

A Study on Diversity in Classifier Ensembles

Catherine A. Shipp

December 9, 2004

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.....(candidate)

Date.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed.....(candidate)

Date.....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed.....(candidate)

Date.....

Summary of Research

In this thesis we carry out a series of investigations into the relationship between diversity and combination methods and diversity and AdaBoost.

In our first investigation we study the relationships between nine combination methods. Two data sets are used. We consider the overall accuracies of the combination methods, their improvement over the single best classifier, and the correlation between the ensemble outputs using the different combination methods.

Next we introduce ten diversity measures. Using the same two data sets, we study the relationships between the diversity measures. Then we look at their relationship to the combination methods previously studied. The ranges of the ten diversity measures for three classifiers are derived. They are compared with the theoretical ranges and their implications for the accuracy of the ensemble are studied.

We then proceed to investigate the diversity of classifier ensembles built using the AdaBoost algorithm. We carry out experiments with two datasets using ten-fold cross validation. We build 100 classifiers each time using linear classifiers, quadratic classifiers or neural networks. We study how diversity varies as the classifier ensemble grows and how the different types of classifier compare.

Next we consider ways of improving AdaBoost's performance. We conduct an investigation into how modifying the size of the training sets and the complexity of the individual classifiers alter the ensemble's performance. We carry out experiments using three datasets.

Lastly we consider using pareto optimality to determine which classifiers built by AdaBoost to add to the ensemble. We carry out experiments with ten datasets. We compare standard AdaBoost to AdaBoost with two versions of the Pareto-optimality method called Pareto 5 and Pareto 10, to see whether we can reduce the ensemble size without harming the ensemble accuracy.

Acknowledgements

I do of course need to thank my supervisor, Lucy Kuncheva, profusely. Without your constant support, encouragement and motivation I would never have come close to finishing this work. Your insistence on my writing and submitting papers and attending and presenting them at conferences has meant that I have had to grow in confidence despite myself.

I would also like to thank Chris Whitaker who, along with Lucy I have had the pleasure of collaborating with on other research. You, and our other colleagues, who attended discussions and brain-storming sessions with regards to this thesis were always able to bring a fresh eye and a new angle to my work.

On a personal level, I would like to thank my family and friends for putting up with my moods over the past three years. Now I have finally finished my work I will no longer have an excuse and will have to try to behave better to all of you.

Finally, I would like to thank my husband Mathew for never complaining if I stayed late or was in a bad mood after fighting the computer all day and for nagging me until I finally finished my corrections.

Thank you with love.

Catherine Allison Shipp,
December 2004

Contents

Notions, Notations and Abbreviations	xvii
1 Introduction	1
1.1 Background	1
1.2 Classifiers	2
1.3 Classifier Design	3
1.3.1 Bayes	3
1.3.2 Parametric Classifiers	4
1.3.3 Linear Discriminant Classifiers	5
1.3.4 Quadratic Discriminant Classifiers	5
1.3.5 Neural Networks	5
1.3.6 Tree Classifiers	7
1.4 Aims	9
1.5 Organisation of thesis	10
2 Combination Methods	11
2.1 Why combine classifiers?	11
2.2 How multiple classifier systems work	13
2.3 Strategies for building classifier ensembles	13
2.4 Combination methods	16
2.4.1 Classifier Fusion	16
2.4.2 Simple combination methods	17
2.4.3 Voting Methods	18
2.4.4 Majority vote	19
2.4.5 Weighted voting	20
2.4.6 Limits of majority vote	21
2.4.7 Naive Bayes	22
2.4.8 Behavior-Knowledge Space and Wernecke's method	23
2.4.9 Decision Templates	25
2.4.10 The Oracle	26

2.4.11	Data-dependent weights and multi-level classifiers	26
2.5	Existing empirical studies of combination methods	27
2.6	Experimental set-up	29
2.7	Combination Method Results	31
2.7.1	Overall Accuracies	31
2.7.2	Relationships among the combination methods	34
2.8	Combination Methods Conclusions	37
3	Diversity in Classifier ensembles	39
3.1	Measures of diversity	40
3.1.1	Pairwise Diversity Measures	41
3.1.2	Non-pairwise Diversity Measures	44
3.2	Limits of the measures	49
3.2.1	The case of Identical Classifiers	49
3.2.2	The case of Highly Diverse Classifiers	51
3.2.3	Examining the limits	57
3.3	Existing empirical studies of Diversity	59
3.4	Experimental set-up for diversity measures	60
3.5	Diversity Measures Results	61
3.5.1	Overall Diversities	61
3.5.2	Relationships among the diversity measures	62
3.5.3	Relationship with accuracy	65
3.5.4	Relationships between the combination methods and the diversity measures	68
3.6	Diversity Measures Conclusions	70
4	Ensemble Construction Methods	73
4.1	Bias and Variance	73
4.2	Bagging	74
4.3	Arcing and Boosting	75
4.4	Which method to use?	76
4.5	AdaBoost	77
4.5.1	The AdaBoost algorithm	77
4.5.2	Optimality of the combiner for AdaBoost	79
4.5.3	Boosting the margins	81
4.6	Existing empirical studies about AdaBoost	82
4.6.1	AdaBoost and modifications	82
4.6.2	Considering margins directly	86

4.6.3	The multi-class case	86
4.6.4	Comparing with other methods	87
4.6.5	Different combination methods	90
4.6.6	Overproduce and choose	90
4.6.7	Summary of Existing Studies of AdaBoost	91
4.7	Experimental Set-Up	92
4.8	AdaBoost and Classifier Diversity Results	94
4.9	AdaBoost and Classifier Diversity Conclusions	99
5	Improving AdaBoost	101
5.1	Modifying AdaBoost	102
5.1.1	Experimental set-up	103
5.1.2	Training Errors	104
5.1.3	Varying the Sample Size	105
5.1.4	Modifying the Number of Neurons Used	108
5.1.5	Varying Both Sample Size and Number of Neurons	108
5.2	Kappa-error diagrams and Pareto-optimal sets	112
5.2.1	Kappa for class label outputs	112
5.2.2	Kappa-error Diagrams	113
5.2.3	Pareto-Optimal Sets	115
5.3	AdaBoost with Pareto Optimality	116
5.3.1	Experimental set-up to investigate AdaBoost with Pareto Optimality	118
5.3.2	AdaBoost with Pareto Optimality results	118
5.4	Improving AdaBoost Conclusions	121
6	Conclusions	123
6.1	Main Investigations and Findings of this Thesis	123
6.2	Limitations of the Thesis	125
6.3	Summary of My Contributions	125
6.4	Possible Future Considerations	127
6.5	My References	127
A	Proof of Equivalence Relationships	129
A.1	Proof that Max is equivalent to Min for two classes	129
A.2	Proof that KW, Ent and D are equivalent for 3 classifiers	129
B	Data Sets Used in this Thesis	131
B.1	The Sonar Identification dataset	132
B.2	The Glass Identification dataset	132

B.3	The Haberman dataset	133
B.4	The Ecoli dataset	133
B.5	The Liver dataset	133
B.6	The Johns Hopkins University Ionosphere dataset	134
B.7	The 1984 United States Congressional Voting Records dataset	134
B.8	The Wisconsin Breast Cancer dataset	135
B.9	The Pima Indian Diabetes dataset	136
B.10	The Vehicle Silhouette Identification dataset	136
B.11	The German Credit dataset	138
B.12	The Phoneme dataset	138

List of Figures

1.1	Schematic of how a classifier works	2
1.2	Schematic of an artificial neuron	6
1.3	A feed-forward neural network	7
1.4	A decision tree classifier	8
2.1	A multiple classifier system	14
2.2	What can we change in a multiple classifier system?	15
2.3	Testing set accuracy for the individual classifiers and the ensemble	32
2.4	Improvement on the testing set for the individual classifiers and the ensemble	33
2.5	Illustration of the correlation between the combination methods	35
2.6	The cluster dendrograms for the combination methods	36
3.1	Comparing the two entropy measures	47
3.2	Example probability mass function for the measure of difficulty	48
3.3	Probability mass function for identical classifiers	50
3.4	The most diverse classifiers for $p \leq \frac{2}{3}$	52
3.5	The most diverse classifiers for $p \geq \frac{2}{3}$	53
3.6	Possible range of values for the five measures of diversity	59
3.7	Possible range of values for the five measures of similarity	59
3.8	Overall correlation between the diversity measures	64
3.9	Cluster dendrograms for the diversity measures	66
3.10	Correlation between the combination methods and the diversity measures	69
4.1	Resampling implementation of the AdaBoost algorithm	78
4.2	Various aspects of, and studies into, AdaBoost	83
4.3	Change in the average error and value of Q as we add linear classifiers to the ensemble	95
4.4	Change in the average error and value of Q as we add quadratic classifiers to the ensemble	96
4.5	Change in the average error and value of Q as we add neural networks to the ensemble	97
4.6	Average error versus average training Q	98

5.1	The training error for 15 neurons and sample size N	104
5.2	The effect on the testing error of varying the sample size	106
5.3	The effect on the testing error of varying the number of neurons in the hidden layer	107
5.4	The effect on the testing error of randomly varying the sample size and/or the number of neurons	109
5.5	Example of calculating $Kappa_E$	114
5.6	Example of a Kappa-error plot showing the convex hull and the Pareto-optimal set	115
5.7	Pareto-optimality	116
5.8	The AdaBoost algorithm with Pareto-optimal sets	117
5.9	Comparing percentage change in error and average number of runs with the two Pareto versions	120

List of Tables

2.1	Classifier fusion algorithms	17
2.2	How the simple combination methods work	18
2.3	Accuracies on combining with majority vote	22
2.4	The confusion matrices of three classifiers	23
2.5	How the behavior-knowledge space table works	24
2.6	Summary of datasets and the experiments	30
2.7	The correlation coefficients between the different combination methods	34
3.1	The 2×2 relationship table with probabilities	41
3.2	The 2×2 relationship tables for identical, independent and negatively dependent classifiers	41
3.3	The 2×2 relationship table for slightly negative dependent classifiers	42
3.4	An example of classifiers for the non-pairwise diversity measures	45
3.5	Limits for the diversity measures in terms of p	58
3.6	Examining the diversity values found in the example	58
3.7	Observed range and theoretical range of diversity measures for the breast cancer data	62
3.8	Observed range and theoretical range of diversity measures for the Pima data . . .	63
3.9	Correlation coefficients for the relationships between the combination methods and the diversity measures	70
4.1	Summary of the data sets	93
4.2	Testing error using various stopping criteria	99
5.1	Data used in the experiments	103
5.2	BEM procedure comparing the AdaBoost implementations	110
5.3	Successes for different bootstrap sample sizes	111
5.4	Successes for different numbers of hidden neurons	111
5.5	Summary of the data sets	118
5.6	Average error for AdaBoost, Pareto 5 and Pareto 10	119

A.1	Possible values for Ent and kw	130
-----	--	-----

Notions, Notations and Abbreviations

For convenience the commonly used notations and abbreviations are placed here for easy referral.

General General notations

- $\Omega = \{\omega_1, \dots, \omega_c\}$ - The set of Class labels
- c - the number of class labels
- \mathfrak{R}^n the feature space
- $X = \{X_1, \dots, X_n\}$ - the set of feature labels
- n - the number of features
- $\mathbf{x} = [x_1, \dots, x_n]^T$, or $\mathbf{x} \in \mathfrak{R}^n$ - the feature vector describing object \mathbf{x}
- y_i - the class label of object \mathbf{x}_i
- $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$, $\mathbf{z}_j \in \mathfrak{R}^n$ - the training data set
- $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ - the labelled training data
- N - the number of training examples
- $D : \mathfrak{R}^n \rightarrow \Omega \forall \mathbf{x} \in \mathfrak{R}^n$ s.t. $D(\mathbf{x}) \in \Omega$ - a classifier
- $p = \frac{N_c}{N}$ -the apparent accuracy of a classifier where N_c is the number of correctly classified elements of Z
- $g_i(\mathbf{x})$ - the discriminant function

Multiple Classifier Systems Additional notations for MCS

- $\mathcal{D} = \{D_1, \dots, D_L\}$ - a set of classifiers
- $D_i(\mathbf{x}) = [d_{i,1}(\mathbf{x}), \dots, d_{i,c}(\mathbf{x})]^T$ -classifier outputs
- $d_{i,j}(\mathbf{x})$ - the degree of support given by classifier D_i to hypothesis that \mathbf{x} comes from class ω_j

- $\mathcal{D}(\mathbf{x}) = [\mu_1(\mathbf{x}), \dots, \mu_c(\mathbf{x})]^T$ - the combined output of the classifiers for object \mathbf{x}
- $\mu_j(\mathbf{x})$ - the combined support to the hypothesis that \mathbf{x} comes from class ω_j
- $\tilde{\mathcal{D}}(\mathbf{x})$ - the aggregated classification decision

Combination Abbreviations Abbreviations used for different combination methods

- *MAX*- the maximum
- *MIN*- the minimum
- *AVR*- the average
- *PRO*- the product
- *MAJ*- the majority vote
- *NB*- the naive Bayes
- *BKS*- the behavior-knowledge space
- *WER*- Wernecke's method
- *DT*- decision templates
- *ORA*- the oracle

Diversity Abbreviations Abbreviations used when dealing with diversity

- *Q*- the *Q*-statistic
- ρ - the correlation coefficient
- *D*- the disagreement measure
- *DF*- the double-fault measure
- *kw*- the Kohavi-Wolpert variance
- κ - the measure of interrater agreement
- *Ent*- the entropy measure
- θ - the measure of difficulty
- *GD*- the generalised diversity
- *CFD*- the coincident failure diversity

AdaBoost notations Notations used with the AdaBoost algorithm

- K_{\max} - the number of iterations required
- k - the current iteration
- S_k - the training set for iteration k
- D_k - the classifier trained on iteration k

- $w_k(i)$ - the weight for object i at iteration k
- $\mathbf{W}_k = \{w_k(1), \dots, w_k(N)\}$ - the set of weights at iteration k
- β_k - the combination weight for classifier D_k
- BEM- Bechofer, Elmaghraby and Morse's procedure for comparing competing pattern recognition algorithms used in [2]
- κ_E - the kappa-statistic used by Margineantu and Dietterich for their κ -error diagrams [74]

Chapter 1

Introduction

1.1 Background

Pattern recognition is concerned with the process of assigning *objects* to *classes*. Its applications are connected with the fields of mathematics, engineering, information technology and computer science.

For each problem tackled using pattern recognition methods we have

- The set of c **class labels** consists of all possible mutually exclusive classes denoted $\Omega = \{\omega_1, \dots, \omega_c\}$.
- The **features** of an object are characteristics that can be expressed in numerical form, e.g. height, pressure, number of vertical strokes in an image, grey level intensity of a pixel, etc.
- The n **feature values** are the particular values of the features for a specific object denoted by the vector $\mathbf{x} = [x_1, \dots, x_n]^T$, or $\mathbf{x} \in \Re^n$. The feature values can be continuous, binary or categorical in nature.
- The **feature labels** are the labels for each of the n features denoted $X = \{X_1, \dots, X_n\}$.
- The **feature space** is the space, consisting of all possible values of our features denoted \Re^n .
- The **training data set** is a set of objects described by their feature values and is denoted $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$, $\mathbf{z}_j \in \Re^n$. The N objects are usually labelled in the c classes so that $\mathbf{z}_i = (\mathbf{x}_i, y_i)$, $y_i \in \Omega$.

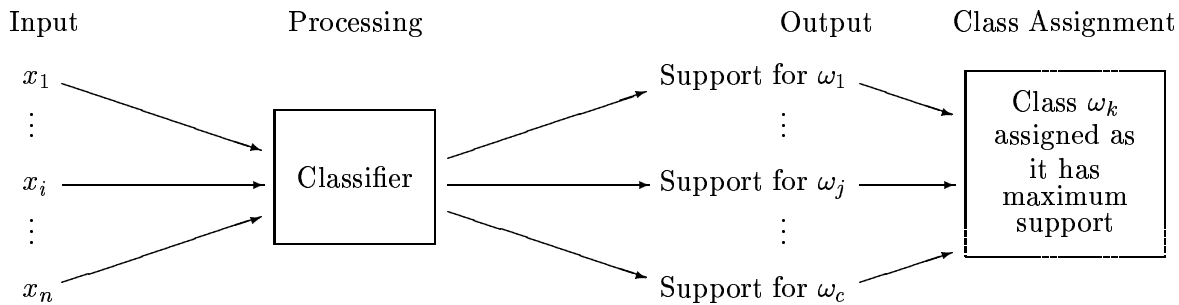


Figure 1.1: SCHEMATIC OF HOW A CLASSIFIER WORKS

1.2 Classifiers

A classifier is any mapping D which assigns a class label to an object, i.e.,

$$D : \mathbb{R}^n \rightarrow \Omega, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad D(\mathbf{x}) \in \Omega. \quad (1.1)$$

Classifiers usually output support for each class label being the correct one for object \mathbf{x} and the label with the most support is assigned to \mathbf{x} . Classifiers can be designed in different ways, and therefore range in their ability to accurately assign a class to an object. The choice of type of classifier can therefore have a big impact on the accuracy of the classification. Figure 1.1 shows a schematic of how a generic classifier works.

