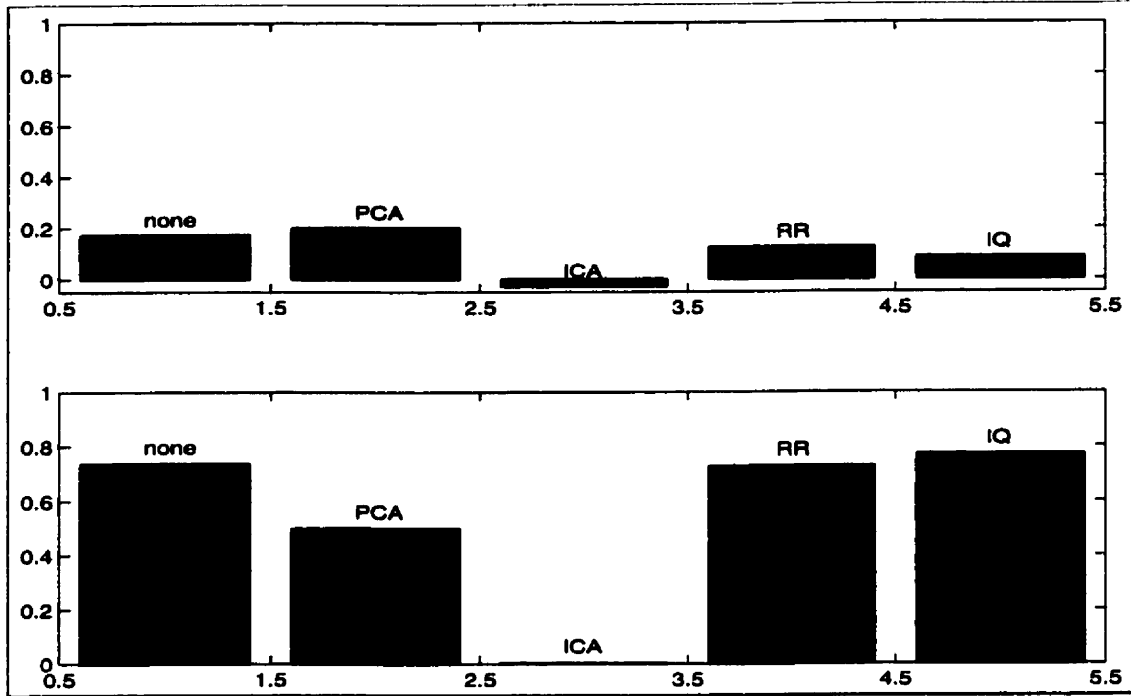


Table 24: Training and test confusion matrices FDT no pre P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	141	26	18	16		18	17	20	12
wind	10	150	49	5		20	21	44	5
hail	24	22	654	34		20	30	145	40
rain	5	2	19	195		12	2	41	33

Table 25: Training and test confusion matrices FDT P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	94	25	43	21		41	30	8	3
wind	11	111	49	9		29	15	6	1
hail	72	64	631	102		78	121	61	36
rain	3	0	17	118		8	18	10	15

Table 26: Training and test confusion matrices FDT P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	68	24	30	15		2	5	18	2
wind	5	74	32	7		25	20	66	21
hail	101	98	661	108		42	44	148	55
rain	6	4	17	120		1	1	18	12

Table 27: Training and test confusion matrices FDT P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	151	22	24	13		7	11	25	16
wind	12	135	47	2		13	19	28	3
hail	22	38	651	39		30	29	157	43
rain	2	2	14	196		13	14	44	28

Table 28: Training and test confusion matrices FDT P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	160	35	16	15		14	17	40	13
wind	6	148	52	14		12	4	39	11
hail	12	16	665	27		34	45	147	34
rain	2	1	7	194		10	4	24	32

5.6 Fuzzy Decision Tree with gains ratio

Figure 30. Results for testing and training sets using

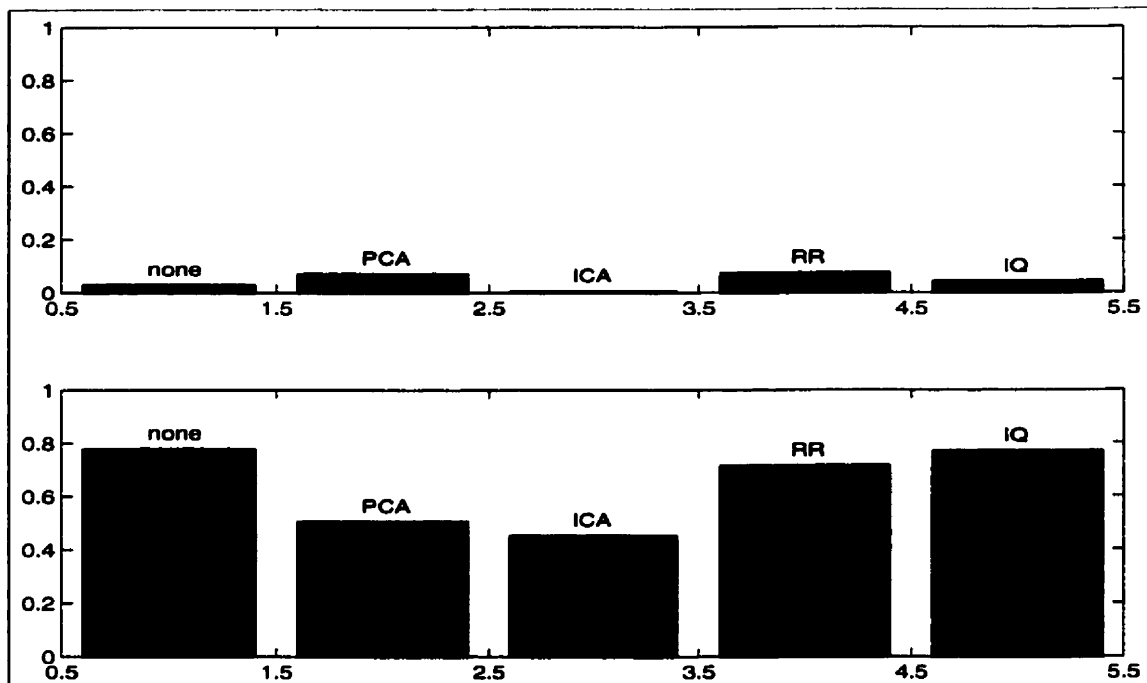


Table 29: Training and test confusion matrices FDT gain P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	116	21	16	8		10	12	21	9
wind	4	103	38	1		15	6	35	9
hail	5	15	461	24		18	29	87	29
rain	1	1	3	142		6	2	32	16

Table 30: Training and test confusion matrices FDT gain P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	97	26	40	21		40	29	8	4
wind	10	109	50	9		27	14	6	0
hail	70	64	632	98		81	124	62	37
rain	3	1	18	122		8	17	9	14

Table 31: Training and test confusion matrices FDT gain P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	69	25	31	15		2	5	19	5
wind	5	79	38	5		31	26	77	24
hail	100	93	650	95		36	38	133	52
rain	6	3	21	135		1	1	21	9

Table 32: Training and test confusion matrices FDT gain P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	160	33	26	17		7	21	36	21
wind	7	123	47	8		14	9	28	4
hail	19	38	652	39		33	38	152	38
rain	1	3	11	186		9	5	38	27

Table 33: Training and test confusion matrices FDT gain P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	164	32	20	11		13	18	32	13
wind	5	141	51	6		19	6	44	12
hail	9	24	663	33		27	42	128	37
rain	2	3	6	200		11	4	46	28

5.7 Pseudo inverse

Figure 31. Results for testing and training sets using PINV

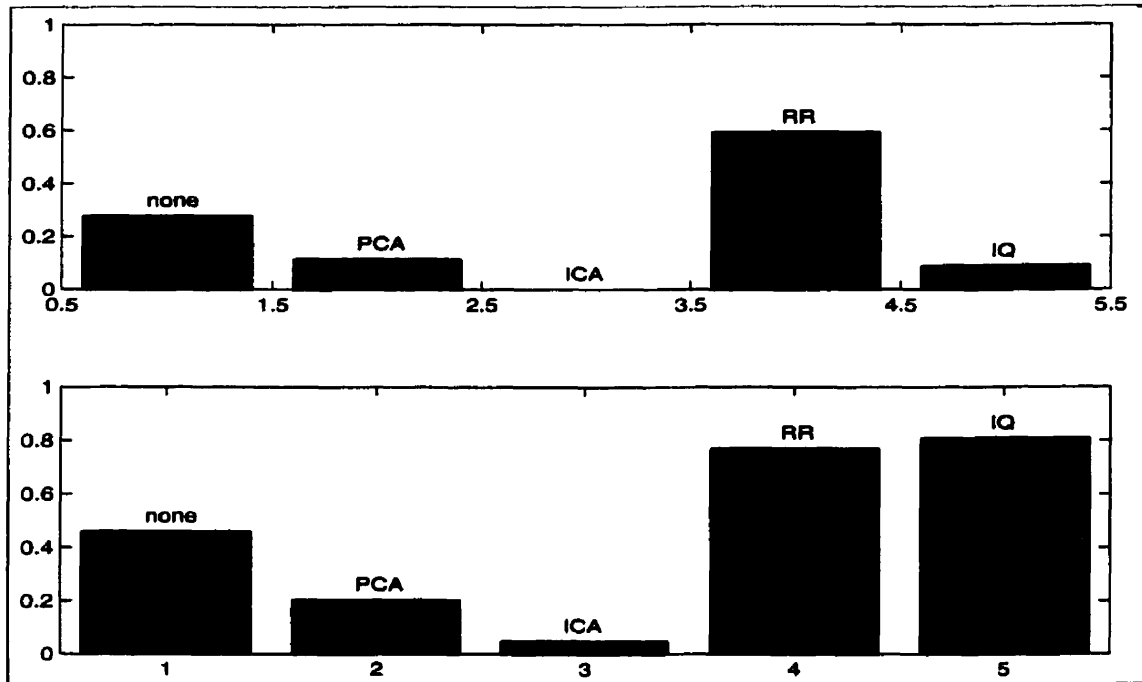


Table 34: Training and test confusion matrices PINV P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	62	11	74	33		15	3	38	14
wind	11	60	123	6		4	11	45	10
hail	4	19	672	45		5	18	208	19
rain	4	1	93	152		7	5	37	41

Table 35: Training and test confusion matrices PINV P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	93	57	20	10		30	22	10	8
wind	41	120	34	5		21	35	13	1
hail	182	178	214	166		72	61	62	55
rain	42	27	63	118		17	8	33	32