Accuracy of Classifiers

Classification accuracy is a major characteristic of a classifier. The so called “apparent” accuracy of classifier D is obtained by running D on the data set Z and calculating

$$p = \frac{N_c}{N} \quad (1.2)$$

where N_c is the number of correctly classified elements of Z .

Example - diagnosing a patient with respiratory problems

This is an analogy of how, in a medical setting, a classifier can work like a doctor in diagnosing a patient. The medical details are taken from [43]. A patient (A) presents with respiratory problems and several measurements are taken. The appropriate measurements (features), healthy person’s values and Patient A’s values are shown in the following table.

Features	Normal Values (healthy patient)	Patient A's values
Temperature ($^{\circ}C$)	37	38.2
Respiratory Rate (Breaths per min)	18	25
Blood pressure (mmHg)	$< \frac{140}{90}$	$\frac{125}{90}$
Heart rate (beats per min)	50 to 100	110
pH of arterial blood (AB)	7.35 to 7.45	7.3
Partial pressure of oxygen in AB (Kpa)	11 to 14	7
Partial pressure of carbon dioxide in AB (Kpa)	4.7 to 6.0	9
Bicarbonate ion concentration (mmol/l)	22 to 26	33
Oxygen saturation of arterial haemoglobin (%)	95 to 98%	85
secretion colour	clear	pale green
secretion cultures (bacteria presence)	none	bacteria present

The choice of possible diagnoses are the class labels,

Class labels:

$$\Omega = \{\text{Emphysema, Chronic Bronchitis, Asthma, Pneumonia}\}$$

The problem is to diagnose the respiratory problem patient A has i.e. to assign a class label from Ω to \mathbf{x}_a . For this example “Chronic Bronchitis” is the most likely diagnosis. We will refer to this example as an analogy for further concepts throughout the thesis.

1.3 Classifier Design

1.3.1 Bayes

When we wish to classify an object, ω_i we need to find the posterior probabilities for each class, i.e., given the object \mathbf{x} we need to know the probability of it belonging to each class. The posterior probabilities for each class $P(\omega_i|\mathbf{x})$ are found by using the Bayes formula which is given by:

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{\sum_{j=1}^c P(\omega_j)p(\mathbf{x}|\omega_j)} \quad (1.3)$$

where $P(\omega_j)$, $j = 1, \dots, c$ are the prior probabilities for each class, and $p(\mathbf{x}|\omega_j)$ are the class-conditional probabilities.

The most natural way of classifying an object is to consider all of the posterior probabilities and then to assign the class label which has the highest value of posterior probability. When used in this way we call the posterior probabilities a set of *discriminant functions* (also known as decision functions or classification functions), and denote them as, g_i , where

$$g_i : \mathbb{R}^n \rightarrow \mathbb{R}. \quad (1.4)$$

Discriminant functions are not unique, in fact any set of functions, $f(g_i(\mathbf{x}))$, where f is monotonically increasing gives a practically equivalent set of discriminant functions, i.e., they will make the same classification decisions. For example, if we consider the Bayes formula, note that the denominator will be identical regardless of the class ω_i , and so we can ignore it, therefore getting another set of Bayes-optimal discriminant functions:

$$g_i(\mathbf{x}) = P(\omega_i)p(\mathbf{x}|\omega_i) \quad (1.5)$$

or we can take logarithms and obtain:

$$g_i(\mathbf{x}) = \log(P(\omega_i)) + \log(p(\mathbf{x}|\omega_i)) \quad (1.6)$$

The discriminant functions we use depend on the problem at hand and the information available.

1.3.2 Parametric Classifiers

Parametric classifiers are based on *estimating* the parameters of the class-conditional probability density functions (p.d.f.'s), $p(\mathbf{x}|\omega_i)$, from which we obtain posterior probabilities as shown in equation (1.3) [53].

Since we usually do not have the required information, we can use estimates of these posterior probabilities as a set of discriminant functions to classify object \mathbf{x} .

We can estimate the prior probabilities $P(\omega_i)$ as the proportion of elements from the training set, Z , which come from class ω_i . The parametric classifiers assume a hypothetical form of the class-conditional p.d.f's, $p(\mathbf{x}|\omega_i)$, and it is only the parameters of the p.d.f's which must be estimated. If we assume that the classes are normally distributed, such that $p(\mathbf{x}|\omega_i) \sim N(\mathbf{m}_i, S_i)$ where $\mathbf{m}_i \in \mathbb{R}^n$ is the mean vector for class ω_i and S_i is the covariance matrix then

$$p(\mathbf{x}|\omega_i) = \frac{1}{\sqrt{(2\pi)^n |S_i|}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T S_i^{-1} (\mathbf{x} - \mathbf{m}_i) \right\} \quad (1.7)$$

Substituting this into 1.6 we get,

$$\begin{aligned} g_i(\mathbf{x}) &= \log(P(\omega_i)) + \log(p(\mathbf{x}|\omega_i)) \\ &= \log(P(\omega_i)) - \frac{n}{2} \log(2\pi) - \frac{1}{2} \log(|S_i|) - \frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T S_i^{-1} (\mathbf{x} - \mathbf{m}_i) \end{aligned} \quad (1.8)$$

The parameters \mathbf{m}_i and S_i are estimated from the training set Z as:

$$\mathbf{m}_i = \frac{1}{N_i} \sum_{\mathbf{z}_j, y_j = \omega_i} \mathbf{z}_j \quad (1.9)$$

where $y_j \in \Omega$ is the class label of \mathbf{z}_j , N_i is the number of elements of Z from class ω_i and

$$S_i = \frac{1}{N_i - 1} \sum_{\mathbf{z}_j, y_j = \omega_i} (\mathbf{z}_j - \mathbf{m}_i)(\mathbf{z}_j - \mathbf{m}_i)^T. \quad (1.10)$$

1.3.3 Linear Discriminant Classifiers

For linear discriminant classifiers we take a linear form of equation (1.8). We assume that the p.d.f is normally distributed with the classes having different means but the same covariance matrix, $p(\mathbf{x}|\omega_i) \sim N(\mathbf{m}_i, S)$. By substituting S for S_i and discarding all terms that are not dependent on ω_i in equation (1.8) we obtain:

$$\begin{aligned}
 g_i(\mathbf{x}) &= \log(P(\omega_i)) - \frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T S^{-1}(\mathbf{x} - \mathbf{m}_i) \\
 &= \log(P(\omega_i)) - \frac{1}{2} [\mathbf{x}^T S^{-1} \mathbf{x} + \mathbf{m}_i^T S^{-1} \mathbf{m}_i - \mathbf{x}^T S^{-1} \mathbf{m}_i - \mathbf{m}_i^T S^{-1} \mathbf{x}] \\
 &= \log(P(\omega_i)) - \frac{1}{2} \mathbf{m}_i^T S^{-1} \mathbf{m}_i + \frac{1}{2} \mathbf{x}^T S^{-1} \mathbf{m}_i + \frac{1}{2} \mathbf{m}_i^T S^{-1} \mathbf{x} \\
 &= \log(P(\omega_i)) - \frac{1}{2} \mathbf{m}_i^T S^{-1} \mathbf{m}_i + \mathbf{m}_i^T S^{-1} \mathbf{x},
 \end{aligned}$$

which can be written as:

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{W}_i^T \mathbf{x} \quad (1.11)$$

where $w_{i0} \in \Re$ and $\mathbf{W}_i \in \Re^n$ are coefficients.

1.3.4 Quadratic Discriminant Classifiers

Quadratic discriminant classifiers are obtained by discarding all terms independent of ω_i from equation (1.8).

$$\begin{aligned}
 g_i(\mathbf{x}) &= \log(P(\omega_i)) - \frac{1}{2} \log(|S_i|) - \frac{1}{2} [\mathbf{x}^T S^{-1} \mathbf{x} + \mathbf{m}_i^T S^{-1} \mathbf{m}_i - \mathbf{x}^T S^{-1} \mathbf{m}_i - \mathbf{m}_i^T S^{-1} \mathbf{x}] \\
 &= \log(P(\omega_i)) - \frac{1}{2} \log(|S_i|) - \frac{1}{2} \mathbf{m}_i^T S^{-1} \mathbf{m}_i + \mathbf{m}_i^T S^{-1} \mathbf{x} - \frac{1}{2} \mathbf{x}^T S^{-1} \mathbf{x},
 \end{aligned}$$

which can be written as:

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{W}_i^T \mathbf{x} + \mathbf{x}^T W_i \mathbf{x} \quad (1.12)$$

where $w_{i0} \in \Re$ and $\mathbf{W}_i \in \Re^n$ are coefficients and \mathbf{W}_i is an $n \times n$ matrix.

The parameters for the linear and quadratic classifiers are the \mathbf{m}_i and S_i which are estimated as shown in equations(1.9) and (1.10) respectively.

1.3.5 Neural Networks

Neural networks can be thought of as a *trained black box* where the features of an object are given as input. They are then processed in some way resulting in a set of c discriminant functions given as output [53]. The idea behind neural networks was to model the *function* of the human brain by using the biological structures used in the brain. This initial idea has not progressed much further than a simplified modelling of a single neuron. However, more mathematical neural networks with less emphasis on the biological structures are a widely used tool in classification.

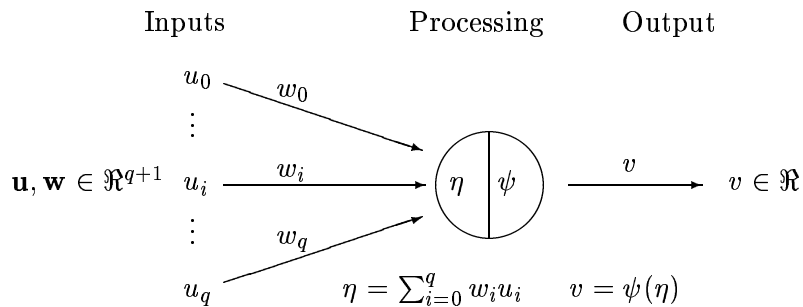


Figure 1.2: SCHEMATIC OF AN ARTIFICIAL NEURON [53]

Modelled neurons act as the processing unit in neural networks, they are often called nodes to prevent confusion with their biological versions. A neuron in the brain receives electrical impulses as its input and if the impulses reach a certain *activation* level the neuron fires and sends impulses onwards as outputs. Nodes take the input values and multiplies them by a vector of **synaptic weights**, these values are then combined and submitted to an **activation function**. The value obtained from the activation function is then given as the output. Figure 1.2 illustrates how a node operates [53]. $\mathbf{u} = [u_0, \dots, u_q]^T \in \mathbb{R}^{q+1}$ is the input vector given to the node, $\mathbf{w} = [w_0, \dots, w_q]^T \in \mathbb{R}^{q+1}$ is the vector of synaptic weights, $\psi : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function and $v = \psi(\eta)$, $\eta = \sum_{i=0}^q w_i u_i$ is the output from the node. There are various choices of activation function available, some of the more common ones are [53]:

- The threshold function

$$\psi(\eta) = \begin{cases} 1, & \text{if } \eta \geq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (1.13)$$

- The sigmoid function

$$\psi(\eta) = \frac{1}{1 + \exp(-\eta)}. \quad (1.14)$$

- The identity function

$$\psi(\eta) = \eta. \quad (1.15)$$

Of these the sigmoid function is most widely used as it can approximate linear and threshold functions and is easily differentiable which is necessary for neural network training algorithms [53].

In a neural network we can have many nodes where the output from the activation functions of some nodes act as inputs to other nodes. We may have several layers of nodes; the input layer, several hidden layers and an output layer. A structure of this type is the *multi-layer perceptron* (MLP). The input layer nodes take the feature values as input and the output layer nodes produce the set of c discriminant functions as output. MLP is

feed-forward in nature, because the hidden layers will take values from previous nodes as input and output values to the next set of nodes with no feedback allowed. Figure 1.3 illustrates a feed-forward neural network of this type [53]. The information can only pass up the neural network as it is a feed-forward neural network. The box surrounds the *black box* aspect of the classifier. This is one form of classifier which may be used in the place of the classifier indicated in Figure 1.1.

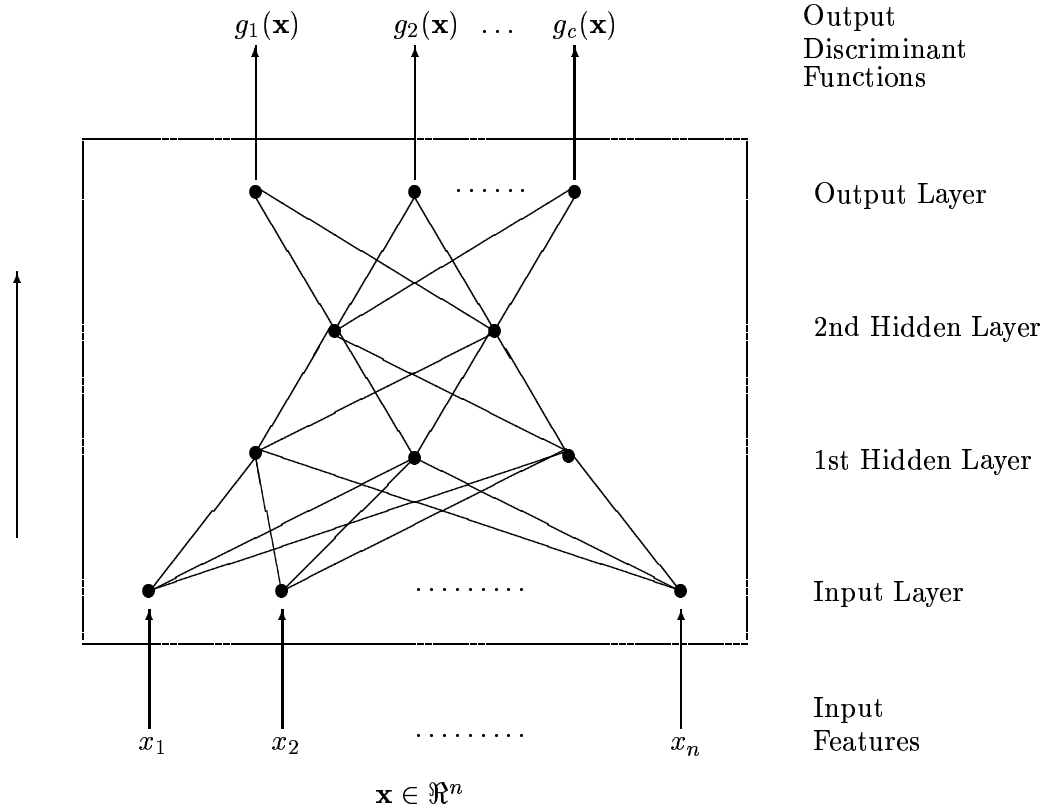
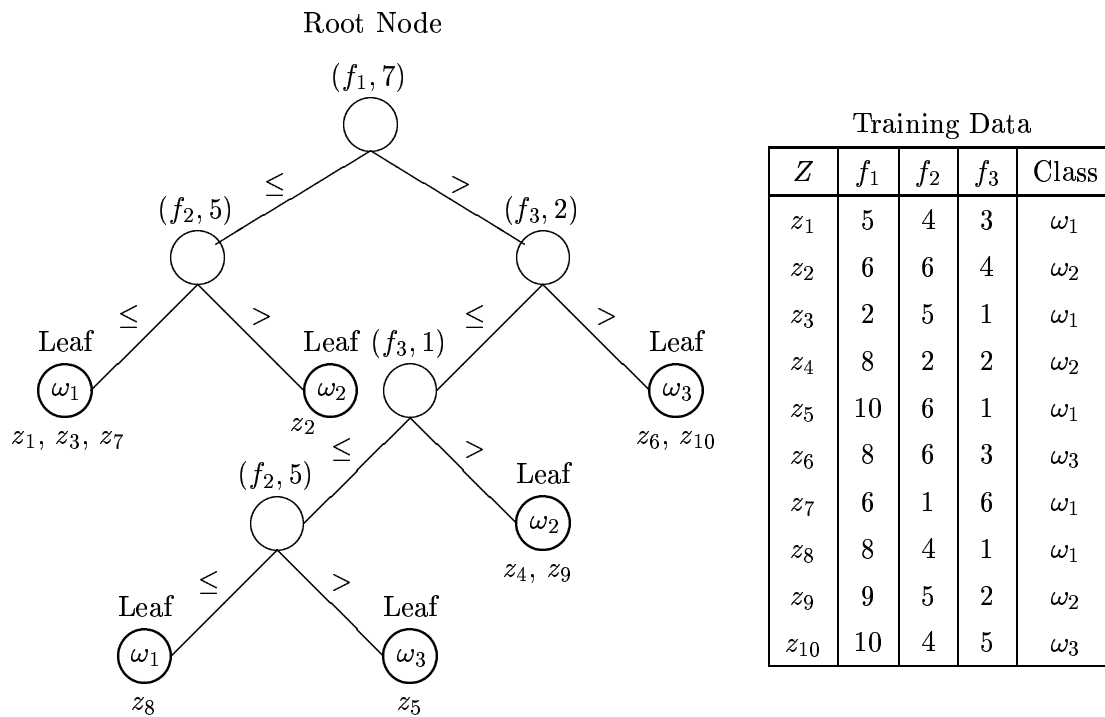


Figure 1.3: A SCHEMATIC OF A FEED-FORWARD NEURAL NETWORK WITH TWO HIDDEN LAYERS [53]

1.3.6 Tree Classifiers

Similarly to neural networks, trees consist of a series of nodes. Each node will consider a single feature from the input data. A single root node will be connected by branches to a set of nodes. These nodes are linked to more nodes in the next layer further down the tree until a terminal node is reached which is often called a leaf node [22]. The most commonly used decision tree classifiers are binary in nature, using a single feature at each node. This results in decision boundaries which are parallel to the feature axes [45]. Each node considers whether the feature is lesser or greater than a critical value. If it is less

we may follow the left branch, for example, and if it is greater we may follow the right branch. We follow the route down the branches until we reach a leaf where we make a classification. Each leaf is normally assigned a specific class label, determined by the training examples [46]. The tree is constructed with continual subdividing until all of the training examples in a node are of the same class and that is then determined to be a leaf node. The class label of the training examples in the leaf node is then assigned as the class label of the leaf. Figure 1.4 shows how a decision tree classifier works. Decision stumps are the simplest type of decision tree as they are classification trees with only one split at the root node which partitions the data into two disjoint classification regions [19]. Obviously the critical value is chosen to most separate the classes. Since stumps are so simple it is often possible to use an exhaustive search method to identify the best critical value, which would not be possible with more complex classifiers.




where (f_i, m) denotes using critical value m for feature f_i ,
 denotes the leaf nodes, ω_i the class assigned by that leaf
 and z_i beneath a node denotes which leaf object z_i ends up in.

Figure 1.4: A SCHEMATIC OF A DECISION TREE CLASSIFIER

The difficulty in constructing a decision tree is to decide which is the ‘best’ feature to use at each node and which is the ‘best’ critical value for that particular feature [22]. Obviously the example tree shown in Figure 1.4 is by no means optimal, and indeed binary trees are intrinsically suboptimal for most applications [45]. However, they make up for

this disadvantage in two major ways:

1. Trees are considerably faster than other classifiers.
2. We can interpret the decision rules used to classify an object in terms of the individual features.

These advantages make them a popular choice for users, as does the public availability of several decision tree classification systems such as Breiman et al.'s CART¹ [13] and Quinlan's C4.5 [84] which are often used as benchmarks to compare new classifiers to.

Like neural networks, decision trees can suffer from over-training. We can simplify the decision tree and improve the generalisation ability by *pruning* the tree [22]. This is done using a separate validation data set with the same statistics as the original training data. We run the validation set through the tree and calculate the pruning error rate. We then consider each node above a leaf node in turn. If the pruning error rate would be improved by turning the node above a leaf node into a terminal node (and removing the leaf nodes below it) then we do this, known as *pruning* the tree. This process is repeated until we cannot improve the pruning error any more.

The decision tree is another type of classifier which can be used in the place of the classifier indicated in Figure 1.1.

When we have various different classifier types available, it is often difficult to know which is the best to use for a given application. Often in studies the test errors of the various classifiers are given and it is left to the individual to interpret and try to determine whether they are significantly different. Clearly, it would be better to be able to determine if there is a statistical difference between classifiers' errors. Then the most accurate classifier could be used, or if they are all statistically similar, the most easily implemented could be chosen. Looney for instance, gives a statistical basis for comparing L classifiers with respect to their individual accuracy [72].

Another approach is to try to use the information provided by several classifiers and combine it in some way. The rest of this thesis is concerned with multiple classifier systems and combining ensembles of classifiers.

1.4 Aims

The main aims of this thesis are:

- To compare the accuracies of some of the more commonly used classifier combination methods to each other and to the single best classifier using an ensemble of three classifiers.

¹<http://www.mlnet.org/> and follow the software link.

- To examine the Pearson's product moment correlation between the outputs from these classifier combination methods and to run a clustering program on the combination methods. This is in order to see whether we can identify if any of the combination methods perform similarly or quite differently from each other.
- To examine the theoretical limits of ten measures which measure diversity amongst classifiers and to compare these limits to the actual levels of diversity attained using experiments on real-world data.
- To examine the Pearson's product moment correlation between the diversity values obtained from these diversity measures and to run a clustering program on the diversity measures. This is in order to see whether we can identify whether any of the diversity measures are measuring the same aspect of 'diversity' or whether they are all measuring different things.
- To examine how the AdaBoost ensemble construction method affects the diversity of the ensemble of classifiers it builds and whether this diversity is related to the generalisation error of the ensemble on combination.
- To examine how modifying the sample size of training data, the number of neurons used or both, affects the generalisation error of AdaBoost.
- To investigate whether or not using Pareto optimal sets can produce considerably smaller ensembles of classifiers without significantly increasing the generalisation error when used with AdaBoost.

1.5 Organisation of thesis

Chapter 2 introduces various combination methods and studies their relationships with one another.

Chapter 3 introduces various diversity measures and studies their relationships with one another and their relationship with the combination methods introduced in Chapter 2.

Chapter 4 considers different approaches to improving the performance of ensembles of classifiers and in particular introduces the AdaBoost algorithm.

Chapter 5 considers varying the sample size and varying the number of neurons with AdaBoost. It also considers how we can use Pareto Optimality in conjunction with AdaBoost.

Chapter 6 gives the overall conclusions and future considerations.

Chapter 2

Combination Methods

2.1 Why combine classifiers?

Combining classifiers is an established research area in the fields of statistical pattern recognition and machine learning to develop highly accurate classification systems [3, 4, 15, 37, 38, 44–46, 48, 51, 66, 67, 71, 91, 109, 111, 115, 116]. It is variously known as committees of learners, mixtures of experts, classifier ensembles, multiple classifier systems, consensus theory etc. This approach has been developed because a highly accurate and reliable classification is required for practical applications.

Classifiers with different data sources, architectures, algorithms or trained on different feature subsets can exhibit complementary classification behaviour. If we have many different classifiers at our disposal, it is sensible to consider using them in some form of combination in the hope of increasing both reliability [66] and the overall accuracy [45]. As described by Battiti and Colla [4] we can therefore use our classifiers as a team similarly to the way that a person may consult a panel of experts before making a decision. Each of the classifiers obtained could attain a different accuracy, but it is unlikely that any will be 100% accurate, and they may not even be as good as expected. Thus, there is the need to integrate the results from a number of different classifiers in order to obtain an improved result [116]. If we recall the analogy of a respiratory patient (see 1.2) it would be preferable to the patient to have several doctors' opinions. This is the intuition behind multiple classifier systems.

Some researchers have found that even trying many algorithms it is difficult to improve accuracy beyond a certain point using a single classifier. Therefore, in order to progress further, multiple classifiers have to be utilised. It has been reported by Lam and Suen that a combination of classifiers results in 'a remarkable improvement in recognition results' [67]. It has also been proved theoretically that a group of independent classifiers improve upon the single best classifier when majority vote combination is used (see 2.4.4) [67, 116]. It is

generally assumed that the improvement holds for other combination methods as well.

The classifiers in an ensemble must be accurate enough to contribute information but also different enough from each other to ensure the information is beneficial. Obviously combining an infinite set of identical classifiers with accuracy 90% is not going to obtain an ensemble accuracy any better than 90%! However, combining several classifiers with accuracies of 75% can produce an ensemble accuracy of more than 75%, maybe even higher than 90%, provided the classifiers are different enough, and those differences are complementary. We shall look in more detail at the nature of this difference or ‘diversity’ in the next chapter.

The circumstances when it is sensible to consider combining, due to a set of different classifiers being produced, are described succinctly by Jain et al. [45]:

1. We may have access to a number of different classifiers, based upon different representations of the same problem e.g. person identification via voice, face and hand-writing.
2. Different training sets may be available collected under different circumstances or at different times.
3. Different classifiers may exhibit local differences with each being a specialist in a specific region.
4. Unstable classifiers, such as neural networks, can have quite different results due to different initial conditions. Unstable refers to the fact that very small changes to the initial information can result in large changes to the resulting neural network and its output.

Rather than selecting the best classifier, which may not be that much better than the others, we combine their opinions taking advantage of all of the attempts to learn the data.

The following notation must also be introduced to deal with multiple classifier systems:

- Let $\mathcal{D} = \{D_1, D_2, \dots, D_L\}$ be a set of classifiers.
- The classifier outputs are usually c -dimensional vectors $D_i(\mathbf{x}) = [d_{i,1}(\mathbf{x}), \dots, d_{i,c}(\mathbf{x})]^T$ where $d_{i,j}(\mathbf{x})$ is the degree of “support” given by classifier D_i to the hypothesis that \mathbf{x} comes from class ω_j , $j = 1, \dots, c$. Without loss of generality we can restrict $d_{i,j}(\mathbf{x})$ within the interval $[0, 1]$, $i = 1, \dots, L$, $j = 1, \dots, c$, and call the classifier outputs “soft labels”. Most often $d_{i,j}(\mathbf{x})$ is an estimate of the posterior probability $P(\omega_j|\mathbf{x})$.
- Combining classifiers means we combine the L classifier outputs $D_1(\mathbf{x}), \dots, D_L(\mathbf{x})$ to get a soft label for \mathbf{x} , denoted $\mathcal{D}(\mathbf{x}) = [\mu_1(\mathbf{x}), \dots, \mu_c(\mathbf{x})]^T$. Here $\mu_j(\mathbf{x})$ is the

combined value for class ω_j , i.e., the support to the hypothesis of object \mathbf{x} being from class ω_j , which can be interpreted in some cases as probability or likelihood.

- If a crisp class label of \mathbf{x} is needed, ('crisp' refers to the need for a single choice of class label for an object \mathbf{x}), we can use the maximum membership rule to calculate $\tilde{D}(\mathbf{x})$, the combined classification decision, i.e., the decision made by the aggregation algorithm as to which class object \mathbf{x} should be in :

Assign \mathbf{x} to class ω_s iff,

$$d_{i,s}(\mathbf{x}) \geq d_{i,j}(\mathbf{x}) \quad \forall j = 1, \dots, c. \quad \text{for individual crisp labels by } D_i, \quad (2.1)$$

$$\mu_s(\mathbf{x}) \geq \mu_t(\mathbf{x}), \quad \forall t = 1, \dots, c. \quad \text{for the final crisp label.} \quad (2.2)$$

Ties are resolved arbitrarily. The minimum-error classifier is recovered from (2.2) when $\mu_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$.

2.2 How multiple classifier systems work

Figure 2.1 shows how a multiple classifier system works. The feature values for object \mathbf{x} , (x_1, \dots, x_n) , are submitted individually to the L classifiers. Each of the classifiers uses the feature values to classify the object. The results from the classifiers are then combined (see Section 2.4). The combination provides a classification for the object \mathbf{x} .

Two approaches to this combination are [57,115]:

- **Dynamic Classifier Selection** which tries to predict which classifier is most likely to be correct for each object and only that classifier's output is used to assign the class label to \mathbf{x} .
- **Classifier Fusion** which takes all of the individual classifier outputs and combines them to calculate the support for each class.

In our work we are only considering classifier fusion methods.

2.3 Strategies for building classifier ensembles

Figure 2.2 shows four different aspects of the multiple classifier system which we can choose to manipulate to try to improve the classification accuracy [54]. We can select the combination method used (A), the classifier models used (B), the feature subsets we submit to the classifiers (C) or the training set used (D). We can alter one or more of these at any one time. Hopefully, by changing these we can produce an ensemble of classifiers which are different enough from each other to provide complementary information and thus an improved accuracy over the individual classifiers when combined.

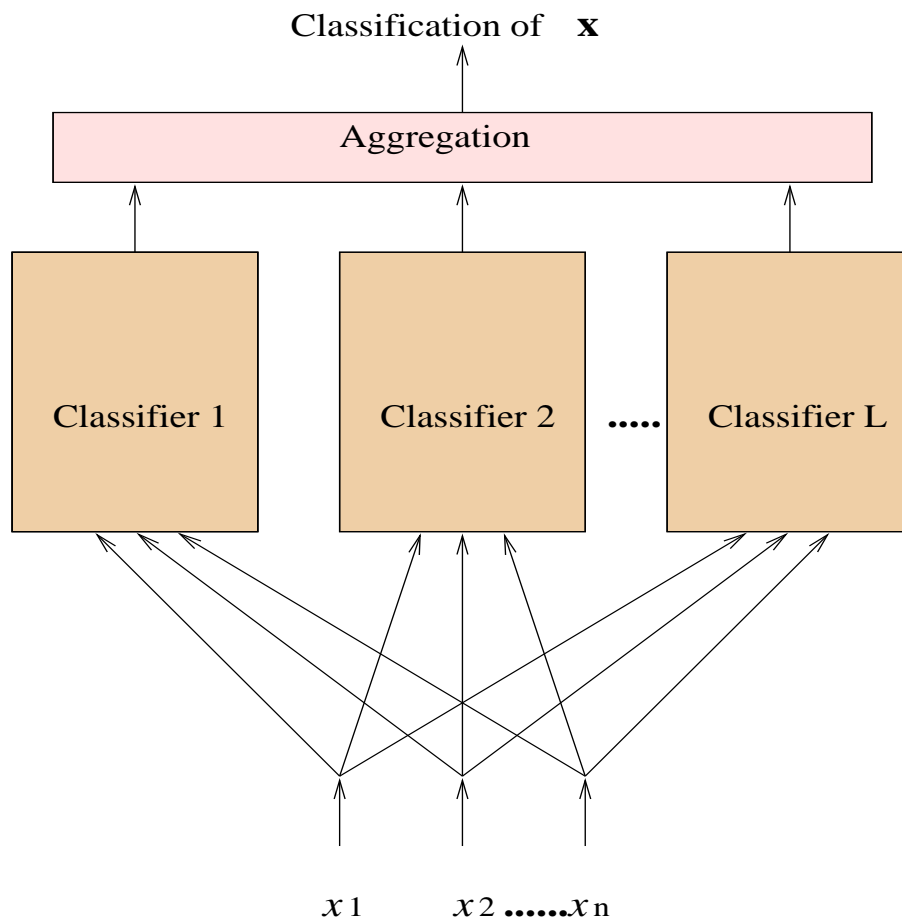


Figure 2.1: A MULTIPLE CLASSIFIER SYSTEM

A Combination methods There are many different combination methods we could use, there is more information on some of them in the rest of this chapter.

B Classifier models There are many different classifier models we could choose from, some of the various classifier models have already been mentioned (see 1.3).

C Feature subsets If a set of classifiers is built on different features then intuitively they should be different from each other. There are two ways of obtaining different feature subsets, we can use

1. Feature selection- finding as small a subset of the features as possible whilst still ensuring that the accuracy of the classifier using the subset is as high as possible. The random subspace method is a feature selection approach [41].
2. Feature extraction- usually uses Principal Component Analysis to perform a set of transformations (linear or non-linear) on the whole feature set to obtain a different set of features.