Table 36: Training and test confusion matrices PINV P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	0	1	167	12		0	0	70	0
wind	0	1	193	6		0	0	70	0
hail	0	2	709	29		0	0	250	0
rain	0	0	218	32		0	0	90	0

Table 37: Training and test confusion matrices PINV P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	333	35	14	28		97	10	11	22
wind	10	300	1	19		7	84	3	16
hail	21	31	68	60		10	18	18	24
rain	0	5	2	443		2	9	9	140

Table 38: Training and test confusion matrices PINV P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	9	11	7		13	8	37	12
wind	21	128	51	0		17	5	46	2
hail	8	25	691	16		35	42	140	33
rain	2	1	14	233		14	4	33	39

5.8 MLP results

Results for testing and training sets using

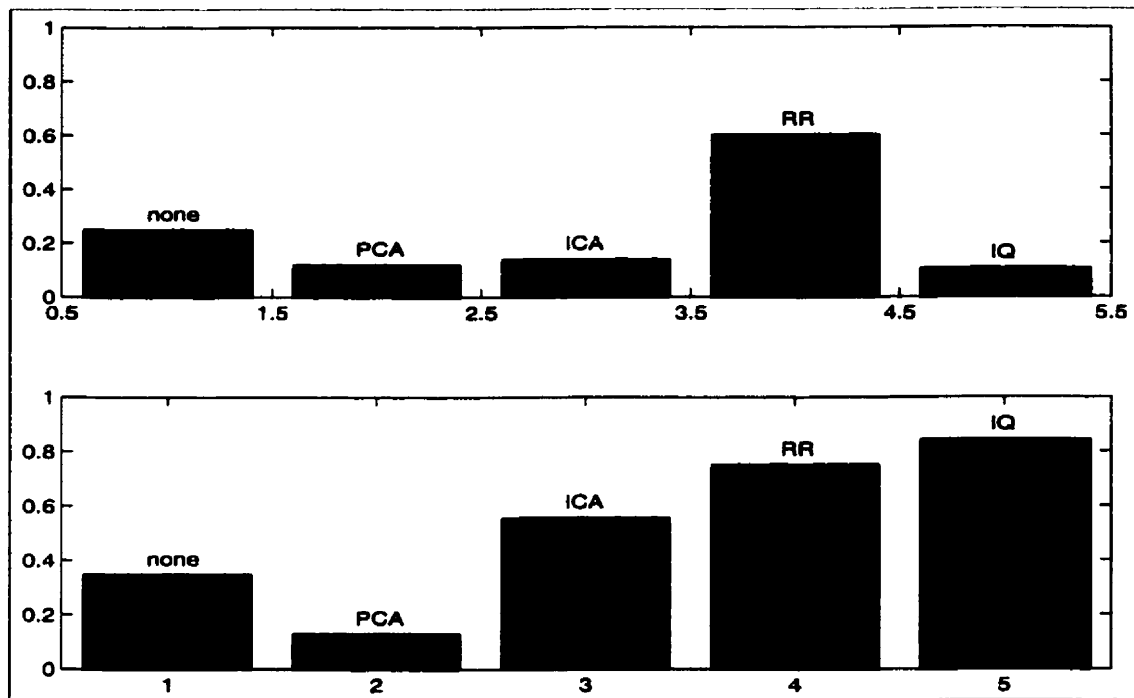


Table 39: Training and test confusion matrices MLP P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	136	6	208	10		14	2	122	2
wind	4	136	260	0		14	6	118	2
hail	10	18	1436	16		10	44	442	4
rain	10	8	388	94		12	4	156	8

Table 40: Training and test confusion matrices MLP P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	38	0	314	8		2	12	124	2
wind	2	12	384	2		2	12	124	2
hail	0	0	1470	10		8	12	472	8
rain	14	0	416	70		2	0	178	0

Table 41: Training and test confusion matrices MLP P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	160	20	168	12		2	22	92	24
wind	18	160	194	28		2	22	92	24
hail	30	58	1330	62		6	90	306	98
rain	8	6	114	372		0	30	114	36

Table 42: Training and test confusion matrices MLP P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	608	94	18	100		154	28	18	80
wind	136	388	16	120		58	84	4	74
hail	74	50	164	72		40	38	14	48
rain	104	120	18	658		8	42	12	258

Table 43: Training and test confusion matrices MLP P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	284	24	40	12		14	60	34	32
wind	24	294	80	2		8	44	80	8
hail	2	46	1416	16		78	96	274	52
rain	10	0	14	476		28	0	92	60

5.9 KNN with K = 1

Figure 32. Results for testing and training sets using KNN1

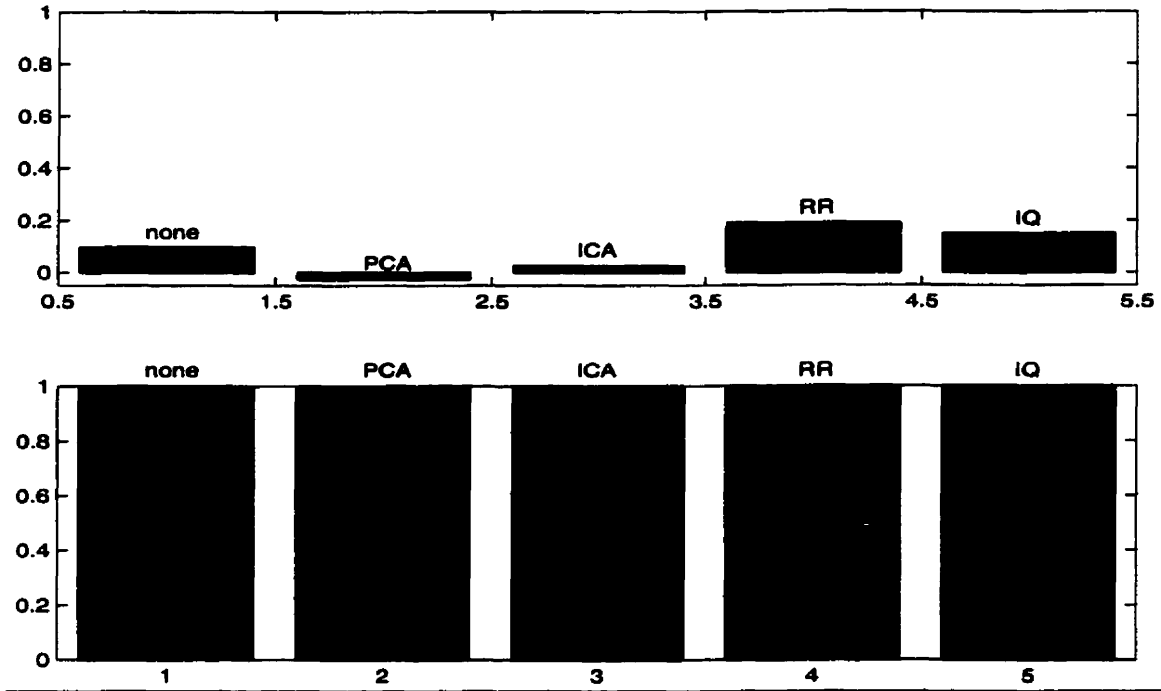


Table 44: Training and test confusion matrices KNN K = 1 P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		24	13	31	2
wind	0	146	54	0		19	3	39	9
hail	0	0	740	0		26	49	142	33
rain	0	0	0	250		11	12	39	28

Table 45: Training and test confusion matrices KNN k = 1 P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		8	14	39	9
wind	0	146	54	0		21	10	35	4
hail	0	0	740	0		34	53	121	42
rain	0	0	0	250		9	6	55	20

Table 46: Training and test confusion matrices KNN K = 1 P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		22	20	27	1
wind	0	146	54	0		26	19	23	2
hail	0	0	740	0		74	55	105	16
rain	0	0	0	250		20	22	41	7

Table 47: Training and test confusion matrices KNN k = 1 P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	76	53	216	65		40	14	65	21
wind	42	73	215	0		23	27	45	15
hail	23	10	128	19		6	8	56	0
rain	15	19	250	166		13	5	91	51

Table 48: Training and test confusion matrices KNN k = 1 P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		8	13	33	16
wind	0	146	54	0		16	14	40	0
hail	0	0	740	0		22	51	147	30
rain	0	0	0	250		11	0	33	46

5.10 KNN K= 3

Figure 33. Results for testing and training sets using KNN3

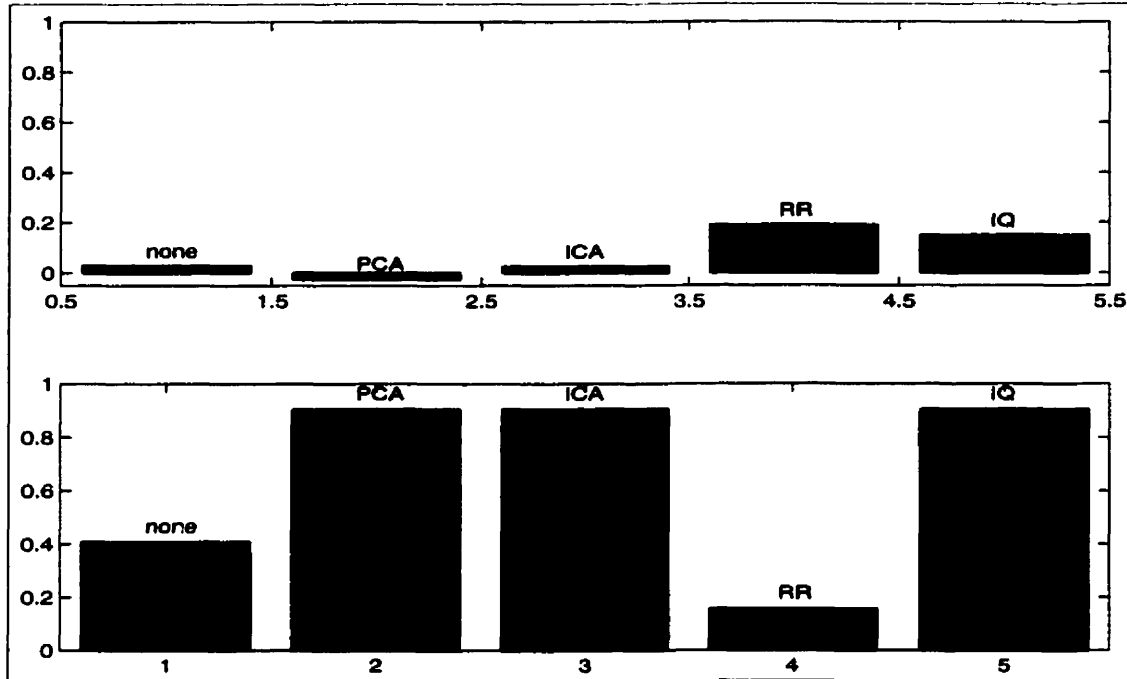


Table 49: Training and test confusion matrices KNN k = 1 P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	94	19	52	15		11	5	43	11
wind	5	57	101	37		13	0	49	8
hail	29	34	604	73		30	28	149	43
rain	11	10	102	127		5	5	49	31