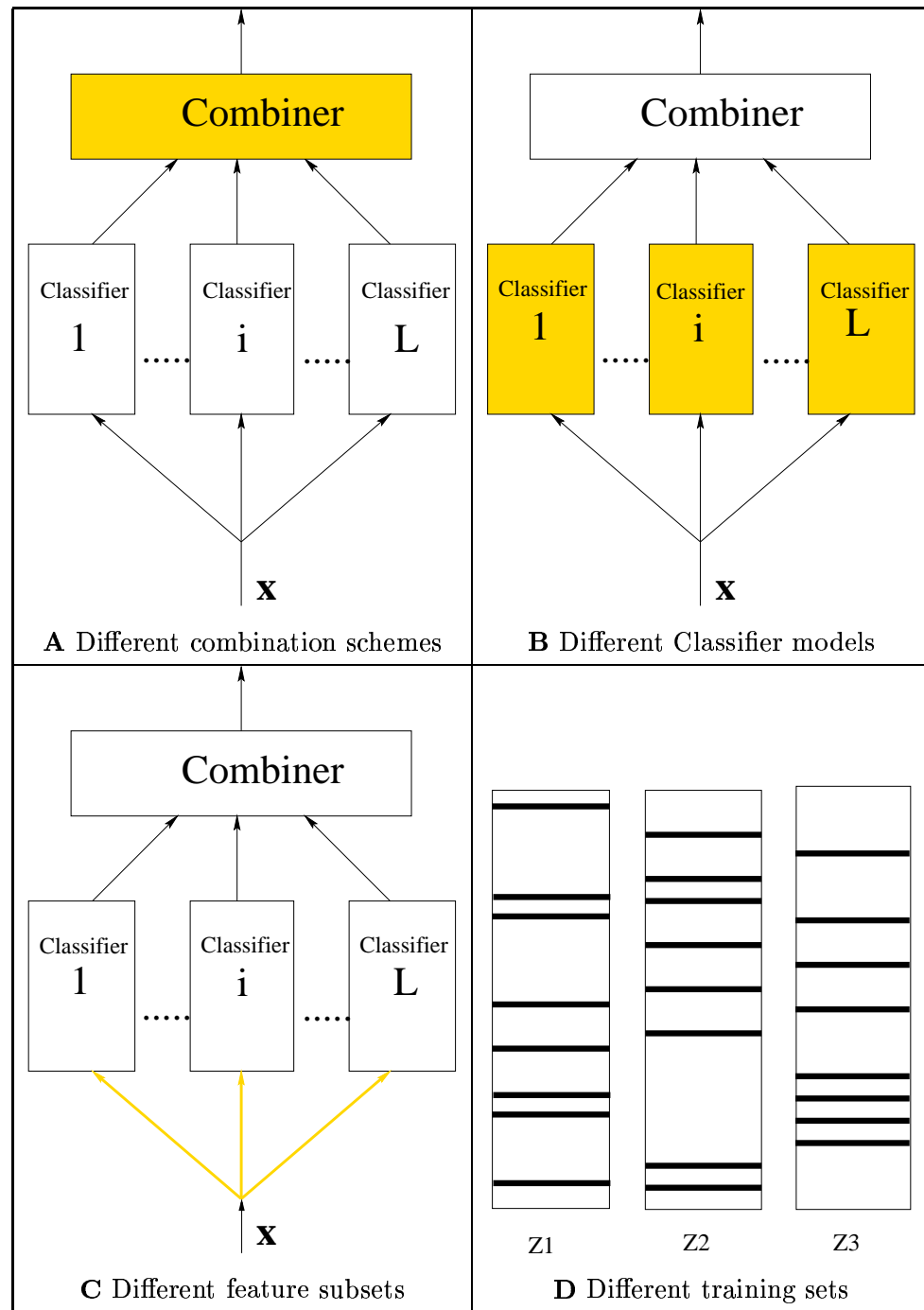


Figure 2.2: WHAT CAN WE CHANGE IN A MULTIPLE CLASSIFIER SYSTEM? [54]

D Training sets These can be modified in several ways to obtain a different training set for each classifier. In this way we hope to obtain a set of different classifiers (see 4). Some approaches are:

- Bagging [8, 9]. Here we take bootstrap replications of the data set, i.e., if we have an original set of size N we take a random set of N examples from the

data set (allowing repeats) for each classifier.

- AdaBoost with re-sampling [9, 31]. Here instead of taking *random* bootstrap replications of the data set, weights are assigned to each example in the data set and those which are deemed to be difficult to classify by earlier classifiers have higher chance of being put in the training set for future classifiers.

Other Approaches These are methods which do not easily fit into any of the categories above.

1. Injecting Randomness. This approach involves adding an element of randomness into the procedure. For neural networks this is done by randomly choosing the initial weights [21], and for decision trees by randomly selecting the feature that decides the split at each node [11].
2. Manipulating Output Features. We can alter the output feature for example, by turning a multi-class problem into a set of binary problems as in error-correcting output coding [76, 77]. Also we can include some randomness by randomising the outputs. This can involve introducing noise by altering some class labels whilst maintaining the same proportion of each (known as output flipping) [10]. Alternatively we can create a vector for each training example with value 1 for the true class label and 0 for all the other possible class labels, rather than a single class label, and then we can add Gaussian noise to this vector (known as output smearing) [10].

2.4 Combination methods

2.4.1 Classifier Fusion

There are many different classifier fusion algorithms. These take the outputs of several classifiers to give a combined output which is hopefully more accurate than that of the individual classifiers. There are three classes of classifier depending on the amount of information produced for a given input \mathbf{x} [116].

1. Abstract type - when a classifier D only outputs a single class label.
2. Rank type - when a classifier D ranks all classes; the class with the highest rank is the most likely label for \mathbf{x} (according to D) and the class with the lowest rank is the least likely label for \mathbf{x} .
3. Measurement type - when a classifier D attributes a measurement value to each class label according to its support for that class label (this could be a probability value or a distance measurement, etc.).

All types of classifiers can produce information at the abstract level and so combining at this level is possible even for very different types of classifiers. The measurement level contains the most amount of information and the abstract level the least. Many classifiers pass through a measurement level as an intermediate stage in the classification process, e.g., those that approximate $P(\omega_i)$ and $p(\mathbf{x}|\omega_i)$ or $P(\omega_i|\mathbf{x})$ (the prior probabilities, class conditional p.d.f. and posterior probabilities) or those that measure the distance between the object \mathbf{x} and each prototype sample, \mathbf{z}_j , from each class. In order to combine these different forms a transformation to a common scale would be necessary before combination could occur. The rank level avoids this problem since ranking can easily be obtained from the measurement level allowing combination of different types of classifier but still retaining more information than the abstract level. However, the rank level is effectively ignored in practice so we will only concern ourselves with the abstract and measurement levels of information.

Table 2.1 gives the level of information required for a set of commonly seen algorithms.

Table 2.1: CLASSIFIER FUSION ALGORITHMS AND THE LEVEL OF INFORMATION THEY REQUIRE

<u>Abstract Level Fusion Algorithms</u>
Voting methods including majority vote
Naive Bayes combination
Behavior-knowledge space and Wernecke's method
<u>Measurement Level Fusion Algorithms</u>
Minimum, maximum, average and product
Probabilistic product
Fuzzy integral
Decision templates
Dempster-Schafer combination

2.4.2 Maximum, Minimum, Average and Product (*MAX*, *MIN*, *AVR*, *PRO*) [45, 48]

These are some of the simplest and most commonly used combination methods. Once the classifiers in the ensemble are trained, these combination methods do not require any further training. The equation for implementing these combination methods is given

below.

$$\mu_j(\mathbf{x}) = \mathcal{O}(d_{1,j}(\mathbf{x}), \dots, d_{L,j}(\mathbf{x})), \quad j = 1, \dots, c. \quad (2.3)$$

where \mathcal{O} is the respective operation (maximum, minimum, average or product) and $d_{i,j}(\mathbf{x})$ is the support given by classifier D_i to the hypothesis that \mathbf{x} comes from class ω_j . The class ω_j with maximum μ_j is the assigned class for the given input \mathbf{x} . Table 2.2 shows an example of how these simple combination methods work. Recall, ‘Crisp Decision’ means the choice of a single class label for an object \mathbf{x} . The highest $\mu_j(\mathbf{x})$, $j = 1, 2$, for each of the combination methods is underlined indicating which class will be chosen for the crisp decision. Note that for the same set of classifier outputs *MIN* and *MAX* give the same crisp decision as each other but that this is different from *AVR* and *PRO*.

Table 2.2: AN EXAMPLE SHOWING HOW THE SIMPLE COMBINATION METHODS WORK

Classifier	Support for ω_1	Support for ω_2	Crisp Decision
D_1	0.8	0.2	ω_1
D_2	0.4	0.6	ω_2
D_3	0.3	0.7	ω_2
D_4	0.6	0.4	ω_1
D_5	0.3	0.7	ω_2
<i>MIN</i>	<u>0.3</u>	0.2	ω_1
<i>MAX</i>	<u>0.8</u>	0.7	ω_1
<i>AVR</i>	0.48	<u>0.52</u>	ω_2
<i>PRO</i>	0.01728	<u>0.2352</u>	ω_2

2.4.3 Voting Methods

Threshold voting considers each classifier as a voter and assigns a threshold value to the situation. The general formula for threshold voting is:

$$\tilde{\mathcal{D}} = \begin{cases} e, & \text{if } \sum_{d=1}^L d_{d,e}(\mathbf{x}) > \sum_{d=1}^L d_{d,f}(\mathbf{x}) \geq \alpha \times L \quad \forall f = 1, \dots, c, f \neq e, \\ c+1 & \text{otherwise} \end{cases} \quad (2.4)$$

where $0 \leq \alpha \leq 1$, L is the number of classifiers, $d_{i,j}$ is the support given by classifier D_i to the hypothesis that \mathbf{x} belongs to class ω_j and $c+1$ denotes the option to reject the object if it cannot be assigned a class label. There are various special cases of the threshold vote.

Unanimous Consensus (UC) [116] This requires all classifiers to agree on the class label of an object otherwise it rejects classification. This corresponds to $\alpha = 1$ in equation 2.4.

Consensus with abstentions (CA) [116] This is similar to the unanimous consensus but allows individual classifiers to reject classification as long as no classifier supports a different class. This has a slightly different formula:

$$\tilde{D} = \begin{cases} e, & \text{if } \sum_{d=1}^L d_{d,e}(\mathbf{x}) > 0 \text{ and } d_{d,f}(\mathbf{x}) = 0 \ \forall f = 1, \dots, c, f \neq e, \\ c + 1 & \text{otherwise} \end{cases} \quad (2.5)$$

Majority vote [116] This requires that a majority of the classifiers agree on the class label, i.e., that $\alpha = \frac{1}{2}$ in equation 2.4. If there are more than two classes the class with the most may not necessarily have more than half the votes. In this case plurality vote may be used.

Plurality vote [116] This is the weakest combination using voting, it assigns the class label for which the highest number of classifiers vote. This also has a slightly different formula:

$$\tilde{D} = \begin{cases} e, & \text{if } \sum_{d=1}^L d_{d,e}(\mathbf{x}) > d_{d,f}(\mathbf{x}) \ \forall f = 1, \dots, c, f \neq e, \\ c + 1 & \text{otherwise} \end{cases} \quad (2.6)$$

If there is a tie for the number of votes it may be broken randomly or a rejection may occur.

2.4.4 Majority vote

Majority vote (*MAJ*) takes the individual classifier opinions and assigns the object to the class which the majority of classifiers would assign it. We consider L classifiers acting on a data set of size N . Let $C_j = [c_{1j}, \dots, c_{Nj}]^T$, $C_j \in \{0, 1\}^N$ be an N -place binary vector formed in the following way:

$$c_{ij} = \begin{cases} 1, & \text{if } \mathbf{z}_i \in Z \text{ is correctly classified by } D_j \\ 0, & \text{otherwise} \end{cases}, \quad (2.7)$$

where $j = 1, \dots, L$ and $i = 1, \dots, N$. Let $M = [m_1, \dots, m_N]^T$ be the vector containing the majority vote result calculated by:

$$m_i = \begin{cases} 1, & \text{if } \sum_{j=1}^L c_{ij} \geq k \\ 0, & \text{otherwise} \end{cases}, \quad (2.8)$$

where $i = 1, \dots, N$ and $k = \frac{L+1}{2}$ for odd L and $k = \frac{L}{2} + 1$ for even L . The accuracy is therefore:

$$P_{majority} = \frac{|M|}{N}, \quad (2.9)$$

where $|M|$ is the sum of elements of M .

Let $\{D_1, \dots, D_L\}$ be the set of classifiers and $\Omega = \{\omega_1, \omega_2\}$. Let $D_i(\mathbf{x}) = \omega_j$ $i = 1, \dots, L$ $j \in \{1, 2\}$. If L is odd then no ties are possible. If L is even then ties are possible. If $c > 2$ then ties are always possible whether L is odd or even.

For example, let $c = 3$, $L = 5$, $\Omega = \{\omega_1, \omega_2, \omega_3\}$. Suppose that for some $\mathbf{x} \in \mathfrak{R}$, $D_1(\mathbf{x}) = \omega_1$, $D_2(\mathbf{x}) = \omega_2$, $D_3(\mathbf{x}) = \omega_2$, $D_4(\mathbf{x}) = \omega_3$, $D_5(\mathbf{x}) = \omega_3$. Here the votes tie (two votes for each of ω_2 and ω_3), and, besides there is no class label for which the majority (50%+1, i.e., 3 in this case) is obtained.

Another example is with $c = 4$, $L = 5$, i.e., $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$, and $D_1(\mathbf{x}) = \omega_2$, $D_2(\mathbf{x}) = \omega_4$, $D_3(\mathbf{x}) = \omega_2$, $D_4(\mathbf{x}) = \omega_3$, $D_5(\mathbf{x}) = \omega_1$. Here there are no ties, the maximum votes are for ω_2 , but there is no class label for which the majority (3, as before) is reached. In cases like this it may be preferable to use the class with the most votes, called ‘plurality vote’, or reject the object depending on the particular problem.

It has been shown [52, 67], that if we assume that the L classifiers are *independent* with the same probability of correct classification, p , then the probability of an accurate consensus \mathcal{P} can be calculated using the binomial distribution as:

$$\mathcal{P} = \sum_{i=k}^L \binom{L}{i} p^i (1-p)^{L-i} \quad (2.10)$$

where $k = \frac{L+1}{2}$, and L is odd. k is the value at which a majority is reached, e.g., $L = 5 \Rightarrow k = 3$ (3 classifiers agree gives a majority). For a justifiable combination of classifiers we require this \mathcal{P} to be greater than any one of the individual classifiers and hence:

$$\sum_{i=k}^L \binom{L}{i} p^i (1-p)^{L-i} > p \quad (2.11)$$

Consider the case for 5 classifiers, $L = 5, k = 3$

$$\begin{aligned} \sum_{i=3}^5 \binom{5}{i} p^i (1-p)^{5-i} &= \binom{5}{3} p^3 (1-p)^2 + \binom{5}{4} p^4 (1-p) + \binom{5}{5} p^5 \\ &= p^3 (10 - 15p + 6p^2) > p \end{aligned} \quad (2.12)$$

This is true $\forall p \in (0.5, 1)$.

Provided the classifiers have individual accuracy greater than 50% and are *independent*, it is guaranteed that the majority vote will give an improvement in accuracy on the individual classifiers [67].

2.4.5 Weighted voting

If we consider the analogy of the respiratory patient once again (Chapter 1.2), in an ideal world we would hope to have a team of experienced nurses, doctors and consultants all giving their opinion and discussing with one another. Unanimous consensus would be the

case where all the medical staff agree on a diagnosis. Consensus with abstentions, would be the case where some medical staff may decline to give an opinion and the rest would all agree on the diagnosis. Majority vote would be when the majority of staff agree, and plurality would go with the diagnosis which got the most votes from members of staff. In reality a consultant's opinion would be considered to carry more weight than a junior doctor's or a nurse's regardless of their experience. In classifier systems we encounter this inequality of the classifiers. Some classifiers may be much more accurate than others but the ensemble of classifiers could still benefit from considering all opinions. Also sometime a classifier may become incredibly specialised, i.e. in our analogy a consultant may be a specialist in respiratory care but know very little about other conditions. To deal with this kind of situation we can assign weights to our experts, so we may give a high weight to the consultant's opinion and lesser weights to the junior doctors' and nurses' opinions. In a similar way we assign weights to our classifiers before combining with voting [3, 46, 111]. The equation for weighted majority voting is:

$$\tilde{\mathcal{D}} = \begin{cases} e, & \text{if } \sum_{d=1}^L w_d d_{d,e}(\mathbf{x}) > \sum_{d=1}^L w_d d_{d,f}(\mathbf{x}) \geq \frac{L+1}{2} \quad \forall f = 1, \dots, c, f \neq e, \\ c + 1 & \text{otherwise} \end{cases} \quad (2.13)$$

where L is odd, and w_d is the weight assigned to classifier D_d .

2.4.6 Limits of majority vote

In previously published work [64] we derived upper and lower limits on the majority vote accuracy with respect to individual classifier accuracy, p and the number of classifiers in the ensemble, L . As has been previously mentioned *independent* classifiers with individual accuracy $p > 0.5$ guarantee an improvement when combined by majority vote. We considered the case for *dependent* classifiers and how their similarity/dissimilarity could affect the combined accuracy.

We did this by considering two probability distributions over the possible combinations of L correct/incorrect votes. Consider the pool \mathcal{D} of L (odd) classifiers, each with accuracy p . For the majority vote to give a correct classification we need $\lfloor \frac{L}{2} \rfloor + 1$ or more classifiers to be correct.