Table 50: Training and test confusion matrices KNN k = 1 P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		8	14	39	9
wind	0	146	54	0		21	10	35	4
hail	0	0	740	0		34	53	121	42
rain	0	0	0	250		9	6	55	20

Table 51: Training and test confusion matrices KNN k = 1 P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		22	20	27	1
wind	0	146	54	0		26	19	23	2
hail	0	0	740	0		74	55	105	16
rain	0	0	0	250		20	22	41	7

Table 52: Training and test confusion matrices KNN k = 1 P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	76	53	216	65		40	14	65	21
wind	42	73	215	0		23	27	45	15
hail	23	10	128	19		6	8	56	0
rain	15	19	250	166		13	5	91	51

Table 53: Training and test confusion matrices KNN k = 1 P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	153	22	5	0		8	13	33	16
wind	0	146	54	0		16	14	40	0
hail	0	0	740	0		22	51	147	30
rain	0	0	0	250		11	0	33	46

5.11 FCM results

Figure 34. Results for testing and training sets using FCM, $c=16$

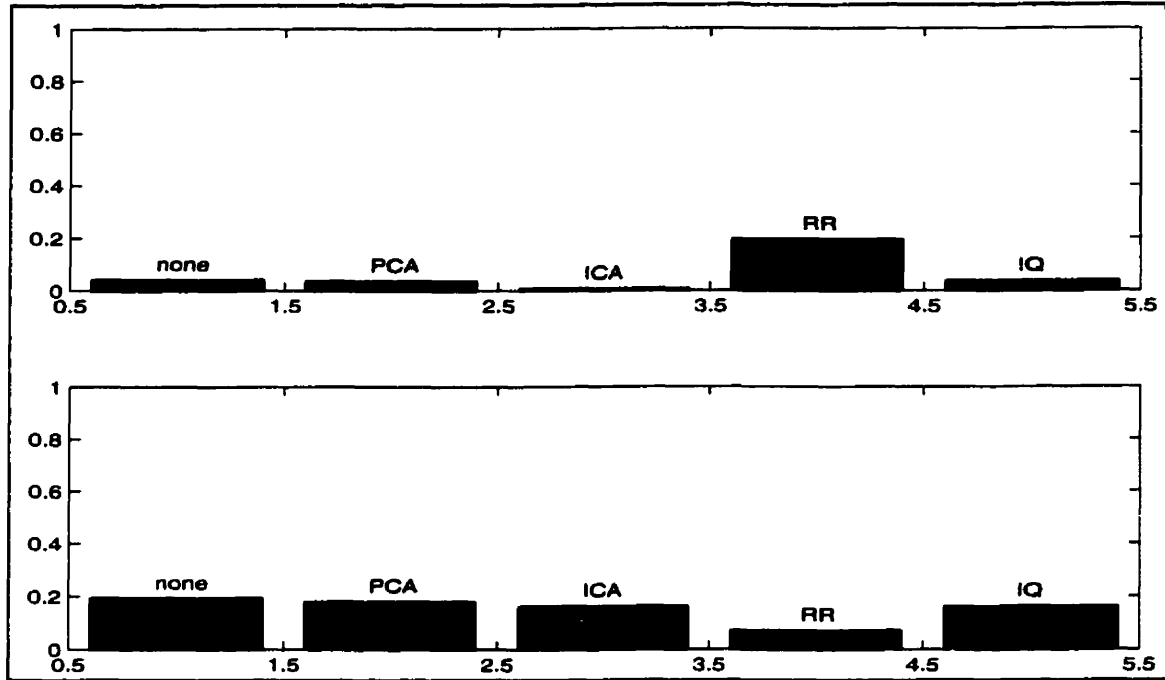


Table 54: Training and test confusion matrices FCM P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	74	14	266	6		6	18	112	4
wind	28	84	268	20		18	4	114	4
hail	58	72	1296	54		26	48	410	16
rain	30	36	344	90		4	20	126	30

Table 55: Training and test confusion matrices FCM P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	22	32	296	10		0	4	18	2
wind	12	82	288	18		0	2	12	0
hail	18	54	1350	58		0	12	104	2
rain	10	16	362	112		2	0	70	6

Table 56: Training and test confusion matrices FCM P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	42	6	300	12		52	18	70	0
wind	18	28	332	22		58	14	66	2
hail	34	22	1350	74		166	48	276	10
rain	12	14	342	132		54	6	118	2

Table 57: Training and test confusion matrices FCM P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	8	2	32	0		0	2	6	0
wind	10	6	28	4		2	4	0	0
hail	16	10	98	2		6	0	64	2
rain	2	2	48	6		0	0	10	0

Table 58: Training and test confusion matrices FCM P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	22	6	36	8		8	4	14	2
wind	18	20	42	0		8	4	16	0
hail	58	14	208	16		26	6	58	10
rain	12	0	66	22		4	0	26	6

5.12 Fuzzy Label Results

The fuzzy event labels were used only with MLP as backpropagation lends itself to weighted desired outputs. Recall that the fuzzy event label has a non-zero value in its desired event output where the magnitude of the response is determined by the relative distance in space and time from the schedule of events.

Figure 35. Results for testing and training sets using MLP with fuzzy labels

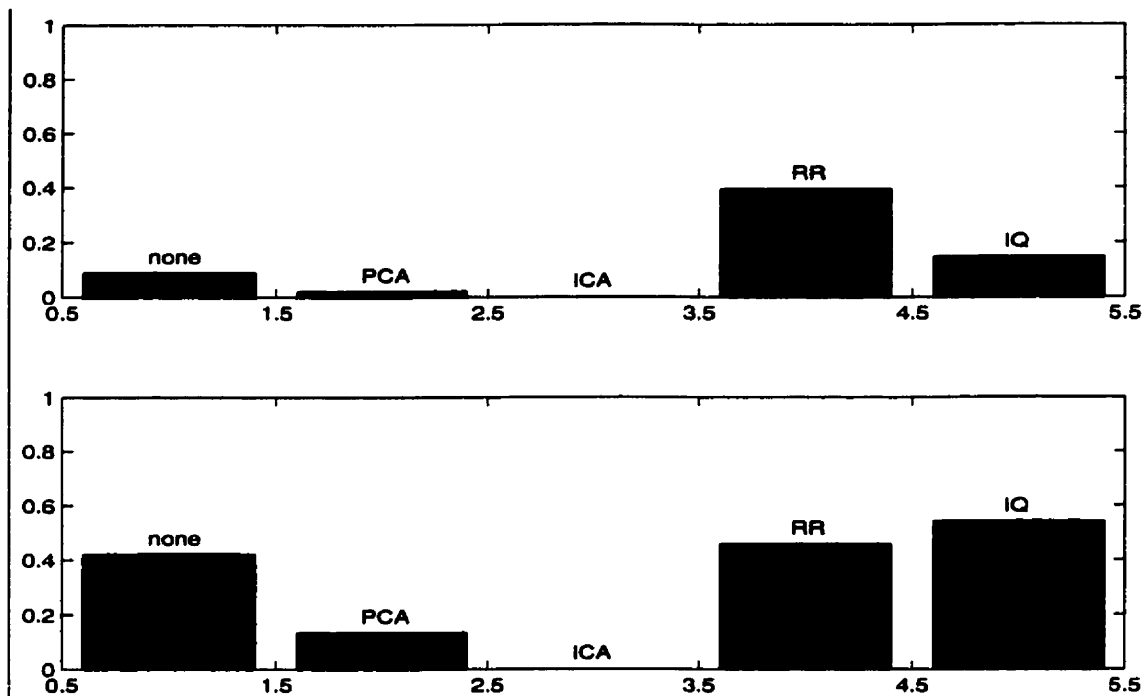


Table 59: Training and test confusion matrices MLP fuzzy labels P1

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	36	0	36	0		6	0	22	0
wind	2	24	54	0		6	0	22	0
hail	6	2	276	12		6	6	88	0
rain	4	0	58	38		2	0	30	4

Table 60: Training and test confusion matrices MLP fuzzy labels P2

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	12	0	60	0		0	4	24	0
wind	0	4	76	0		0	4	24	0
hail	0	0	296	0		2	4	92	2
rain	4	0	88	8		0	0	36	0

Table 61: Training and test confusion matrices MLP fuzzy labels P3

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	0	0	72	0		0	0	28	0
wind	0	0	80	0		0	0	28	0
hail	0	0	296	0		0	0	100	0
rain	0	0	100	0		0	0	36	0

Table 62: Training and test confusion matrices MLP fuzzy labels P4

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	120	18	18	32		48	10	0	6
wind	28	52	4	56		12	10	4	22
hail	12	18	34	20		12	0	6	14
rain	6	4	2	124		4	0	0	44

Table 63: Training and test confusion matrices MLP fuzzy labels P5

Train	tornado	wind	hail	rain	Test	tornado	wind	hail	rain
tornado	52	4	14	2		2	14	12	0
wind	6	36	38	0		0	12	16	0
hail	14	0	280	2		30	2	68	0
rain	4	0	60	36		8	6	16	6

5.13 Rejection classes

Limits were put upon the number of samples that could be rejected; we required that 80% of the training vectors remain classified by the threshold. FCM rejection using the mean and median cluster entropy rejected more than half of the samples and achieved a moderately improved classification [Ale99].

5.13.1 FCM rejection

A cluster entropy threshold was chosen by constraining the FCM classifier to assign labels to 80% of the samples. Test samples which were assigned (had the highest membership) in clusters with entropy exceeding the threshold were not classified. Due to time constraints, other methods of rejection (μ_{min} - subsets) were not implemented.

Table 64: Comparison of rejection class with FCM

preprocessing	testing accuracy	testing kappa score
no rejection	43.75	-0.0109
rejection	0.50	0.2174

Table 65: Test and reject confusion matrices FCM c= 16

normal	tornado	wind	hail	rain	reject	tornado	wind	hail	rain
tornado	0	1	6	0		0	0	2	0
wind	0	0	7	0		0	0	3	0
hail	0	5	20	0		0	1	4	0
rain	0	3	5	1		0	0	0	2

5.13.2 MLP rejection

Similarly to the FCM case, we define a threshold on the design set such that 80% of the design vectors will be accepted by the threshold. Alternatively, the percentage kept may be defined on the test set. We determine the threshold by measuring the difference between the two largest positive MLP outputs. This difference is sorted and the $(\text{floor}(0.8*N))$ th value chosen. Here N is the number of training samples.