Intuitively the best improvement over the individual classifier accuracy will be achieved when *exactly* $\lfloor \frac{L}{2} \rfloor + 1$ votes are correct (where $\lfloor \star \rfloor$ indicates the floor, or largest integer smaller than, \star). We denoted this case the 'Pattern of Success'. Any extra correct votes will be 'wasted' as they will be unnecessary to gain the correct vote and any correct votes in combinations not leading to a correct combined decision will also be 'wasted'. Similarly the 'Pattern of Failure' occurs when exactly one less than the majority, i.e. $\lfloor \frac{L}{2} \rfloor$, of the classifiers are correct.

Table 2.3: ACCURACY UPPER LIMITS WHEN COMBINING INDEPENDENT AND DEPENDENT CLASSIFIERS WITH MAJORITY VOTE

L	3			5			7		
p	Ind	DepS	DepF	Ind	DepS	DepF	Ind	DepS	DepF
0.5	0.5	0.75	0.25	0.5	0.83	0.16	0.5	0.875	0.125
0.6	0.648	0.9	0.4	0.683	1	0.3	0.71	1	0.3
$\frac{2}{3}$	0.741	1		0.790	1		0.827	1	
0.7	0.784	1		0.837	1		0.874	1	
0.8	0.896	1		0.942	1		0.967	1	
0.9	0.972	1		0.991	1		0.997	1	

In [64] we derived two formulas to give us an upper and lower limit on the majority vote accuracy.

$$\text{'Pattern of Success', Upper limit:} \quad P_{majmax} = \min \left\{ 1, \frac{pL}{\lfloor \frac{L}{2} \rfloor + 1} \right\} \quad (2.14)$$

$$\text{'Pattern of Failure', Lower limit:} \quad P_{majmin} = \frac{pL - \lfloor \frac{L}{2} \rfloor}{\lfloor \frac{L}{2} \rfloor + 1} \quad (2.15)$$

$$\text{and as } L \text{ increases :} \quad \lim_{L \rightarrow \infty} P_{majmin} = 2p - 1 \quad (2.16)$$

If $p > \frac{2}{3}$ then $P_{majmax} = 1$ but $P_{majmin} \rightarrow \frac{1}{3}$ for large L . Both the pattern of success and the pattern of failure cases are cases of *dependent* classifiers and show the problem a user faces. We can achieve much higher accuracies, than by using independent classifiers, but we can also achieve much lower accuracies by using dependent classifiers. It is knowing whether or not we will gain an improvement that is the problem. Considering the independent case (Ind) shown in equation 2.10 and the dependent cases Success, (DepS) shown in equation 2.14, and Failure shown in equation 2.15, Table 2.3 gives the various values of majority vote accuracy obtained for $L = 3, 5, 7$ individual classifiers with accuracy $p = 0.5, 0.6, \frac{2}{3}, 0.7, 0.8, 0.9$. This illustrates both the theoretical advantages in using dependent classifiers rather than using independent classifiers and the disadvantages. These values show that if we have the 'right' sort of dependence we can improve the accuracy considerably, especially with a small number of classifiers. It is therefore worth looking in more detail at the nature of this dependence or diversity, as we shall see in chapter 3.

2.4.7 Naive Bayes (NB) [15, 44, 57, 116]

Xu et al. and others more often refer to this combination method as Bayes combination [15, 44, 116]. However this method relies on the assumption that the classifiers are mutually

independent. In reality this situation does not occur and this is the reason we use the term “naive” [57]. Consider the crisp class labels obtained from $D_1(\mathbf{x}), \dots, D_L(\mathbf{x})$ by (2.1), so in this case $D_i(\mathbf{x}) \in \Omega, i = 1, \dots, L$. Let s_1, \dots, s_L be the crisp class labels assigned to \mathbf{x} by classifiers D_1, \dots, D_L , respectively. The independence assumption leads to

$$\mu_j(\mathbf{x}) \propto \prod_{i=1}^L \hat{P}(\omega_j | D_i(\mathbf{x}) = s_i), \quad (2.17)$$

where $\hat{P}(\omega_j | D_i(\mathbf{x}) = s_i)$ are probability estimates calculated from the data.

$$\hat{P}(\omega_j | D_i(\mathbf{x}) = s_i) = \frac{\text{number of objects labelled } s_i \text{ by } D_i \text{ whose true label is } \omega_j}{\text{number of objects labelled } s_i \text{ by } D_i}$$

The following example illustrates the *NB* combination method. Let $L = 3$ and $c = 2$. Suppose that the confusion matrices of the three classifiers, calculated on a data set \mathbf{Z} with 100 objects are as shown in Table 2.4.

Table 2.4: THE CONFUSION MATRICES OF CLASSIFIERS D_1 , D_2 , AND D_3 .

	D_1			D_2			D_3		
	Guessed label			Guessed label			Guessed label		
		ω_1	ω_2		ω_1	ω_2		ω_1	ω_2
True label	ω_1	36	22	ω_1	41	17	ω_1	23	35
	ω_2	22	20	ω_2	20	22	ω_2	26	16

Let the output of the three classifiers for some $\mathbf{x} \in \mathfrak{R}^n$ be such that $[s_1, s_2, s_3] = [\omega_2, \omega_1, \omega_2]$. Majority vote would label \mathbf{x} in ω_2 . However, the support for that class label may not be very strong even though it is hypothesised by 2 of the 3 classifiers. For the naive Bayes combination,

$$\begin{aligned} \hat{P}(\omega_1 | s_1 = \omega_2) &= \frac{22}{42} & \hat{P}(\omega_2 | s_1 = \omega_2) &= \frac{20}{42} \\ \hat{P}(\omega_1 | s_2 = \omega_1) &= \frac{41}{61} & \hat{P}(\omega_2 | s_2 = \omega_1) &= \frac{20}{61} \\ \hat{P}(\omega_1 | s_3 = \omega_2) &= \frac{35}{51} & \hat{P}(\omega_2 | s_3 = \omega_2) &= \frac{16}{51} \end{aligned} \quad (2.18)$$

$$\mu_1(\mathbf{x}) \propto \frac{22}{42} \cdot \frac{41}{61} \cdot \frac{35}{51} \approx 0.242 \quad > \quad \mu_2(\mathbf{x}) \propto \frac{20}{42} \cdot \frac{20}{61} \cdot \frac{16}{51} \approx 0.05$$

Accordingly, class ω_1 will be assigned.

2.4.8 Behavior-Knowledge Space and Wernecke’s method (*BKS*, *WER*) [44, 112, 115]

BKS works by considering every possible combination of class labels as an index to a cell in a look-up table (*BKS* table) [44]. Recall that $\mathbf{s} = (s_1, \dots, s_L) \in \Omega^L$ are the crisp

class labels assigned to \mathbf{x} by classifiers D_1, \dots, D_L , respectively. \mathbf{s} can be considered as an L -dimensional random variable and we try to estimate $\mu_i(\mathbf{x}) = \hat{P}(\omega_i|\mathbf{s})$. We design the table using a labelled data set \mathbf{Z} . For each training object, $\mathbf{z}_j \in \mathbf{Z}$, we consider $D_1(\mathbf{z}_j), \dots, D_L(\mathbf{z}_j)$. \mathbf{z}_j is then placed in the cell indexed by $D_1(\mathbf{z}_j), \dots, D_L(\mathbf{z}_j)$ and we tally the number of objects from each class in each cell. The class label with maximum occurrence is selected as the label for this cell. Sometimes we may have ties or the cell may be empty, in which case we resolve the ties arbitrarily, and label the empty cells either at random or by some other method if applicable. After the table has been designed, the *BKS* method labels an $\mathbf{x} \in \mathbb{R}^n$ to the class of the cell indexed by $D_1(\mathbf{x}), \dots, D_L(\mathbf{x})$.

For the example discussed in section 2.4.7, assume again that D_1, D_2 and D_3 produce output $(s_1, s_2, s_3) = (\omega_2, \omega_1, \omega_2)$. As we can see in Table 2.5 there have been 22 objects in \mathbf{Z} for which this combination of labels occurred; 15 having label ω_1 , and 7 having label ω_2 . Hence the table cell indexed by $(\omega_2, \omega_1, \omega_2)$ is labelled ω_1 no matter that the majority of the classifiers suggest otherwise. Therefore *BKS* would assign class ω_1 to object \mathbf{x} .

Table 2.5: EXAMPLE SHOWING HOW THE *BKS* TABLE IS CONSTRUCTED AND USED

$D_1(\mathbf{x}), D_2(\mathbf{x}), D_3(\mathbf{x})$	1,1,1		1,1,2		1,2,1		2,1,1		1,2,2		2,1,2		2,2,1		2,2,2	
Number from class ω_1, ω_2	5	0	17	4	11	13	4	9	3	5	15	7	3	4	0	0
Cell Label	ω_1		ω_1		ω_2		ω_2		ω_2		ω_1		ω_2		ω_2^*	

* decision by majority since all three classifiers would assign class ω_2 .

Wernecke's model is similar to the *BKS*. The difference is that in constructing the table, Wernecke [112] considers the 95% confidence intervals of the frequencies in each cell. If there is overlap between the intervals, the "least wrong" classifier among the L members of the team is identified and authorised to label \mathbf{x} . For this, L estimates of the probability $P(\text{error and } D_i(\mathbf{x}) = s_i)$ are calculated. Then the classifier with the smallest probability is nominated for labelling the cell. For an $\mathbf{x} \in \mathbb{R}^n$, the cell is identified by the labels assigned by D_1, \dots, D_L and then either the cell label is recovered or the label of the nominated classifier is taken as the label of \mathbf{x} .

To continue the example illustrating *BKS* combination method shown in Table 2.5, we would calculate the 95 % confidence intervals, using Chebyshev's inequality (e.g., see [36]). If the confidence intervals are overlapping, estimates of $P(\text{error and } D_i(\mathbf{x}) = s_i)$ have to be obtained. Using the data in the confusion matrices, Table 2.4,

$$\hat{P}(\text{error and } D_1(\mathbf{x}) = \omega_2) = \hat{P}(\omega_1|s_1 = \omega_2)\hat{P}(s_1 = \omega_2) = \frac{22}{42} \cdot \frac{42}{100} = \frac{22}{100}$$

$$\begin{aligned}\hat{P}(\text{error and } D_2(\mathbf{x}) = \omega_1) &= \hat{P}(\omega_2 | s_2 = \omega_1) \hat{P}(s_2 = \omega_1) = \frac{20}{61} \cdot \frac{61}{100} = \frac{20}{100} \\ \hat{P}(\text{error and } D_3(\mathbf{x}) = \omega_2) &= \hat{P}(\omega_1 | s_3 = \omega_2) \hat{P}(s_3 = \omega_2) = \frac{35}{51} \cdot \frac{51}{100} = \frac{35}{100}\end{aligned}$$

As $\hat{P}(\text{error and } D_2(\mathbf{x}) = \omega_1)$ is the smallest of the three, classifier D_2 is authorised to label \mathbf{x} , and thus the assigned class is ω_1 .

2.4.9 Decision Templates (DT) [57]

The classifier outputs can be conveniently organised in a **decision profile** as the following matrix

$$DP(\mathbf{x}) = \begin{bmatrix} d_{1,1}(\mathbf{x}) & \dots & d_{1,j}(\mathbf{x}) & \dots & d_{1,c}(\mathbf{x}) \\ \dots & & & & \\ d_{i,1}(\mathbf{x}) & \dots & d_{i,j}(\mathbf{x}) & \dots & d_{i,c}(\mathbf{x}) \\ \dots & & & & \\ d_{L,1}(\mathbf{x}) & \dots & d_{L,j}(\mathbf{x}) & \dots & d_{L,c}(\mathbf{x}) \end{bmatrix}. \quad (2.19)$$

Using decision templates (DT) for combining classifiers is proposed in [57]. Given L (trained) classifiers in \mathcal{D} , c decision templates are calculated from the data, one per class. This approach works by comparing the c DT's produced to a characteristic template for each class, the decision profile. It is therefore able to use outputs for all classes to calculate the final support for each class unlike other methods which only use the support for that particular class. For each class label, i , there is a decision template, DT_i , and each decision template is an $L \times c$ matrix whose (k, l) th element is:

$$dt_i(k, l) = \frac{\sum_{j=1}^N Ind(\mathbf{z}_j, i) d_{k,l}(\mathbf{z}_j)}{\sum_{j=1}^N Ind(\mathbf{z}_j, i)}, \quad k = 1, \dots, L, \quad l = 1, \dots, c. \quad (2.20)$$

where $Ind(\mathbf{z}_j, i)$ is an indicator function with value 1 if \mathbf{z}_j has crisp label i , and 0 otherwise.

DT_i can be regarded as the expected $DP(\mathbf{x})$ for class ω_i . The support for the class offered by the combination of the L classifiers, $\mu_i(\mathbf{x})$ is then found using a measure of *similarity* between the current $DP(\mathbf{x})$ and DT_i :

$$\mu_i(\mathbf{x}) = \mathcal{S}(DT_i, DP(\mathbf{x})) \quad (2.21)$$

Here we use the squared Euclidean distance for calculating the similarity, i.e., the lower the squared Euclidean distance between the matrices entries, the more similar they are. Therefore we calculate μ_i by taking 1 minus the squared Euclidean distance. Any other measure of similarity/dissimilarity can also be applied in a similar manner.

$$\mu_i(\mathbf{x}) = 1 - d_E(DP(\mathbf{x}), DT_i) = 1 - \sum_{j=1}^c \sum_{k=1}^L (d_{k,j}(\mathbf{x}) - dt_i(k, j))^2, \quad (2.22)$$

where $dt_i(k, j)$ is the k, j -th entry in decision template DT_i .

As an example, assume that the following Decision Templates have been obtained from a data set \mathbf{Z} using equation (2.20), with $c = 2$, $L = 3$:

$$DT_1 = \begin{pmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \\ 0.6 & 0.4 \end{pmatrix}, \quad DT_2 = \begin{pmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix}.$$

Given an object \mathbf{x} with Decision Profile

$$DP(\mathbf{x}) = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.3 & 0.7 \end{pmatrix},$$

we use the squared Euclidean distance to calculate the similarity, the closer DT_1 is to $DP(\mathbf{x})$ the more similar they are, and subsequently the higher the support for class ω_1 and similarly for DT_2 .

$$\begin{aligned} d_E(DP(\mathbf{x}), DT_1) &= (0.6 - 0.7)^2 + (0.4 - 0.3)^2 + (0.4 - 0.5)^2 \\ &\quad + (0.6 - 0.5)^2 + (0.3 - 0.6)^2 + (0.7 - 0.4)^2 = 0.22 \\ d_E(DP(\mathbf{x}), DT_2) &= (0.6 - 0.4)^2 + (0.4 - 0.6)^2 + (0.4 - 0.6)^2 \\ &\quad + (0.6 - 0.4)^2 + (0.3 - 0.2)^2 + (0.7 - 0.8)^2 = 0.18 \\ \mu_1(\mathbf{x}) &= 1 - d_E(DP(\mathbf{x}), DT_1) = 1 - 0.22 = 0.78 \\ \mu_2(\mathbf{x}) &= 1 - d_E(DP(\mathbf{x}), DT_2) = 1 - 0.18 = 0.82 \end{aligned}$$

Since $\mu_1 > \mu_2$ we assign class label ω_1 to \mathbf{x} .

2.4.10 The Oracle (ORA)

The Oracle is named after the seers of ancient mythology as it works by correctly classifying an object provided at least one of the L classifiers correctly classifies the object. It is obviously not a true combination method but an abstraction which gives the possible upper limit on the classification accuracy. Here we use the oracle to compare the combination methods' performance.

2.4.11 Data-dependent weights and multi-level classifiers

Data-dependent weights are often used with combination methods for multiple classifier systems. These weights are developed from the training data and are then used during combination. The weights are adjusted to improve the ensemble performance during the training process.

Multi-level classifiers work by having one classifier's outputs as the inputs for another classifier. That is, the output decision profile, $DP_{Da}(\mathbf{x})$, from classifier, D_a , is submitted

to another classifier, D_b , as its input and its output is $DP_{D_b}(DP_{D_a}(\mathbf{x}))$. They are also known as stacked classifiers.

2.5 Existing empirical studies of combination methods

There have been many studies comparing various methods of combination methods (e.g. [48, 57, 111, 116]). Many of these studies introduce a new combination method and then compare its performance with other popularly used methods. Kittler et al. [48] compared some of the simpler combination methods which do not require second level training: the minimum, maximum, sum, product, median rules and majority vote. They compared the methods using four types of base classifiers on a U.S. postal service database consisting of handwritten digits - CEDAR-CDROM. Their results showed that the sum and median rules were the best, and that in fact the sum rule is very robust despite being the rule with the most restrictive assumptions. The majority vote rule and maximum rule also performed better than the single best classifier whilst the minimum and product rules were both worse than each of the individual classifiers.