Table 66: Comparison of rejection class with MLP

preprocessing	testing accuracy	testing kappa score
no rejection	0.4792	0.1585
rejection	1.0	1.0

Table 67: Test and reject confusion matrices MLP

normal	tornado	wind	hail	rain	reject	tornado	wind	hail	rain
tornado	0	4	3	0		1	0	0	0
wind	0	3	4	0		0	2	0	0
hail	5	1	19	0		0	0	3	0
rain	2	2	4	1		0	0	0	2

5.14 Probabilistic Results

Due to time constraints, probabilistic learning results were not included.

Preliminary studies showed that the nearest neighbour probabilistic scheme was comparable to the KNN in accuracy and that the kernel estimators performed poorly. The following matrices represent the probability that an event of type A is actually labelled type B. This information is necessary for the probabilistic methods and should be determined empirically.

Table 68. Class joint probability of mislabelled samples I

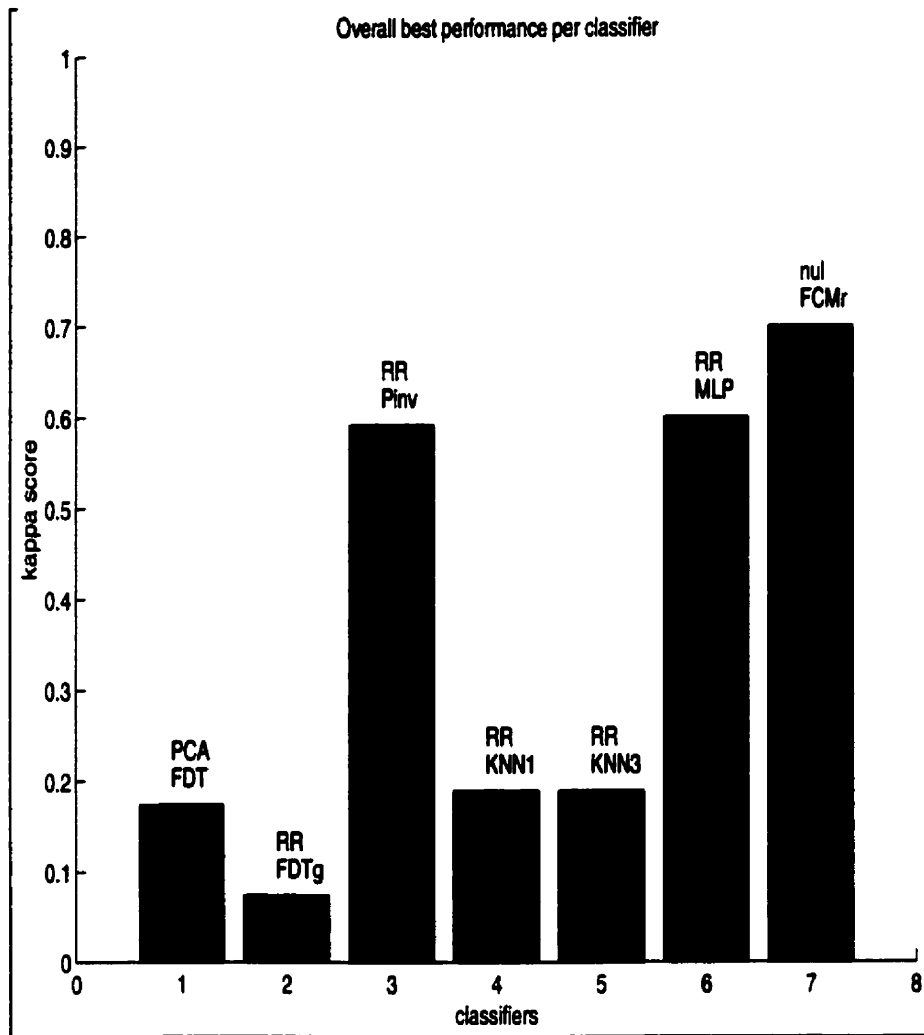
	tornado	wind	hail	rain
tornado	0.8	0.2	0	0
wind	0.2	0.8	0	0
hail	0	0	0.8	0.2
rain	0	0	0.2	0.8

Table 69. Class joint probability of mislabelled samples II

	tornado	wind	hail	rain
tornado	0.6	0.2	0.1	0.1
wind	0.2	0.6	0.1	0.1
hail	0.1	0.1	0.6	0.2
rain	0.1	0.1	0.2	0.6

5.15 Overall comparisons - classifiers and preprocessing

Figure 36. Best PreProcessing per Classifier



Note that the results listed above for FCM use rejection classes. Accuracy of 100% was achieved (on 12% of the test samples) with FCM rejection classes so a direct comparison should not be made. See Fig.38.

The dominance of robust reclassification as a preprocessing method is clear though one should consider the assumptions behind reclassification. They may or may not be justified.

Figure 37. Best Classifier per PreProcessing

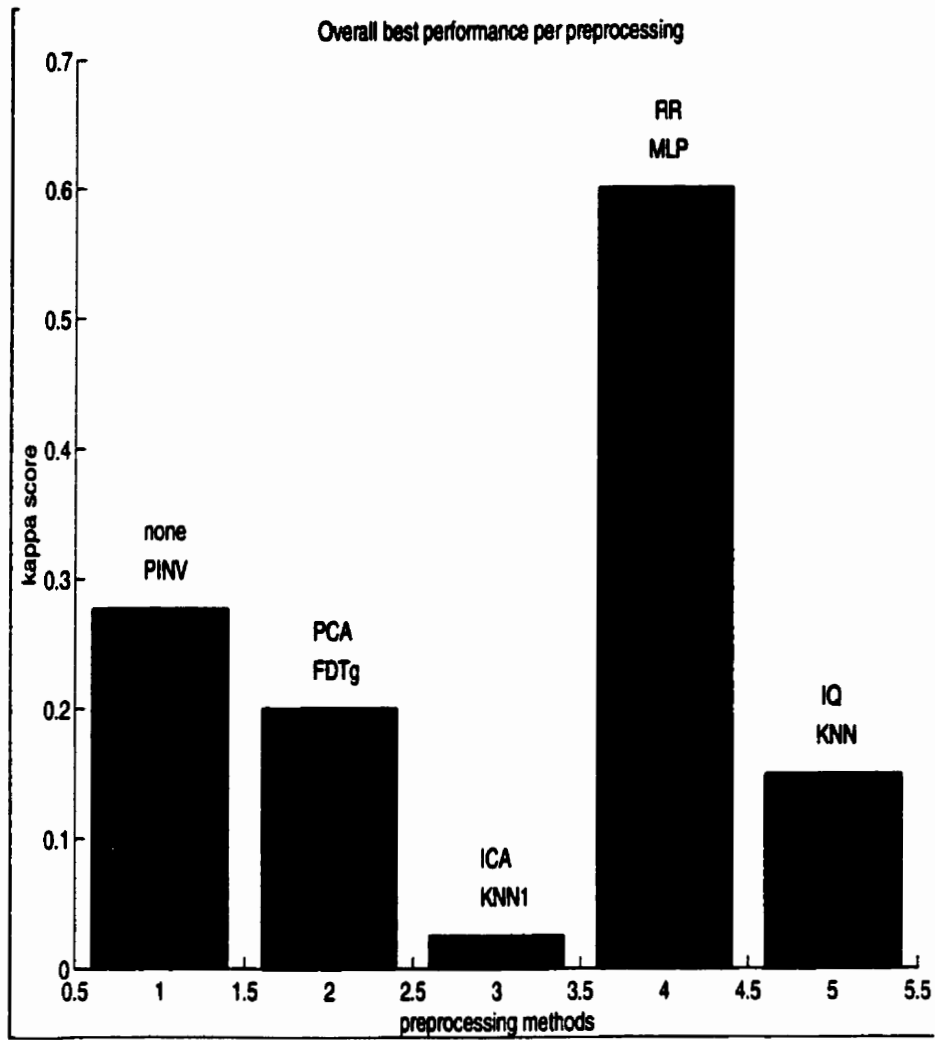


Figure 37 shows the most accurate classifier for each preprocessing method. The performance of the PINV classifier with no preprocessing demonstrates the loss of information that may accompany dimension reduction.