Kuncheva et al. [57] introduced Decision Templates (with various similarity measures) and compared them with 13 other combination methods, minimum, maximum, average, product, majority vote, naive Bayes, behavior-knowledge space, probabilistic product, Dempster-Schafer, fuzzy integral, linear discriminant classifier, quadratic discriminant classifier, and the logistic classifier, (the latter three use the base classifier outputs as input allowing them to act as a combination method). Using quadratic discriminant classifiers as the base classifiers and using the satimage and phoneme databases their results show that if you choose the ‘correct’ similarity measures decision templates outperform all the other combination methods. They give a ranking order of the combination methods over the two datasets. Out of eleven similarity methods, decision templates (DT) with 5 of the similarity measures gave higher accuracies than all other methods, and DT with another similarity measure was better than all combination methods apart from Product (which was next best after the five DT methods) and Dempster-Schafer. Minimum and Average were the next best followed by another DT version. The order of the remaining methods was: majority vote, fuzzy integral, logistic classifier, maximum, probabilistic product, behavior knowledge space and naive Bayes. All these methods were better than the single best individual classifiers. DT with the four remaining measures were worse than the single best classifier but outperformed both the linear and quadratic discriminant classifiers, which were ranked in that order.

Xu et al.’s [116] study into combination methods, compared several versions of Bayesian, voting and Dempster-Schafer approaches on a handwritten digit recognition problem, taken from a US zip-code database. The database consists of 2000 examples and there

were two experiment formats: the first used the whole 2000 examples and prior knowledge, the second used the first 1000 examples for training the combination method and the second 1000 examples for testing the combined ensemble. Their Bayesian approach consists of averaging the estimated posterior probabilities and so they call it the averaged Bayes classifier. They also use the confusion matrices of their trained classifiers as prior knowledge to take the individuals errors of the classifiers into consideration. They found that this approach performed very well for the first format when they had the entire 2000 examples on which to base the confusion matrices as well as for testing, but was unstable when using the second format of 1000 examples for deriving the confusion matrices and 1000 examples for testing. The Dempster-Schafer approach was more robust than the averaged Bayesian approach as was the voting approach but the former performed better than the latter.

Verikas et al. [111] studied twelve combination methods' performances on four datasets from the ELENA project: clouds, concentric, satimage, phoneme. They were interested in how methods from fuzzy logic compared with more widely used combination methods. They studied majority vote, average, Xu's averaged Bayes (as described above), Borda count, weighted average, linear order statistics, fuzzy integral using the Choquet integral, optimised Choquet integral, fuzzy integral with data-dependent weights, weighted average with data-dependent weights, BADD defuzzification strategy and Zimmermann's compensatory operator.

They found that weighted average with data-dependent weights was the best combination method overall closely followed by the fuzzy Choquet integral with data-dependent weights. BADD was one of the best methods for large training sets but could not cope with small training sizes. Optimised Choquet integral was found to be better than basic Choquet integral. Majority vote, Borda count and Bayes were found to be of little use especially for highly correlated networks exhibiting widely varying accuracies. In particular they were poor when there was a network which was significantly better than all of the others.

Fumera and Roli carried out experiments comparing simple average and weighted average combining rules [35]. They discovered that when an ensemble is fairly balanced, with classifiers having similar performance and correlation, simple average has better performance. For im-balanced ensembles where there is a wider variation among the classifiers' performance and correlation, weighted average is superior. They found in experiments that in practice the weight of the worst classifiers is very close to zero. This means they are almost discarded from having any affect on the weighted average. If the optimal weights are significantly greater than zero the advantage of using weighted averaging over that of using simple average is relatively small.

These empirical studies show the problem for ensemble designers in deciding what base

classifiers to build the ensemble from and what combination method to use to combine their outputs. Clearly there is no one, best choice of combination method, since for each of the datasets the various combination methods perform differently. The difficulty of implementation, computer time required and training set size required must be considered as well as the ensemble accuracy, with some kind of trade-off being unavoidable.

2.6 Experimental set-up to investigate the combination methods

Here we are interested in comparing the accuracies of some of the commonly used combination methods to the single best individual classifier and to each other. We aim to examine the product moment correlation between the outputs from each of the classifier combination methods. We also intend to run a clustering program to identify whether any of the methods are behaving similarly to each other. Using this information we hope to be able to identify which combination methods to use on the basis of both accuracy and ease of implementation. If we have two methods which produce similar results and are clustered closely together it will make sense to use the one which is easier to implement. We also hope to learn more about how these methods may be related to each other despite having derived from a wide variety of sources. To investigate these areas we carried out an experimental study [99]. We used two databases both taken from the UCI Repository of Machine Learning Database¹. They are the Wisconsin Breast Cancer Database (wbc)² and the Pima Indian Diabetes Database (Pima).

From the original 30 features for the wbc data we used the first 10 so that we could run an exhaustive experiment with all possible partitions. We chose the first ten because the features in this data set were logically grouped into 1–10, 11–20, 21–30. The wbc data has 569 objects, 2 classes and 10 features and is trained using a hold-out (random halves) method. The Pima data has 768 objects, 2 classes and 8 features and is trained using ten-fold cross-validation.

All partitions of the 10 features into 3 subsets of the form 4,3,3 (4200) and 4,4,2 (3150) were generated so that the first classifier has 4 features as input, the second classifier has 3(4) features as input and the third classifier has 3(2) features as input. For each partition we designed one ensemble of three linear classifiers and one ensemble of three quadratic classifiers. Thus, the total number of ensembles for the wbc data is twice the total number of partitions. This set-up and the combination methods we are examining were initially used by Kuncheva and Whitaker in [59].

¹available at <http://www.ics.uci.edu/~mllearn/MLRepository.html>

²Created by Dr. William H. Holberg, W. Nick Street and Olvi L. Mangasarian, University of Wisconsin

For the Pima data we took all partitions of the form 3,3,2 using 10-fold cross-validation to obtain a total of 560 ensembles.

Table 2.6 shows a summary of the data sets and the initial experimental protocol, the data sets are described in more detail in Appendices B.8 and B.9.

Table 2.6: SUMMARY OF THE DATA SETS AND THE EXPERIMENTS

Name	c	N	n	(n_1, n_2, n_3)	Total number of ensembles	Training/ Testing
Wisconsin Breast Cancer	2	569	10	(4,4,2)	6300	Hold-out (Random halves)
				(4,3,3)	8400	
Pima Indian Diabetes:	2	768	8	(3,3,2)	560	10-fold cross-validation

Key

c : number of classes N : number of objects in the data set n : number of features used
 (n_1, n_2, n_3) : partition sizes; D_1 uses n_1 of the n features, D_2 uses n_2 , and D_3 uses n_3 features.

We consider:

1. The overall accuracies of the combination methods and their improvement over the single best classifier. For each partition we select the most accurate of the three classifiers based on the training data and this is then considered the single best classifier for that partition.
2. The correlation between each method of combination and all other methods of combination.

The correlation coefficient used was Pearson's Product Moment correlation coefficient. The following combination methods were applied:

- majority vote
- naive Bayes
- behaviour-knowledge space
- Wernecke's method
- maximum
- minimum
- average
- product

- decision templates
- and oracle to compare the performance of the other methods.

2.7 Combination Method Results

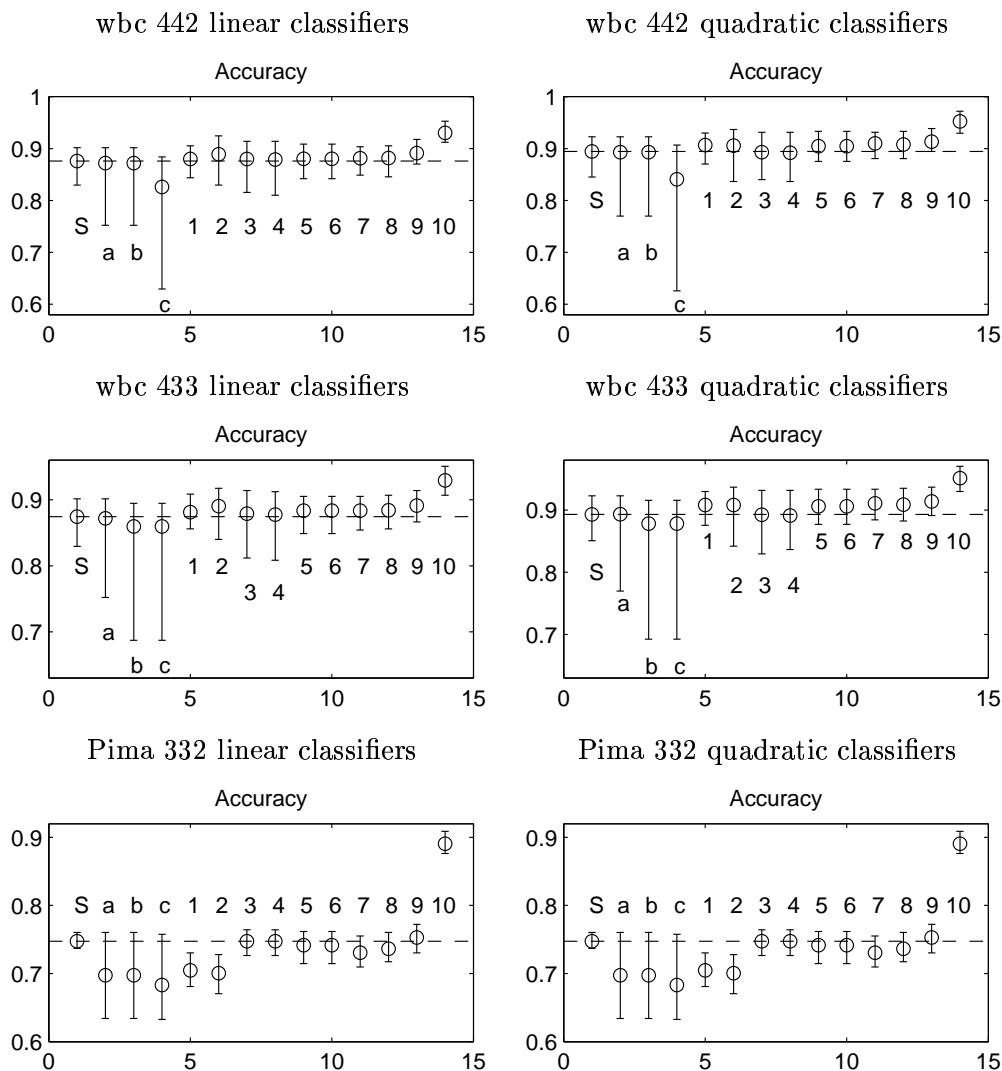
2.7.1 Overall Accuracies

In order to compare the performances of the various combination methods, Figure 2.3 shows the accuracies of the single best classifier, the three individual classifiers, the nine combination methods and the oracle. Figure 2.4 shows the percentage improvement (or decrease) in accuracy of the nine combination methods over the mean single best classifier accuracy. The circles indicate the mean accuracy and the upper and lower lines indicate the minimum and maximum accuracy values. For each partition of the features we have three different classifiers. They are built using the training data and then their accuracy is compared using the testing data as is that of the combination methods combining the three classifiers. We determine the single best classifier by examining the accuracy of the three classifiers on the *training* data. The classifier with the highest accuracy on a particular partition is called the single best classifier *for that particular partition*. The testing accuracy for this single best classifier is then used for that particular partition as a baseline to compare the accuracies of the combination methods. The single best classifier is determined in this way as choosing the best individual classifier on the basis of the testing data would make it even harder for the combination methods to compete as we would be using a form of hindsight which we would not have in a real-world situation. The dashed horizontal line is the mean accuracy of the single best classifier.

The graphs show that the classifiers which receive more information in the form of additional features outperform the classifiers which receive fewer features as input, i.e., for wbc with the partitions 4,4,2 the third classifiers is weakest as we would expect and for 4,3,3 the first classifier is strongest. For the Pima partitions 3,3,2 there is not much difference between the three classifiers.

For the Wisconsin breast cancer data it seems better to choose any of the combination methods rather than a single individual classifier even if it is the strongest one. If we pick the single best classifier for each particular problem based on the training accuracy, we can get better results than some of the combination methods, namely *BKS* and *WER* for linear classifiers and *NB*, *BKS* and *WER* for quadratic classifiers as these all have lower minimum values than the single best classifier. We must also note though, that apart from *BKS* and *WER* with quadratic classifiers, they all have mean value above that of the single best.

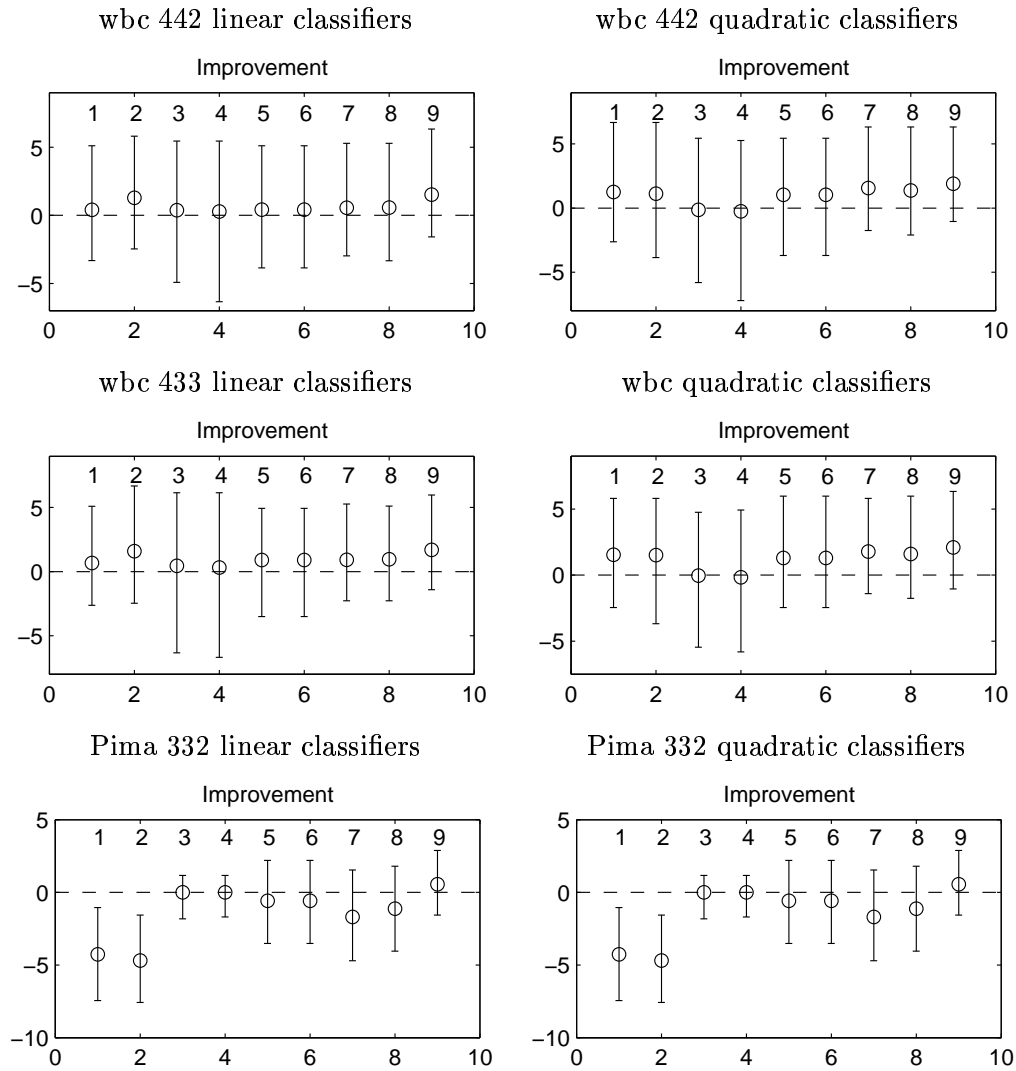
For the Pima data the results are not so good, there is very little improvement over the



S - Single best a - D_1 b - D_2 c - D_3 1 - MAJ 2 - NB 3 - BKS
 4 - WER 5 - MAX 6 - MIN 7 - AVR 8 - PRO 9 - DT 10 - ORA

Figure 2.3: ACCURACY ON THE TESTING SET FOR THE INDIVIDUAL CLASSIFIERS AND THE ENSEMBLES.

single best with only decision templates having a mean accuracy above that of the single best classifier. Majority vote and naive Bayes are clearly under-performing. It is possible that this is due to the fact that Pima has some outliers which we were not aware of at the time of our experiments. The single best classifier is much better than any of the individual classifiers so it is not preferable to pick one individual classifier. If we consider the three individual classifiers' performances and compare them with the combination methods we see that the combination methods (other than MAJ or NB) are more accurate. Choosing the single best classifier is on a par with most of the combination methods and better than some. With linear and quadratic classifiers BKS and WER are on a par with the single



1 - *MAJ* 2 - *NB* 3 - *BKS* 4 - *WER* 5 - *MAX*
 6 - *MIN* 7 - *AVR* 8 - *PRO* 9 - *DT*

Figure 2.4: IMPROVEMENT ON THE TESTING SET FOR THE INDIVIDUAL CLASSIFIERS AND THE ENSEMBLE.

best classifier with *DT* being slightly better. The remaining combination methods *MAX*, *MIN*, *AVR* and *PRO* are all worse than the single best.

These results show that whilst it is possible to improve upon the accuracy of an individual classifier, the success of different combination methods varies with the data set and the type of classifier used. This highlights the difficulty for the practitioner of the problem specific nature of optimum solutions.

2.7.2 Relationships among the combination methods

We were interested in how the different combination methods might be related to each other. In order to investigate these relationships we calculated the correlation between their outputs and illustrated them as shown in Figure 2.5. The intensity of the colour is determined by the correlation. The stronger the correlation the lighter the colour. We found that the combination methods show only positive correlation amongst themselves as we would expect as they are all attempting to improve accuracy.

To further aid analysis of these relationships we used a clustering program to illustrate the strength of the relationships between the different combination methods. Figure 2.6 shows the dendrograms formed when we cluster the combination methods using average-linkage relational clustering³. The lower the branches joining the different combination methods the stronger the relationship between them.

Table 2.7: THE CORRELATION COEFFICIENTS BETWEEN THE DIFFERENT COMBINATION METHODS FOR BOTH TYPES OF CLASSIFIER AND FOR ALL PARTITIONS. BOLD VALUES ARE THOSE WITH ABSOLUTE VALUES OF 0.5 OR GREATER.

	<i>NB</i>	<i>BKS</i>	<i>WER</i>	<i>MAX</i>	<i>MIN</i>	<i>AVR</i>	<i>PRO</i>	<i>DT</i>	<i>ORA</i>
<i>MAJ</i>	0.8665	0.5290	0.4871	0.8240	0.8240	0.9474	0.9098	0.9046	0.7507
<i>NB</i>	1.0000	0.5167	0.4649	0.7280	0.7280	0.8157	0.7800	0.7794	0.5685
<i>BKS</i>		1.0000	0.9163	0.4158	0.4158	0.4973	0.4681	0.5262	0.4291
<i>WER</i>			1.0000	0.3751	0.3751	0.4586	0.4287	0.4868	0.4043
<i>MAX</i>				1.0000	1.0000	0.9018	0.9398	0.8676	0.6618
<i>MIN</i>					1.0000	0.9018	0.9398	0.8676	0.6618
<i>AVR</i>						1.0000	0.9700	0.9375	0.7628
<i>PRO</i>							1.0000	0.9232	0.7611
<i>DT</i>								1.0000	0.7832

Table 2.7 shows the correlation coefficient values between all of the combination methods for all of the results pooled, i.e., with both types of classifier and all partitions. The values show that all of the methods have a positive correlation with each other and several have very high correlation.

Clearly examining the shading graphs, the dendrograms and the correlation table we can see that *BKS* and *WER* are in a group entirely on their own and behaviour-knowledge space is highly positively correlated with Wernecke’s method, which can be expected, knowing that Wernecke’s method is a “regularised” version of *BKS*. We also see that

³The clustering routine and the dendrogram drawing routine are from the package PRTOOLS for Matlab [26]

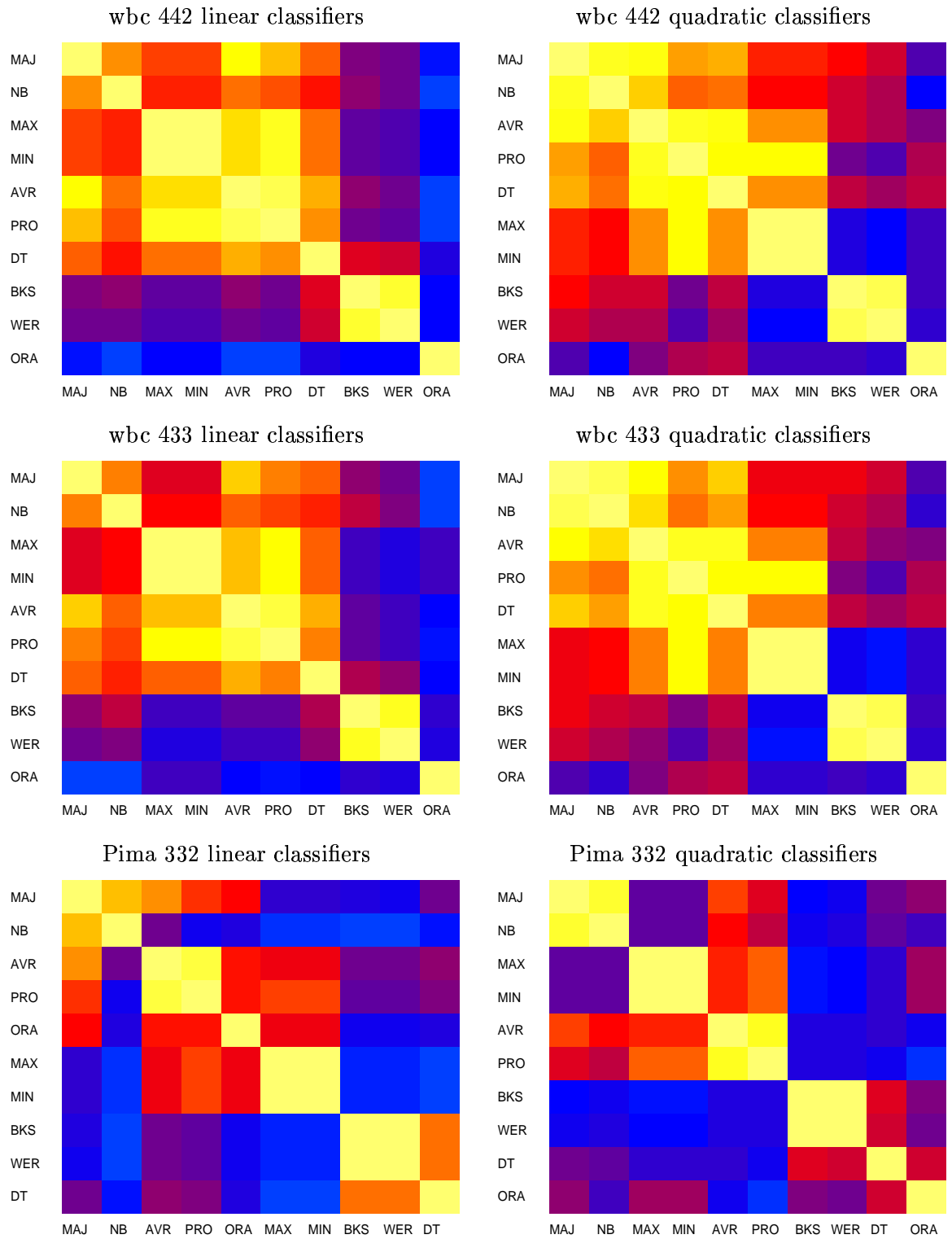


Figure 2.5: ILLUSTRATION OF THE CORRELATION BETWEEN THE COMBINATION METHODS.

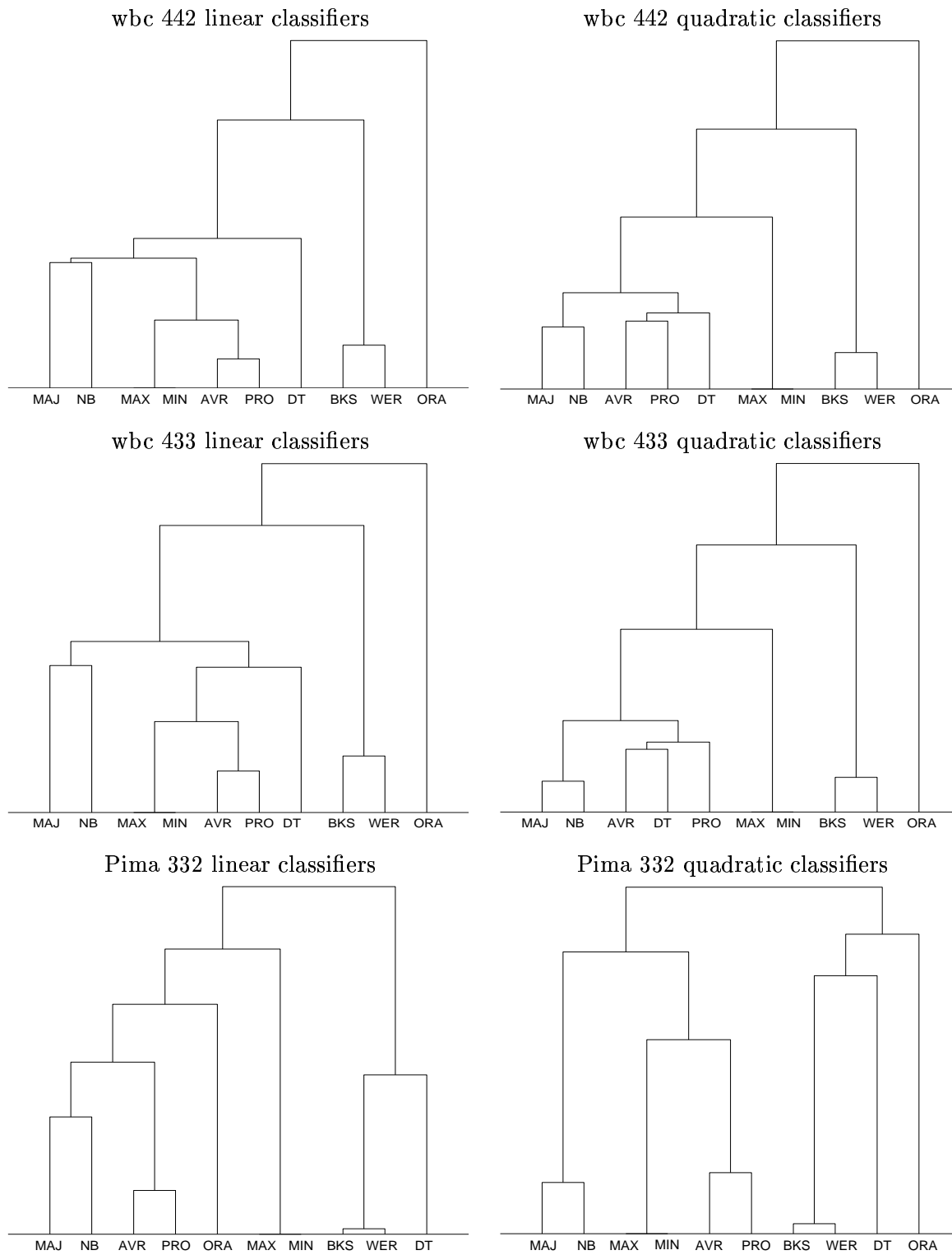


Figure 2.6: THE CLUSTER DENDROGRAMS FOR THE COMBINATION METHODS.

MAX and *MIN* are in fact identical for this case (for the case of two classes, it can be proved that maximum is always equivalent to minimum, proof in Appendix A.1). Average

is highly positively correlated with product. *MAJ* and *NB* are also grouped together. *DT* is not consistently correlated with any of the other combination methods since it is related to *AVR* and *PRO* for the breast cancer data with quadratic classifiers, *BKS* and *WER* for Pima data and is fairly isolated for breast cancer data with linear classifiers. The overall correlation between the combination methods with the Pima data was lower hence there are darker shades in the correlation pictures.

The dendrograms suggest the following grouping of the methods $\{(MAJ, NB), (AVR, PRO), (MIN, MAX), (BKS, WER), (DT), (ORA)\}$. These groupings are also supported by the correlation coefficient diagrams in Figure 2.5 as they show up as lighter coloured areas consistently in all of the partitions and both datasets.

2.8 Combination Methods Conclusions

In this chapter we introduced nine combination methods and the oracle and studied the relationships between them. We took a breast cancer data-set of 10 feature values for 569 patients and using all partitions of the form (4,4,2) and (4,3,3) for two types of classifier (linear and quadratic), conducted a set of four enumerative experiments. We also took a diabetes data set of 8 feature values for 768 patients and conducted a set of ten-fold cross-validation experiments using all possible partitions of the form (3,3,2).

We then considered the overall accuracies of the combination methods and their improvement over the single best classifier. In the next chapter we introduce the concept of diversity (differences amongst classifiers) and show that the classifiers used here were not very diverse and this is probably why the combination methods did not improve notably over the single best classifier.

We were interested in whether any of the combination methods performed in a similar way to each other and hoped that this would be shown by the clustering program. We anticipated that those combination methods which operated in a similar manner would consistently be clustered together early on and that those with different performances would end up in different clusters. We found some interesting correlation amongst the combination methods. In particular maximum is equivalent to minimum for the two class case. We also found that average has close relationship with fellow simple combination method product. Unsurprisingly behaviour-knowledge space is correlated with Wernecke's method as they both originate from the same concept but more interestingly majority vote is correlated strongly with naive Bayes which does not seem to have much in common with it in the computation process. Decision templates were found to have lower correlation with all the other methods.

Chapter 3

Diversity in Classifier ensembles

When we have a group of classifiers at our disposal it is intuitively accepted that the classifiers to be combined should be different from each other, or *diverse* [17, 18, 37, 38, 48, 91]. The experimental results in chapter 2 show that sometimes it is not particularly beneficial to combine a set of classifiers, this can be caused by a lack of diversity amongst the ensemble of classifiers. If once again, we recall the analogy of the respiratory patient (Chapter 1.2), it obviously would be of no benefit to the patient if all the nurses, doctors and consultants always agreed. In that case we would only ever need one nurse, which whilst saving some money, would not improve the accurate diagnosis rate. By having differing opinions we can pool the various diagnoses and hopefully come up with correct one more often.

Clearly, a set of identical classifiers does not gain us any advantage over having just one of them. Therefore, diversity, also related to negative dependence, independence, orthogonality, complementarity, among an ensemble of classifiers has been recognised as a key issue [18, 65]. In fact, Cunningham and Carney claim that “any work with classification ensembles should explicitly measure diversity in the ensemble and use this measure to guide decisions on the constitution of the ensemble...” [18]. Giacinto and Roli believe that there is a fundamental need for methods aimed at designing accurate and diverse classifiers and that this is currently acknowledged in the field [37]. In a later study, Cunningham goes further and says that over-fitting can be used to provide this diversity within an ensemble, provided there is variety amongst the over-fitted members [17]. He says that several over-fitted classifiers can be combined to average out their over-fitting and give an accurate performance provided there is sufficient diversity amongst the classifiers.

Theoretically, a group of independent classifiers improve upon the single best classifier when majority vote combination is used. Using a set of dependent classifiers may result in either better performance than the independent set’s performance or worse performance than the single worst member of the team, depending on the differences. Thus diversity

can be both beneficial or harmful [40, 63]. Understanding and measuring these differences in diversity is an important issue in classifier combination [65] and there are several different measures of diversity being used. These measures aim to quantify the dependence between classifiers. This chapter considers some of the diversity measures available to the practitioner of statistical pattern recognition and investigates how they are related to each other. We are also interested in whether there is any connection between combination accuracy and diversity in the ensemble. In a previous study we investigated the relationship between the Q -statistic and majority vote [64]. We proved that there is a functional relationship between the Q -statistic and the upper and the lower limits of the *majority vote* accuracy. However, there is no theoretical proof of any relationship in the general case. Thus, in this chapter we investigate in detail whether there is any relationship between the diversity measures and the combination methods introduced in the previous chapter. We expect that those instances where the combination methods had high accuracy will be positively correlated to a high level of diversity amongst the ensemble's constituent classifiers.

3.1 Measures of diversity

There are different diversity measures available from different fields of research. Some of these measures, such as the Q -statistic and the correlation coefficient have come directly from mainstream statistics whilst others have developed through the field of statistical pattern recognition, specifically for the problems of multiple classifier systems. Some of these measures work on the whole group of L classifiers whilst other measures consider the classifiers on a pairwise basis and then average the results. We have examined ten measures of diversity which we have used in previous studies. We can consider the measures of diversity to be one of two types :

1. measures looking for diversity: the higher the value the more diverse (\uparrow).
2. measures looking for similarity: the higher the value the less diverse (\downarrow).