Figure 38. Rejection versus Accuracy

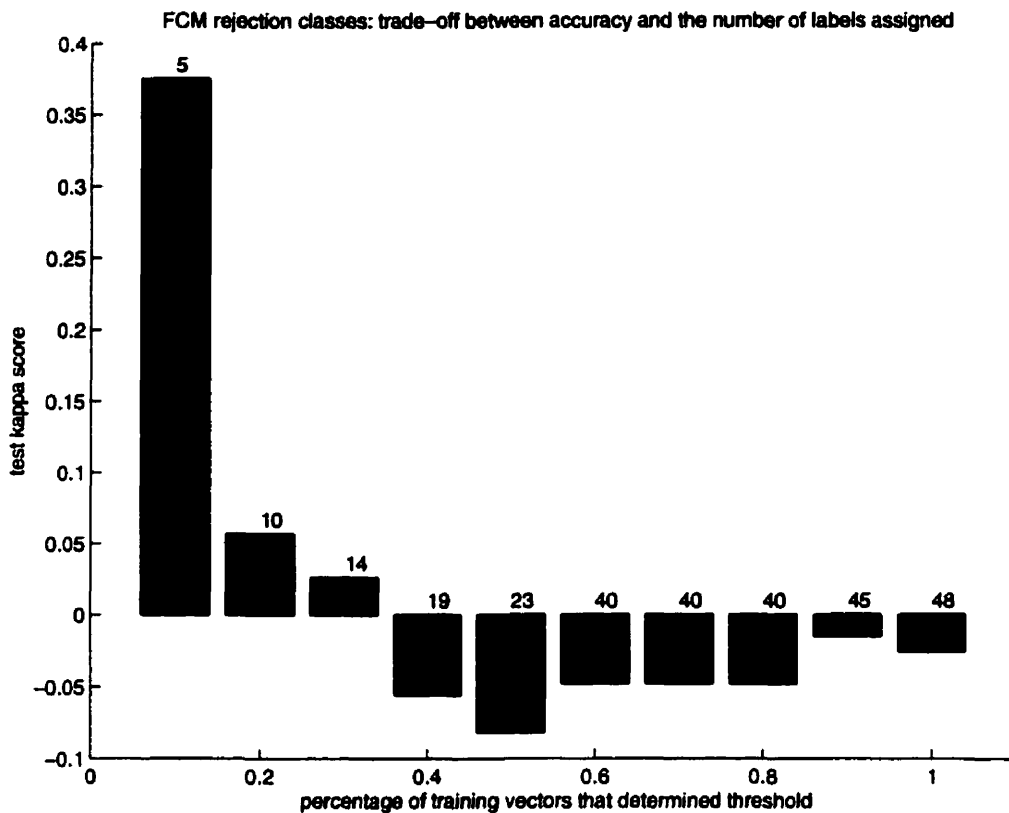
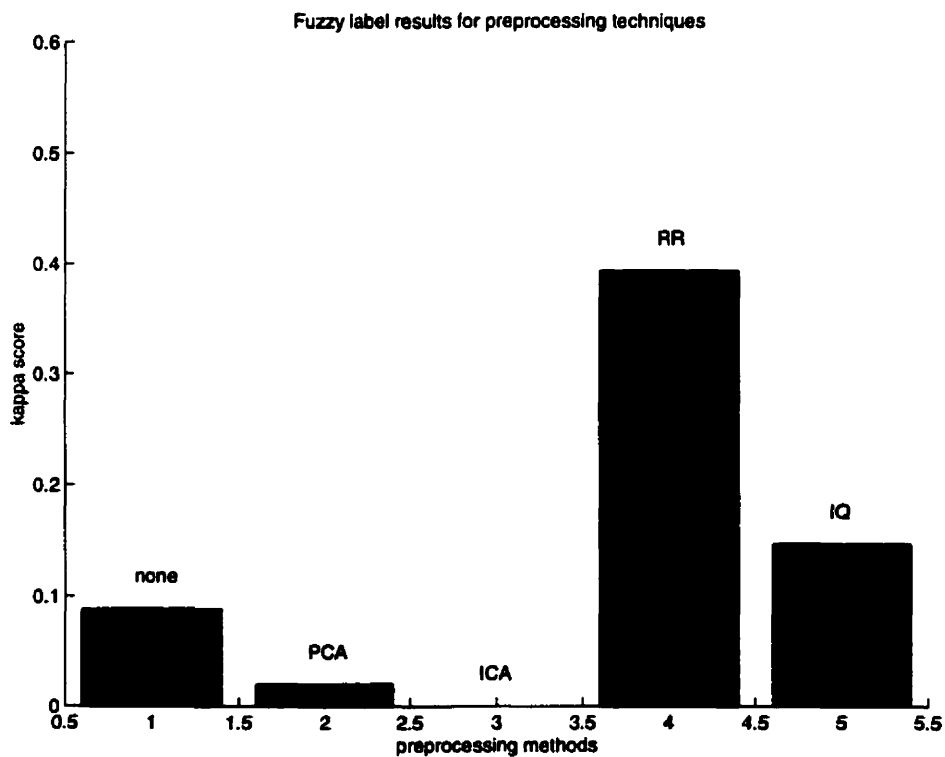


Figure 39. Fuzzy Labelling Results



Chapter 6 Conclusions and Recommendations

6.1 Conclusions

Convective storm cells can be classified to a fair degree using standard classifiers and RDSS-derived features. Improvements in classification accuracy may be realized through the use of rejection classes at the expense of the number of samples classified. Overall, preprocessing speeds convergence and removes noise from a data set (IQ) (PCA) (ICA). Dimensionality reduction preprocessing may remove information content (ICA) and methods for determining the number of components to use should be domain specific and open to alternate interpretation. Equating the number of independent components with the number of principal components for some accumulated variance showed very poor results. A more accurate use of independent components may involve estimating independent storm types and allowing for more than one prototype for each event type. For example, there may be three unique hail thunderstorms prototypes. Independent components could then be used to identify these prototypes.

Robust reclassification may be a valuable tool if empirical studies confirm the number of prototypes for each class. Again, reclassification makes certain assumptions about the distribution. If we err in estimating the number of clusters per class, reclassification will obscure class characteristics. A positive feature of robust reclassification is that test samples which would be reclassified may be flagged and this additional information used for analysis.

Non-linear iterative methods (FCM/MLP) may converge to local minima. This may be addressed by: multiple runs with random initial conditions, adaptive architectures, and expansion of the data set. A non-linear non-iterative classifier is an effective tool to benchmark more complex classifiers. Local minima (poor accuracy relative to the non-linear classifier) may be identified

and the iterative process restarted with a new random initialization or the architecture may be adapted.

6.1.1 Comparison of classifiers

The failure of such celebrated techniques as FCM and MLP to discriminate with ultimate accuracy for all types of preprocessing lies in the selection of the number of clusters and architecture. Improvements that should be made follow. The implementation of an adaptive not necessarily fully connected network may discriminate with a higher accuracy; loss of full connection will allow local refinements to the search space and speed training. Adaptation will augment the existing network architecture with neurons at error prone positions. The distinguishing performance of the pseudo-inverse shows that global information is applicable to the classification of these cells; this information could be used auxiliary to a neural network.

6.1.2 Comparison of preprocessing

Robust reclassification allowed almost all classifiers to improve their performance more than other preprocessing alternatives (including no preprocessing). An improvement on the application of robust reclassification that would hold more weight would be to cluster each class individually and define outliers for each class cluster. This would remove the presupposition that the distributions are unimodal and would provide a higher degree of tractability for overlapping distributions.

6.2 Recommendations

Based on preliminary studies not listed, augmenting the techniques used in this thesis with data of cell raw reflectivity values will improve performance. This data is seen to encapsulate the local features of such cells and provide independent data for cell discrimination. The dataset used in this

thesis should be expanded and such techniques as FCM and MLP refined. This study was constrained by the number of cell prototypes thought to exist. Expansion of the data set may confirm or correct our presuppositions.

6.3 Future research

Future research will be conducted along several lines. Modifications to pattern recognition techniques used in this thesis include:

- using alternate labelling schemes. Work on time averaging and storm type averaging to compose cell prototypes proved unsuccessful. Consideration of windows of cells around the scheduled event may be fruitful.
- use of an adaptive and/or parsimonious network.
- incorporate pruning of decision trees in order to generalize.
- distinguish metrics and distance measures for various data types (real, integer, n-ary). FDT should be updated to allow different number of partitions for different data types.
- alternate rejection methods and fuzzy labelling. Constraints upon the number of test samples may be defined instead of requiring a percentage of design samples to remain classified under the chosen threshold. Various membership functions for fuzzy labelling of the cells should be considered as well as multiple type labelling; assigning non-negative labels for two or more storm types.
- incorporate other current features of RDSS, namely the cell product maps. This data is currently available and provides information regarding the RDSS products such as cell VIL, BWER, etc...This introduces a methodology for comparison since each cell product map has a unique size.
- record the classification rate for each sample explicitly when n of N matched cells are in the training set of a classifier. Any correlation would suggest designing classifiers based on n samples from each matched group.

Expansion of the dataset may be realized to include data from event-years and radar archives. Unsupervised learning and clustering of all the cells that RDSS collects, regardless of ground truth, may provide adequate cell prototypes. Self-organising feature maps (SOFM) and group method of data handling (GMDH) [Far84] are other possible classifiers as are radial basis and probabilistic neural networks.

Currently time series analysis on the evolution of a storm cell is being conducted as well as the use of wavelets to analyse the raw reflectivity values and histograms of the cells.

Collaboration with the National Severe Storms Laboratory in Norman, OK to procure empirical measures for mis-classification tables has also been initiated. Simulated annealing as an alternate method of labelling has also achieved promising initial results [Li99].

7.0 Appendix: Matlab Files

The following files are from a pattern recognition toolbox developed and collected for this thesis. Matlab commands for each preprocessing method are listed in full as well as implementations for most of the classifiers. The FDT code is written in C by Dr. Z. Sosnowski while the independent component module uses version 3.3 of runica(), available from SALK (<http://www.cnl.salk.edu/~scott/ica.html>). The files are organized as follows: preprocessing, classifier code, miscellaneous functions. A brief description accompanies each file; complete documentation does not yet exist.

Preprocessing Modules

```
*****
% ICA matlab commands for Sload
% performs independent component analysis using
% runica() from SALK
*****

[Wic,Sphic] = runica(train_(:,1:col-1)', 'ncomps', num_C);

if SPHERE
train_ = train_*Sphic;
end

% dataset is now ranked by row labels IC
train = abs([train_*Wic' train(:,col)]);

% convert test set
test = abs([test(:,1:col-1)*Wic' test(:,col)]);

ica_mod_completed = 1
newlength_of_vectors = num_C + 1
wcol = 1:num_C
zcol = 1:num_C+1

return
*****
% PCA module using Matlab's svd PCA
% dataset is [row=no.dataentries,col=no.of dimensions]
% where the last column is the class label of the data
% data is already normalized
*****
% choose the number of components through a cumulative variance
% defined by percentage_of_var

    % derive param
    label = train(:,23);
    train_ = train(:,1:22);

    if firsttime
        num_C = scree3(train_,percent_of_var);
    end

min_frac = 0.00000001;          % include all and
```

```

                                % select only num_C
[train_,mean_tr,std_tr] = prestd(train_');
[train_,Xmat] = prepca(train_,min_frac);

train_ = train_';
train = train_(:,1:num_C);
train = [train label];

test_ = test(:,1:22);

for i = 1:length(std_tr)
    if std_tr(i) == 0
        std_tr(i) = 1;
    end
end

for i = 1:ntest
    test_(i,:) = (test_(i,:) - mean_tr') ./ std_tr';
end

test_ = test_*Xmat';
test = [test_(:,1:num_C) test(:,length(zcol))];

PCA_mod_completed = 1
newlength_of_vectors = num_C + 1
wcol = 1:num_C;
zcol = 1:num_C+1;

clear train_
clear test_

return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% scree plot -
% pass dataset (train) and percentage of variance to keep
% provides a discrete scree plot for the principal components of a given
% dataset
% min_frac is the minimum threshold for variance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [num_C] = scree(smallldata,percent_of_var)

[row,col] = size(smallldata);
points = zeros(1,col);
decr = 0.01;
min_frac = .99;
done = 0;

while ~done

[ptrans,transMat] = prepca(smallldata',min_frac);
[new_num,coll] = size(ptrans);

if new_num > 0 & points(new_num) == 0
    points(new_num) = min_frac;
end
end

```

```

min_frac = min_frac - decr;

if min_frac < 0
    done = 1;
end

end

k = 1;
sump = 0;
while sump < percent_of_var
    sump = sum(points(1:k));
    k = k + 1;
    if k == col-1
        sump = -1
    end
end

if sump == -1
    num_C = col;
else
    num_C = k -1
end

return
*****
% module for IQ, interquartile encoding
% pass train and test subset and N, the number of partitions
% use ONLY train set to calculate interquartile thresholds
% vectors = data set including column of labels
% N = number of regions to split data into ( e.g. for quartile, N = 4)
% out = encoded dataset
% partition = values that separate quartile regions ...
% LRLim = left and right limits of each fuzzy membership function
*****

function [train,test,partition,LRLim] = iqmod2(train,test,N)

vectors = train;

[ntrain,col] = size(vectors);
[ntest,col] = size(test);

label_tr = vectors(:,col);
label_ts = test(:,col);

fpq = fopen('partitions_IQ','wb');
partition = zeros(col-1,N+1);

for i = 1:col-1          % do for all dimensions less tag

% calculate thresholds for data ( quartile, etc.. measurements )
clear Q;
Q(i,1) = min(vectors(:,i));
Q(i,N+1) = max(vectors(:,i));

```

```

% partitions are based on distribution
data = sort(vectors(:,i));

incr = (ntrain+1)/N;
fprintf(fpq, 'data points chosen are : ');

    for j = 2:N
        Q(i,j) = data(ceil(incr*(j-1)));
        fprintf(fpq, ' %d ', ceil(incr*(j-1)));
    end
    % j = N
partition(i,:) = Q(i,:);

#####
for j = 1:N
left_lim(i,j) = (3*Q(i,j) - Q(i,j+1))/2;
right_lim(i,j) = (3*Q(i,j+1) - Q(i,j))/2;
end

LRlim(:, :, i) = [left_lim(i, :); right_lim(i, :)];

% print partition for reference
for j=1:N+1
fprintf(fpq, ' %f ', j, Q(j)); end
fprintf(fpq, '\n\n fuzzy domain limits: \n');
for j = 1:N
fprintf(fpq, ' %f %f \n', left_lim(j), right_lim(j)); end

% ensure that all Q are distinct
for kk = 1:N+1
nb(kk) = length(find(Q(i,:) == Q(i, kk) ) );
end
if max(nb(kk) > 1)
fprintf(' equal consecutive Q\n');
end
#####

    for k = 1:ntrain          % for every training vector
clear x; x = vectors(k,i);

    for j = 1:N              % each partition (Fuzzy Encoding)
u(j) = 0;

    if (left_lim(i,j) <= x ) & (x <= right_lim(i,j))

        if Q(i,j+1)==Q(i,j) & x == Q(i,j)
            u(j) = 1;
        elseif Q(i,j+1)==Q(i,j) & x ~= Q(i,j)
            u(j) = 0;

        else
u(j) = 1- abs( x - (Q(i,j) + Q(i,j+1))/2)/( Q(i,j+1)-Q(i,j) );

            if u(j) > 1
                u(j) = 1;
            elseif u(j) < 0
                u(j) = 0;
            end
        end
    end
end

```

```

        end
    end
end
        % x within limits
        % j = N

    out(k,1+(i-1)*N:i*N) = u;

    end
        % k = ntrain

#####
% insert test into cycle
    for k = 1:nctest % for every test vector
        clear x; x = test(k,i);

        for j = 1:N
            % each partition (Fuzzy Encoding)
            u(j) = 0;

            if (left_lim(i,j) <= x ) & (x <= right_lim(i,j))

                if Q(i,j+1)==Q(i,j) & x == Q(i,j)
                    u(j) = 1;
                elseif Q(i,j+1)==Q(i,j) & x ~= Q(i,j)
                    u(j) = 0;

                else
                    u(j) = 1- abs( x - (Q(i,j) + Q(i,j+1))/2)/( Q(i,j+1)-Q(i,j) );

                    if u(j) > 1
                        u(j) = 1;
                    elseif u(j) < 0
                        u(j) = 0;
                    end
                end
            end
        end
        % x within limits

    end
        % j = N

    out_ts(k,1+(i-1)*N:i*N) = u;

    end
        % k = nctest

end
    % num of dimensions i = col-1

train = [out label_tr];
test = [out_ts label_ts];

return
#####
% robust reclassification - MAD median average deviation
% see newcells/mad.m for module
% alternate formulations exist - see paper
% default coefficient is 2.5 (=> 99 % of normal data included)
% perform analysis on original data values ( see data variable )

[row,col] = size(data);
rrlabels = data(:,col);

% calculate tau and medians

```

```

tau = zeros(nclasses,col-1);
med = zeros(nclasses,col-1);

for i = 1:nclasses
    list = find(data(:,col) == i);
for i = 1:nclasses
    list = find(data(:,col) == i);
    tau(i,:) = mad(data(list,1:col-1));
    med(i,:) = med2(data(list,1:col-1));
end

% calculate membership in class medoids
for k = 1:ncells
    D = zeros(1,nclasses);
    for j = 1:nclasses
        for i = 1:col-1
            if abs(data(k,i)-med(j,i) ) <= Z*tau(j,i)    % see thesis RR
                D(j) = D(j) + 1/(1+abs(data(k,i)- med(j,i) ) );
            end
        end
    end
end

% reclassify object
for j = 1:nclasses
    if D(j) == max(D)
        rrlabels(k) = j;
    end
end
end

% reassignment
% compute statistic with only robust labels
% write numbers of reclassified samples to file

reclassified = zeros(nclasses);
rrconftr = zeros(nclasses);

for j = 1:nclasses
    list = find(data(:,col) == j);
    for k = 1:length(list)
        if ( rrlabels(list(k)) ~= j)
            reclassified(j,rrlabels(list(k))) = reclassified(j,rrlabels(list(k))) + 1;
            rrconftr(rrlabels(list(k))) = rrconftr(rrlabels(list(k))) + 1;
        end
    end
end

for j = 1:nclasses*nclasses
    fprintf(frob,'%d ',reclassified(j));
end

fprintf(frob,'\n');

return
*****

```

Classifiers

```
*****
% multilayer perceptron ( or ff net)

Sout = nclasses;          % number of output neurons determined by
                          % number of classes

%S3 = 0;
%S4 = 0;

[S1 S2 S3 S4]

F1 = 'tansig';
PR = [real(min(dataset(:,wcol))); real(max(dataset(:,wcol)))]';

P = train(:,wcol)';
Tn = traintarg';

if S2 == 0 & S3 == 0
net = newff(PR, [S1 Sout], {'tansig' 'tansig'});
elseif S2 > 0 & S3 == 0
net = newff(PR, [S1 S2 Sout], {'tansig' 'tansig' 'tansig'});
elseif S2 > 0 & S3 > 0 & S4==0
net = newff(PR, [S1 S2 S3 Sout], {'tansig' 'tansig' 'tansig' 'tansig'});
else
net = newff(PR, [S1 S2 S3 S4 Sout], {'tansig' 'tansig' 'tansig' 'tansig'
'tansig'});
end

mynetparam

% set labels to fuzzy labels
if Fuzzlab
for i = 1:ntrain
Tn(:,i) = Tn(:,i)*flabeltr(i);
end
end

net = init(net);

old_conf1 = zeros(nclasses);
statick = 0;
done = 0;          % train until training confusion matrix is diagonal
count = 1;

*****          training loop
while ~done
[net,tr] = train(net,P,Tn);
Ao = sim(net,P);
check = maxrowrbf(Ao',nclasses);
[conf1] = confuse3(check,conftr,nclasses)

if kappa(conf1) > 0.8 | count > 40
done = 1;
end
```

```

% count number of time training confusion matrix is static
if old_conf1 == conf1
    statick = statick + 1;
else
    statick = 0;
end

if statick > 5
'statick greater than 5 sessiosn'
done = 1;
end

count = count + 1;
old_conf1 = conf1;
statick
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% training loop

Pts = test(:,wcol)';
Ts = test(:,limit)';

A = sim(net,Pts);
outmod = maxrowrbf(A',nclasses);
[conf2] = confuse3(outmod,conf2ts,nclasses)

kappa(conf1)
kappa(conf2)

cmtr(:, :,xs) = conf1;
cmts(:, :,xs) = conf2;
ovcmtr = ovcmtr + conf1;
ovcmts = ovcmts + conf2;

save net

return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
% PINV

[row,col] = size(train);
weights = pinv(train(:,1:col-1))*traintarg;

% cleans up matrix of output values
out = test(:,1:col-1)*weights;
[outmod,nguess1] = maxrowrbf(out,nclasses);

% calculate percent of correct classifications
check = train(:,1:col-1)*weights;
[checkmod,nguess2] = maxrowrbf(check,nclasses);

[conf2] = confuse3(outmod,conf2ts,nclasses);
[conf1] = confuse3(checkmod,conftr,nclasses);

return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% k nearest neighbors for training and test set
% given training set, for each test vector determine the
% k nearest neighbors and assign the majority class

```



```

clear dmark
clear dmark2

for i = 1:ntrain
    for j = 1:ntrain
        dmark(j) = dist(train(i,wcol),train(j,wcol)');
    end

dmark2 = sort(dmark);

% train set
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% consider knum neighbors

        nvote = zeros(1,nclasses);

        for j = 1:knum
            loc = find(dmark == dmark2(j));
nvote(train(loc,zcol(length(zcol)))) = nvote(train(loc,zcol(length(zcol))))
+1;
        end

        for j = 1:nclasses
            if nvote(j) == max(nvote)
                senator(i) = j;
            end
        end

end    % i = ntrain

[conf1] = confuse3(senator,conftr,nclasses);

% test set
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:nctest
    for j = 1:ntrain
        dmark(j) = dist(test(i,wcol),train(j,wcol)');
    end

    dmark2 = sort(dmark);

    nvote = zeros(1,nclasses);

    for j = 1:knum
        loc = find(dmark == dmark2(j));
nvote(train(loc,zcol(length(zcol)))) = nvote(train(loc,zcol(length(zcol))))
+1;
    end

    for j = 1:nclasses
        if nvote(j) == max(nvote)
            senator(i) = j;
        end
    end

end    % i = nctest

[conf2] = confuse3(senator,conftr,nclasses);

```

```

return
*****
% hard or fuzzy c-means ( k-means )
% Fuzzy c- means clustering based on Bezdek 1995 (IEEE FS vol3 no3)
%   p is the dimension of vectors
%   n the number of vectors
%   x = vector, v = cluster center, u = degree of membership to cluster
% parameters c = number of clusters
%           error and distance norm
%           T = iteration limit
%           err = error threshold
%           m = exponent , generally m = 2, ( or 1 < m < 5 )

limit = zcol(length(zcol));

    n = ntrain;
    x = train(:,1:limit-1);

% select centers from all classes
% replaces ss = randperm(ntrain);

    initcenters

clear vold
if RAND_CENTERS
    for i = 1:c
        vold(i,:) = x(ss(i),1:length(zcol)-1);    % randn(1,length(wcol));
    end
    noise = randn(size(vold))/100;
    vold = vold + noise;

else
    % choose centers based on class type

end
done = 0; i = 0;

'running fuzzy c-means'

while(~done)

i = i + 1;
%i/erriter
    calcu;
    calcv;

    E(i) = sum(sum(vnew(:,wcol) - vold(:,wcol)));
    vold = vnew;

% if no vector clusters to any of the new centers, place this center
% vector in the largest heterogeneous class offset from that center
% checks purity, and gets count of classes per cluster

    checkpop

    if (abs(E(i)) < err | i > erriter)
        done = 1;
    end
end

```

```

end      % while

% evaluate fcm : for each class, determine which cluster they dominate.
% if two classes share a cluster, -> contention

centers = vold;
[cluster_validity_index] = XieBeni(train,centers,u')

num = conftr + confcs;
modd = maxrowrbf(u,c);
modd_tr = modd;
modd_ts = zeros(1,ntest);

getcount

fcmevall

cmtr(:, :,xs) = conf1;
cmts(:, :,xs) = conf2;
ovcmtr = ovcmtr + conf1;
ovcmts = ovcmts + conf2;

return
*****
confuse3.m
% takes target and actual vector and return confusion matrix

function [confusion] = confuse3(act,num,nclasses)

confusion = zeros(nclasses);

start = 1;
last = num(1);

for k = 1:nclasses
for i=start:last
    for j = 1:nclasses
        if (act(i) == j);
            confusion(k,j) = confusion(k,j) + 1;
        end
    end
end

end

start = last+1;
if k~= nclasses
last = last + num(k+1);
end
end
return
*****
fcmeval.m
% evaluation method for fcm; take max membership value and assign hard label

set1 = 1;
clusterclass
conf1 = confusion;

```

```

set1 = 0;
clusterclass
conf2 = confusion;
*****
clusterclass.m
% uses cluster information to classify vectors
if set1
vectors = train;
[nvectors,col] = size(vectors);
count = count_tr;
conftr= conftr;
else
vectors = test;
[nvectors,col] = size(vectors);
count = count_ts;
conftr = conftr;
end

count

[c,col] = size(centers);

conftr = zeros(1,nclasses);
for i = 1:nclasses
conftr(i) = length( find(vectors(:,limit) == i ) );
end

for i = 1:nvectors

    for j = 1:c                % number of clusters
dtov(j) = dist(vectors(i,wcol),centers(j,wcol)');
end

    for j = 1:c
if dtov(j) == min(dtov)
        out(i) = centerclass(j);
        clusterassigned(i) = j;
        pos = j;
end
end

end

[confusion] = confuse3(out,conftr,nclasses);
p4 = sum(diag(confusion))/nvectors;

% calculate the entropy of each cluster
meminc = max(count_tr(:,1:c))./sum(count_tr(:,1:c));
pops = sum(count_tr);

% REJECT OPTION
% apply Bayesian approach with threshold for rejecting vector
% valid vectors = 0 if meminc = mean(meminc) for all i

' per changes number of vectors kept'

num_keep = floor(per*nvectors);
Bp4 = 0;

```

```

Bvalid = 0;

if set1 == 1
notgood = 1;
tcheq = 0.9;
while(notgood)
poss = find(meminc > tcheq);
    if sum(pops(poss)) >= num_keep
        notgood = 0;
    else
        tcheq = tcheq - 0.005;
    end
    clear poss;
end
thresholdB = tcheq;
end

for i = 1:nvectors
    if meminc(clusterassigned(i)) > thresholdB    % don't reject !
        Bvalid = Bvalid + 1;
        if vectors(i,limit) == out(i)
            Bp4 = Bp4 + 1;
        end
    end
end

off = 1;
for i = 1:nclasses
    list = find(vectors(:,limit)==i);
    confftsR(i) = length(find(meminc(clusterassigned(list))>thresholdB));
    outR(off:length(list)+off-1) = out(list);
    off = off + length(list);
end

[confR] = confuse3(outR,confftsR,nclasses);

if set1 == 1
TRassign = clusterassigned;
else
TSassign = clusterassigned;
end
clear clusterassigned

return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

8.0 References

[Abr63] Abramson N. Information Theory and Coding. United States of America; McGraw-Hill Inc, 1963.

[Agr70] Agrawala A. Learning with a Probabilistic Teacher. IEEE Transactions on Information Theory 1970; 16(4):373-379.

[Agr96a] Agresti A. An Introduction to Categorical Data Analysis. New York; John Wiley and Sons Inc, 1996.

[Ale99] Alexiuk M, Pizzi N, Pedrycz W. Classification of Volumetric Storm Cell Patterns. CCECE, 1999.

[Bar] Baras J, Deys S. Combined Compression and Classification with Learning Vector Quantization. Technical Research Report: Institute for Systems Research.

[Bar] Barni M, Cappellini V, Mecocci A. Comments on 'A Possibilistic Approach to Clustering'. IEEE Trans Fuzzy Systems 1996;4(3):393-396.

[Bel95] Bell A, Sejnowski TJ. An information-maximisation approach to blind separation and blind deconvolution. Neural Computation 1995;7(6):1004-1034.

[Bel] Bell A, Sejnowski TJ. The 'independent components' of natural scenes are edge filters. Vision Research. To appear.

[Ben97] Benn DE, Nixon MS and Carter JN. Robust eye center extraction using the Hough transform. Bigun J, Chollet G, Borgefors G, Ed. Proceedings of 1st Int. Conf. on Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science (IAPR) 1997. Berlin; Springer Verlag:3-9.

[Bez81] Bezdek JC. Pattern recognition with Fuzzy Objective Function Algorithms. New York: Plenum Press, 1981.

[Bez87] Bezdek JC. Analysis of Fuzzy Information Volume 1. Boca Raton (Fla); CRC Press, 1987.

[Bis94] Bishop CM. Neural Networks and their applications. Rev. Sci. Instrum 1994 ;65(6): 1803-1832.

[Bla] Blanchard DO. Comments on Mesoscale Convective Patterns of the Southern High Plains. Bulletin of the American Meteorological Society; 72:389-390.

[Bog] Bogacz R, Giraud-Carrier C. Supervised Competitive Learning for Finding Positions of Radial Basis Functions. Unpublished.

[Bou96] Bouzerdoum A, Cole RJ, Deer P. Classification of Satellite Imagery Based on Texture Features Using Neural Networks. Proceedings Fourth International Conference on Control, Automation, Robotics and Vision (ICARCV'96) 1996. Singapore;2257-2261.

[Bre84] Breiman Friedman Olshen Stone. Classification and Decision Trees. Wadsworth, 1984.

[Car] Carpenter GA, Grossberg S. Pattern Recognition by Self-Organizing Neural Networks. Cambridge: The MIT Press, 1991.

[Che73] Chernoff H. The use of faces to represent points in k-dimensional space graphically. Journal of the American Statistical Association 1973;68:361-367.

[Cho70] Chow CK. On Optimum Recognition Error and Reject Tradeoff. IEEE Trans IT 1970;16(1):41-46.

[Coo75] Cooper DB. On some Convergence Properties of 'Learning with a Probabilistic Teacher' Algorithms. IEEE Trans on Information Theory 1975;IT-VOL 21:669-702.

[Cov67] Cover TM, Hart PE. Nearest Neighbour pattern recognition. IEEE Trans IT 1967; IT-13:21-27.

[Cov91] Cover TM, Thomas JA. Elements of Information Theory. New York: John Wiley and Sons Inc, 1991.

[Cze94] Czezowski P. Topics in Pattern Recognition Using Unsupervised Learning, M.Sc. Thesis, University of Manitoba, 1994.

Dave RN, Krishnapuram R. Robust Clustering in Kohonen Networks for Vector Quantization. Unpublished.

[Den95] Denoeux T, Rizand P. Analysis of Radar Images for Rainfall Forecasting using Neural Networks. Neural Computations and Applications 1995;3:50-61.

[Dos94] Doswell III CA. Extreme Convective Windstorms: Current Understanding and Research. Report of the Proceedings of the US-Spain Workshop on Natural Hazards; 1994:44-55.

[Dud73] Duda RO, Hart PE. Pattern Classification and Scene Analysis. New York; John Wiley and Sons Inc, 1973.

[Due96] Duempelmann M, Elger CE. Separating EEG Spike-Clusters in Epilepsy by a Growing and Splitting Net. Artificial Neural Networks, Proceedings ICANN96, Berlin, Springer, 1996:239-244.

[Eag83] Eagleman JR. Severe and Unusual Weather. Lenexa (KS): Van Nostrand Reinhold Company Inc, 1983.

[Eve93] Everitt BS. Cluster Analysis. New York; Halstead Press, 1993.

[Far84] Farlow SJ. Self-Organizing Methods in Modeling: GMDH Type Algorithms. New York; Marcel Dekker Inc, 1984.

[Fed] Federal Meteorological Handbook Number 11: Doppler Radar Meteorological Observations. Federal Co-ordinator for Meteorological Services and Supporting Research: Washington DC.

[Gat89] Gath I, Geva AB. Unsupervised Optimal Fuzzy Clustering. IEEE Trans Pattern Analysis and Machine Intelligence 1989; PAMI-11(7):773-781.

[Geb] Gebhardt J, Kruse R. Learning Possibilistic Networks From Data. Unpublished.

[Geo95] George KM, Carrol JH, Deaton E, Oppenheim D, Hightower J. Editors. Applied Computing 1995. Nashville, ACM Press, 1995.

[Gim74] Gimlin DR. A Parametric Procedure for Imperfectly Supervised Learning with Unknown Class Probabilities. IEEE Trans on Information Theory 1974; 20: 661-663.

[Gli78] Glick N. Additive Estimators for the Probabilities of Correct Classification PR 1978;10:221-222.

[Gos96] Gosselin B. Multilayer Perceptrons Combination Applied to Handwritten Character Recognition. Neural Processing Letters 1996;3:3-10.

[Gre80] Greblicki W. Learning to Recognize Patterns with a Probabilistic Teacher. Pattern Recognition 1980;12:159-164.

[Gus79] Gustafson BB, Kessel WC. Fuzzy clustering with a fuzzy covariance matrix. Proceedings IEEE CDC, San Diego, 1979.

[Ha97] Ha TM. The Optimum Class-Selective Rejection Rule IEEE Trans PAMI 1997; 19(6):608-614.

[Han86] Hand DJ. Recent advances in error rate estimation. Pattern Recognition Letters 4 (1986) 335-346 North Holland

[Van98] van Hateren JH, van der Schaaf A. Independent Component filters of natural images compared with simple cells in primary visual cortex. Proceedings of the Royal Society London 1998;265:359-366.

[Her] Herrera F, Lozano M, Moraga C. Hierarchical Distributed Genetic Algorithms. Technical Report #DECSAI-97-01-18 University of Granada

[Hof] Hoffman T, Buhmann JM. An Annealed 'Neural Gas' Network for Robust Vector Quantization. Unpublished.

[How90] Howard PJA, An Introduction to Environmental Pattern Analysis. Park Ridge (NJ): The Parthenon Publishing Group, 1990.

[Hur96] Hurri J, Hyvarinen A, Karhunen J, Oja E. Image feature Extraction using Independent component analysis. Proc NORSIG'96: 475-478.

[Hyv97] Hyvarinen A, Oja E. A fast Fixed-Point Algorithm for Independent Component Analysis Neural Computation 1997;9: 1483-1492.

[Hyv96] Hyvarinen A, Oja E. Simple Neuron Models for Independent Component Analysis. International Journal of Neural Systems 1996;7(6):671-687.

[Imy92] Imy DA, Pence KJ, Doswell III CA. On the need for Volumetric radar data when issuing severe thunderstorm and tornado warnings. National Weather Digest 1992; 17(4):2-18.

[Jai] Jain A, Zongker D. Feature Selection: Evaluation, Application and Small Sample Performance. IEEE PAMI 1997; PAMI-19:153-158.

[Jeo94] Jeon B, Landgrebe DA. Fast Parzen Density Estimation Using Clustering-Based Branch and Bound. IEEE Trans PAMI 1994;16 (9): 950-954.

[Kan82] Kandel A. Fuzzy Techniques in Pattern Recognition. New York; John Wiley and Sons Inc, 1982.

[Kar97] Karhunen J, Oja E, Wang L, Vigarrio R, Joutsenalo J. A class of Neural

Networks for Independent Analysis. IEEE Transactions on Neural Networks 1997;8(3): 486-503.

[Kar1] Karhunen J. Neural Approaches to Independent Component Analysis and Source Separation. Unpublished.

[Kar2] Karhunen J, Malaroiu S. Local Independent Component Analysis using Clustering. Unpublished.

[Kau75] Kaufmann A, Introduction to the Theory of Fuzzy Subsets Volume 1. New York: Academic Press, 1975.

[Kec94] Keck A, Legal L. Automated Recognition of Severe Summer Weather Features in Radar Data. 5th Annual Geographic Information Seminar Resource Technology 94 Symposium 1994, Toronto.

[Kel] Keller D, Sahami M. Toward Optimal Feature Selection. Unpublished.

[Khu94] Khuri S, Back T, Heitkotter J. An Evolutionary Approach to Combinatorial Optimization Problems. Proc CSC'94; Phoenix(AZ):ACM Press, 1993.

[Kit78] Kittler J. Classification of Incomplete Pattern Vectors Using Modified Discriminant Functions. IEEE Trans Comp 1978; c-27(4):367-375.

[Kit86] Kittler J. Feature Selection and Extraction in the Handbook of Pattern Recognition and Image Processing. Academic Press Inc, 1986.

[Kla] Klawonn F, Kruse R. Derivation of Fuzzy Classification Rules from Multidimensional Data. Unpublished.

[Koh88] Kohonen T. An introduction to Neural Computing. Neural Networks 1988;1:3-16.

[Kri90] Krishnan T, Nandy SC. Efficiency of Discriminant Analysis when Initial Samples are Classified Stochastically. Pattern Recognition 1990; 23(5):529-537.

[Kri96] Krishnapuram R, Keller J. The Possibilistic C-Means Algorithm: Insights and Recommendations. IEEE Trans on Fuzzy Systems 1996; 4(3):385-396.

[Kri93] Krishnapuram R, Keller J. A possibilistic approach to Clustering. IEEE TRans on Fuzzy Systems 1993; 1(2):98-109.

[Krz88] Krzanowski W. Principals of Multivariate Analysis. New York; Oxford University Press, 1988.

[Kur] Kurkova V, Smid J. An incremental Architecture Algorithm for Feedforward Neural Nets. Proc. IEEE Workshop on Computer-Intensive Methods in Control and Signal Processing. In press.

[Lak97] Lakshmanan T, Witt A. A Fuzzy Logic Approach to Detecting Severe Updrafts. AI Applications 1997;11(1):1-12.

[Lau79] Launer RL, Wilkinson GN. Robustness in Statistics. New York Academic Press, 1979.

[Lee91] Lee S, Kil RM. A Gaussian Potential Function Network with

Hierarchically Self-Organizing Learning. Neural Networks 1991;4: 207:224.

[Lee99a] Lee T, Girolami M, Sejnowski TJ. Independent Component Analysis using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources. Neural Computation 1999; 11(2):409-433.

[Lee91b] Lee Y. Handwritten Digit Recognition Using K Nearest-Neighbour, Radial Basis Function, and Backpropagation Neural Networks. Neural Computation 1991;3:440-449.

[Lem80] Lemon LR. Severe Thunderstorm Radar Identification Techniques and Warning Criteria. National Oceanic and Atmospheric Administration Technical Memorandum NWS NSSFC-3 1980.

[Les83] Lesaffre E. Normality tests and transforms. Pattern Recognition Letters 1 1983; North Holland Publishing Company:187-199.

[Li99] Li CP. Private communication, 1999.

[Lip97] Lippmann RP. An Introduction to Computing with Neural Nets, IEEE ASSP 1987.

[Mar81] Maritz JS. Distribution-Free Statistical Methods. Chapman and Hall Limited, 1981.

[McC88] McCulloch N, Ainsworth WA. Speaker Independent Vowel Recognition using a Multi-Layer Perceptron. Proceedings of Speech'88, Edinburgh, 1988.

[Mur94] Murthy SK, Kasif S, Salzberg S. A system for Induction of Oblique Decision Trees. Journal of Artificial Intelligence Research 2 1994;1-32.

[Nar93] Narazaki H, Ralescu AL. An Improved Synthesis Method for Multilayered Neural Networks Using Qualitative Knowledge. IEEE Trans on Fuzzy Systems 1993;1(2):125-137.

[Nau94] Nauck D. A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches. Proc Fuzzy-Systeme'94, and GI-Workshop, Siemens Corporation 1994.

[Orr95] Orr MJL. Local Smoothing of Radial Basis Function Networks. 1995 International Symposium on Neural Networks, Hsinchu (Taiwan).

[Orr] Orr MJL. Regularisation in the Selection of Radial Basis Function centers. Unpublished.

[Pal95] Pal NR, Bezdek JC. On Cluster Validity for the Fuzzy C-means Model IEEE Trans on Fuzzy Systems 1995; 3(3).

[Pao89] Pao Y. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley Publishing Co, 1989.

[Paw93] Pawlak M. Kernel Classification Rules from Missing Data. IEEE Trans IT 1993;39(3):979-988.

[Paw88] Pawlak M. On the Asymptotic Properties of Smoothed Estimators of the Classification Error Rate. PR 1988;21:515-524.

[Paw99] Pawlak M. Pattern Recognition graduate course notes. Winnipeg,

University of Manitoba, 1999.

[Ped95] Pedrycz W. Fuzzy Sets Engineering. Boca Raton; CRC Press, 1995.

[Piz95] Pizzi N. Report on Storm Cell Classification Using Fuzzy Clustering and Artificial Neural Networks. InfoMagnetics Technology Report 1995.

[Piz97] Pizzi N. Pattern Recognition Using Robust Discrimination and Fuzzy Set Theoretic Preprocessing, Ph.D Thesis University of Manitoba 1997

[Por88] Porat M, Zeevi YY. The generalized Gabor scheme of image representation in biological and machine vision. IEEE Trans PAMI 1988; 10(4):452-468.

[Pre94] Prechelt L. A study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice. Technical Report University of Karlsruhe, 1994.

[Qui93] Quinlan JR. C4.5: Programs for Machine Learning. Los Altos (CA): Morgan Kauffman, 1993.

[Qui87] Quinlan JR. Simplifying decision trees. International Journal of Man-Machine Studies 1987;27:221-234.

[Rac] Rachlin J, Kasif S, Salzberg S, Aha DW. Towards a Better Understanding of Memory-Based Reasoning Systems. Unpublished.

[Rea88] Read TRC, Cressie NAC, Goodness-of-Fit Statistics for Discrete Multivariate Data, Springer-Verlag New York Inc, 1988.

[Rip94] Ripley BD. Neural Networks and Related Methods for Classification. J.R. Statistics Soc. B 1994; 56(3): 409-456

[Rou87] Rousseeuw PJ, Leroy AM. Robust Regression and Outlier Detection, New York; John Wiley and Sons, Inc 1987.

[Sha72] Shanmugam K. A Parametric Procedure for Learning with an Imperfect Teacher. IEEE Trans on Information Theory 1972; IT-18:300-302.

[Shi99] Shi Y, Eberhart R, Chen Y. Implementation of Evolutionary Fuzzy Systems. IEEE Transactions on Fuzzy Systems 1999;7(2).

[Sos98] Sosnowski Z. Private communication.

[Sug93] Sugeno M, Yasukawa T. A Fuzzy-Logic-Based Approach to Qualitative Modelling. IEEE Trans Fuzzy Systems 1993;1(1):7-28.

[Tit89] Titterington D. M. An Alternative Stochastic Supervisor in Discriminant Analysis. Pattern Recognition 1989;22(1):91-95.

[Wes] Westmore D. RDSS IMT Technical Report

[Win92] Winston PH. Artificial Intelligence, Third Edition; Addison-Wesley, 1992.

[Yag93] Yager RR. On the completion of Qualitative Possibility Measures. IEEE Trans on Fuzzy Systems 1993;1(3).

[Yee93] Yee P, Haykin S. Pattern Classification as an Ill-posed, Inverse

Problem: A regularisation Approach. To appear ProcICASSP-93.

[You86] Young TY, Fu K. Handbook of Pattern recognition and Image Processing. Academic Press, 1986.

[Zad73] Zadeh L. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. IEEE Trans. Systems, Man, and Cybernetics 1973; SMC-3: 28-44.

[Zhu90] Zhu Q. On the Minimum Probability of Error of Classification with Incomplete Patterns. Pattern Recognition 1990; 23(11):1281-1290.

[Zhu92a] Zhuang X, Wang T, Zhang P. A highly robust estimator through partially likelihood function modelling and its application in computer vision. IEEE Trans Pattern Anal. Machine Intell 1992; 14:19-35.