The measures we have considered are as follows:

pairwise

The Q -statistic (Q), (\downarrow)	The correlation coefficient (ρ), (\downarrow)
The disagreement measure (D), (\uparrow)	The double-fault measure (DF), (\downarrow)

non-pairwise

The Kohavi-Wolpert variance (kw), (\uparrow)	The measurement of interrater agreement (κ), (\downarrow)
The entropy measure (Ent), (\uparrow)	The measure of difficulty (θ), (\downarrow)
The generalised diversity (GD), (\uparrow)	The coincident failure diversity (CFD), (\uparrow)

3.1.1 Pairwise Diversity Measures

The following diversity measures require consideration of the diversity between each of the pairs of classifiers and then averaging of the values. Consider two classifiers, D_i and D_k , and a 2×2 table that summarises their outputs as shown in Table 3.1. The entries in the table are the probabilities for the respective pair of correct/incorrect outputs. For example, the value of b in the table is the proportion of examples which are correctly classified by classifier D_i and misclassified by classifier D_k .

Table 3.1: THE 2×2 RELATIONSHIP TABLE WITH PROBABILITIES

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	a	b
D_i wrong (0)	c	d

$$\text{Total, } a + b + c + d = 1$$

Table 3.2 illustrates the case of identical (ID), independent (IND) and negatively dependent (ND) pairs of classifiers.

Table 3.2: THE 2×2 RELATIONSHIP TABLES FOR IDENTICAL, INDEPENDENT AND NEGATIVELY DEPENDENT CLASSIFIERS

Identical			Independent			Negatively Dependent		
	$D_2(1)$	$D_2(0)$		$D_2(1)$	$D_2(0)$		$D_2(1)$	$D_2(0)$
$D_1(1)$	0.6	0	$D_1(1)$	0.3	0.2	$D_1(1)$	0	0.5
$D_1(0)$	0	0.4	$D_1(0)$	0.3	0.2	$D_1(0)$	0.5	0

Using real-world data we are unlikely to find a set of highly, negatively dependent classifiers which are also more accurate than random guessing. The most likely situation is an accurate but also highly, positively dependent ensemble of classifiers. From the point of view of a user looking for diverse classifiers the best we could hope for in the real-world case is a slight negative dependence (SND) between a pair of classifiers, (Table 3.3). There are various statistics to assess the similarity/diversity of two classifier outputs.

The Q -statistic (Q) [118]

Yule's Q statistic for two classifiers, e.g., D_i and D_k , is

$$Q_{i,k} = \begin{cases} \frac{ad-bc}{ad+bc}, & \text{if } a, b, c, d < 1 \\ 1, & \text{otherwise} \end{cases} \quad (3.1)$$

Table 3.3: THE 2×2 RELATIONSHIP TABLE FOR SLIGHTLY NEGATIVE DEPENDENT CLASSIFIERS

	D_2 correct (1)	D_2 wrong (0)
D_1 correct (1)	0.30	0.27
D_1 wrong (0)	0.23	0.20

There are several special cases taken into account here. If $a = 1$ then both classifiers are identical but the denominator would be zero so that is why we have the ‘otherwise’ case. If $b = 1$, $c = 1$ or $d = 1$ this implies that the accuracy of one or both classifiers is zero and so it would not be used in an ensemble anyway. In our code only classifiers better than random guessing are used and if the denominator was zero then the value of Q would be 1. For statistically *independent* classifiers, $Q_{i,k} = 0$. Q varies between -1 and +1, with the lower the value the more diverse the classifiers. For a set of L classifiers, the averaged Q statistics of all pairs is taken. From the examples we can calculate the corresponding values of Q :

$$\begin{aligned}
Q_{ID} &= \frac{0.6 \times 0.4 - 0}{0.6 \times 0.4 + 0} = \frac{0.24}{0.24} = 1 \\
Q_{IND} &= \frac{0.3 \times 0.2 - 0.2 \times 0.3}{0.3 \times 0.2 + 0.2 \times 0.3} = 0 \\
Q_{ND} &= \frac{0 \times 0 - 0.5 \times 0.5}{0 \times 0 + 0.5 \times 0.5} = \frac{-0.25}{0.25} = -1 \\
Q_{SND} &= \frac{0.30 \times 0.20 - 0.27 \times 0.23}{0.30 \times 0.20 + 0.27 \times 0.23} = \frac{-0.0021}{0.1221} = -0.0172
\end{aligned}$$

The Correlation coefficient (ρ)

The correlation coefficient, ρ is well known in mainstream statistics. The correlation between two binary classifier outputs (correct/incorrect) is

$$\rho_{i,k} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}. \quad (3.2)$$

For the examples the corresponding values of ρ are:

$$\begin{aligned}
\rho_{ID} &= \frac{0.6 \times 0.4 - 0}{\sqrt{(0.6+0)(0+0.4)(0.6+0)(0+0.4)}} = \frac{0.24}{0.24} = 1 \\
\rho_{IND} &= \frac{0.3 \times 0.2 - 0.2 \times 0.3}{\sqrt{(0.3+0.2)(0.3+0.2)(0.3+0.3)(0.2+0.2)}} = 0 \\
\rho_{ND} &= \frac{0 \times 0 - 0.5 \times 0.5}{\sqrt{(0+0.5)(0.5+0)(0+0.5)(0.5+0)}} = \frac{-0.25}{0.25} = -1 \\
\rho_{SND} &= \frac{0.30 \times 0.20 - 0.27 \times 0.23}{\sqrt{(0.3+0.27)(0.23+0.2)(0.3+0.23)(0.27+0.2)}} = \frac{-0.0021}{0.247} = -0.0085
\end{aligned}$$

As Q the correlation coefficient can also take negative values with the lower the value the more diverse the classifiers, and in fact,

Proposition 1 *For any two classifiers, Q and ρ have the same sign, and:*

$$|\rho| = \left| \frac{ad-bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \right| \leq \left| \frac{ad-bc}{ad+bc} \right| = |Q|.$$

Proof

Since the numerator is the same for Q and ρ we have to compare the denominators.

$$\begin{aligned} \left| \left(\sqrt{(a+b)(c+d)(a+c)(b+d)} \right)^2 \right| &= |(a+b)(c+d)(a+c)(b+d)| \\ &= |a^2bc + a^2cd + abc^2 + ac^2d + a^2bd + \\ &\quad acd^2 + ab^2c + bc^2d + ab^2d + abd^2 + \\ &\quad b^2cd + bcd^2 + (a^2d^2 + 2abcd + b^2c^2)| \\ &\geq |a^2d^2 + 2abcd + b^2c^2| \\ &= |(ad+bc)^2| \end{aligned}$$

This implies that

$$\begin{aligned} \left| \frac{1}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \right| &\leq \left| \frac{1}{ad+bc} \right| \\ \left| \frac{ad-bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \right| &\leq \left| \frac{ad-bc}{ad+bc} \right| \\ |\rho| &\leq |Q| \end{aligned}$$

■

The disagreement measure (D) [41,100]

The disagreement measure, D is used in [41,100] and is given by

$$D_{i,k} = b + c. \tag{3.3}$$

Thus, it is simply the total proportion of examples for which the two classifiers disagree.

So for the identical, independent, and negative dependent cases we obtain:

$$\begin{aligned} D_{ID} &= 0 \\ D_{IND} &= 0.5 \\ D_{ND} &= 1 \\ D_{SND} &= 0.5 \end{aligned}$$

For the disagreement method the range of possible values is from 0 to 1 and the higher the value the more diverse the classifiers.

The double-fault measure (DF) [38]

The double-fault measure, DF is used in [38] and is given by

$$DF_{i,k} = d. \quad (3.4)$$

Thus, it is the proportion of classifiers which both classifiers fail to correctly classify. In this way it is, to some extent, a measure of similarity. The values of DF corresponding to the examples are:

$$\begin{aligned} DF_{ID} &= 0.4 \\ DF_{IND} &= 0.2 \\ DF_{ND} &= 0 \\ DF_{SND} &= 0.2 \end{aligned}$$

The double-fault measure takes values of between 0 and 1, with the lower the value the more diverse the classifiers [38]. In practice 1 only occurs when both are 100% incorrect so in practice the highest value we would expect to obtain is $1 - p$.

3.1.2 Non-pairwise Diversity Measures

For the non-pairwise measures we quote the formulae for L classifiers. Let $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ be a labelled data set, $\mathbf{z}_j \in \mathfrak{R}^n$ coming from the classification problem in question. We can represent the output of a classifier D_i as an N -dimensional binary vector $\pi_i = [\pi_{1,i}, \dots, \pi_{N,i}]^T$, such that $\pi_{j,i} = 1$, if D_i recognises correctly \mathbf{z}_j , and 0, otherwise, $i = 1, \dots, L$. In order to demonstrate how the following diversity measures work we use a small example of $L = 3$ classifiers operating on a labelled data set Z of size $N = 10$. The values in Table 3.4 columns 2-4 are the $\pi_{j,i}$ mentioned above.

The example has a degree of independence between classifiers D_1 and D_2 , a degree of positive dependence between D_2 and D_3 and some negative dependence between D_1 and D_3 . It is used to illustrate how we calculate each of the measures of diversity and to show how the range of values from negative to positive dependence varies amongst them. For these classifiers $Q_{1,2} = 0$, $Q_{2,3} = 0.71$ and $Q_{1,3} = -0.5$, i.e. D_1 and D_2 are independent, D_2 and D_3 are positively dependent and D_1 and D_3 are negatively dependent. Combining them gives an average value of $Q = 0.07$ suggesting fairly independent classifiers. This example will be used to illustrate how all the following diversity measures are calculated and it will be referred to as ‘the Example in Table 3.4’.

The Kohavi-Wolpert variance (KW) [50]

Kohavi and Wolpert introduce this measure in [50]. They give a formula for the error rate of a classifier, showing the variability of the predicted class label b for training object, \mathbf{x}

Table 3.4: AN EXAMPLE OF CLASSIFIERS FOR THE NON-PAIRWISE DIVERSITY MEASURES

Train Set Z	Classifiers			kw, κ			Ent
	D_1	D_2	D_3	$\overbrace{l(\mathbf{z}_j) \quad L - l(\mathbf{z}_j) \quad l(\mathbf{z}_j)(L - l(\mathbf{z}_j))}^{}$			$\overbrace{\min\{l(\mathbf{z}_j), (L - l(\mathbf{z}_j))\}}^{}$
z_1	1	0	0	1	2	2	1
z_2	1	0	0	1	2	2	1
z_3	1	1	1	3	0	0	0
z_4	0	1	1	2	1	2	1
z_5	1	1	1	3	0	0	0
z_6	1	1	0	2	1	2	1
z_7	0	0	1	1	2	2	1
z_8	1	0	1	2	1	2	1
z_9	0	1	1	2	1	2	1
z_{10}	0	0	0	0	3	0	0
Accuracy:	0.6	0.5	0.6		Totals:	14	7

The corresponding pairwise tables are:

	$D_2(1)$	$D_2(0)$		$D_3(1)$	$D_3(0)$		$D_3(1)$	$D_3(0)$
$D_1(1)$	0.3	0.3	$D_2(1)$	0.4	0.1	$D_1(1)$	0.3	0.3
$D_1(0)$	0.2	0.2	$D_2(0)$	0.2	0.3	$D_1(0)$	0.3	0.1

as

$$variance_x = \frac{1}{2} \left(1 - \sum_{i=1}^c P(b = \omega_i | \mathbf{x})^2 \right), \quad (3.5)$$

where $P(b = \omega_i | \mathbf{x})$ is estimated as an average over different data sets. We adapt their idea by looking at the variability of b for \mathbf{x} using the classifier models D_1, \dots, D_L . Instead of considering the class labels in Ω , we consider oracle-type outputs, that is two possible classifier outputs: correct (1) and incorrect (0). Thus, $P(b = 1 | \mathbf{x})$ and $P(b = 0 | \mathbf{x})$ will be obtained as an average over the set of classifiers, \mathcal{D} . If we denote by $l(\mathbf{z}_j)$ the number of classifiers from \mathcal{D} that correctly recognise \mathbf{z}_j , i.e., $l(\mathbf{z}_j) = \sum_{i=1}^L \pi_{j,i}$ we obtain:

$$P(b = 1 | \mathbf{x}) = \frac{l(\mathbf{x})}{L} \quad \text{and} \quad P(b = 0 | \mathbf{x}) = \frac{L - l(\mathbf{x})}{L}. \quad (3.6)$$

Substituting (3.6) into (3.5),

$$variance_x = \frac{1}{2} \left(1 - P(b = 1 | \mathbf{x})^2 - P(b = 0 | \mathbf{x})^2 \right), \quad (3.7)$$

and averaging over the whole of the training set Z , we obtain the kw measure of diversity as

$$kw = \frac{1}{NL^2} \sum_{j=1}^N l(\mathbf{z}_j)(L - l(\mathbf{z}_j)). \quad (3.8)$$

Using the example in Table 3.4 and the values calculated in the 5th, 6th and 7th columns (labelled kw) we obtain:

$$\begin{aligned} kw &= \frac{1}{10 \times 9} \sum_{j=1}^{10} l(\mathbf{z}_j)(3 - l(\mathbf{z}_j)) \\ &= \frac{1}{90} \times 14 = \frac{14}{90} = 0.1\dot{5} \end{aligned}$$

The higher the value of the Kohavi-Wolpert variance the more diverse the classifiers.

The measurement of interrater agreement (κ) [28]

If we denote \bar{p} to be the average individual classification accuracy in the ensemble, then

$$\kappa = 1 - \frac{\frac{1}{L} \sum_{j=1}^N l(\mathbf{z}_j)(L - l(\mathbf{z}_j))}{N(L - 1)\bar{p}(1 - \bar{p})} \quad (3.9)$$

and so κ can be shown to be related to kw and D as follows

$$\kappa = 1 - \frac{L}{(L - 1)\bar{p}(1 - \bar{p})} \quad kw = 1 - \frac{1}{2\bar{p}(1 - \bar{p})}D. \quad (3.10)$$

The value of κ for the Example in Table 3.4 requires the average \bar{p} which is $\frac{0.5+0.6+0.6}{3} = 0.5\dot{6}$, with this we obtain:

$$\begin{aligned} \kappa &= 1 - \frac{\frac{1}{3} \times 14}{1 - \times 2 \times 0.5\dot{6} \times 0.4\dot{3}} \\ &= 1 - \frac{4.\dot{6}}{4.9\dot{1}} = 1 - 0.950 = 0.0498 \end{aligned}$$

κ can take negative values with the lower the value the more diverse the classifiers.

The entropy measure (Ent) [60]

For oracle-type, 0/1 outputs, we can obtain the highest value of diversity amongst a group of classifiers for a particular object, $\mathbf{z}_j \in \mathbf{Z}$, when $\lfloor L/2 \rfloor$ of the votes have one value (1 or 0) and the other $L - \lfloor L/2 \rfloor$ votes have the alternative value (0 or 1). That is, just a majority were correct/incorrect and just less than the majority were incorrect/correct. If they were all 0's or all 1's, there would be no disagreement, and the classifiers could not be deemed diverse. One possible measure of diversity based on this concept is the entropy measure:

$$Ent = \frac{1}{N(L - \lfloor L/2 \rfloor - 1)} \sum_{j=1}^N \min \{l(\mathbf{z}_j), L - l(\mathbf{z}_j)\}. \quad (3.11)$$

Our measure is a non-classical entropy measure because it does not use the logarithm function. For our study into these diversity measures [99] a referee pointed us to a more traditional approach given by Cunningham and Carney [18] (we denote it here as E_{CC}). We compare these two measures to show that our measure is worthwhile. If we take the expectation over the whole feature space, let the number of classifiers $L \rightarrow \inf$, and use again $l(\mathbf{z}_j)$ as the number of 1's (correct outputs) in the team, the two expressions become

$$Ent(l(\mathbf{z}_j)) = \frac{1}{2} \min\{l(\mathbf{z}_j), 1 - l(\mathbf{z}_j)\} \quad \text{and} \quad (3.12)$$

$$E_{CC}(l(\mathbf{z}_j)) = -l(\mathbf{z}_j) \log(l(\mathbf{z}_j)) - (1 - l(\mathbf{z}_j)) \log(1 - l(\mathbf{z}_j)). \quad (3.13)$$

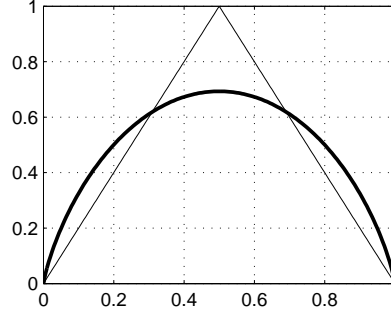


Figure 3.1: THE TWO ENTROPY MEASURES $Ent(l(\mathbf{z}_j))$ (THIN LINE) AND $E_{CC}(l(\mathbf{z}_j))$ (THICK LINE) PLOTTED VERSUS $l(\mathbf{z}_j)$.

Figure 3.1 plots the two entropies versus $\frac{l(\mathbf{z}_j)}{\sum l(\mathbf{z}_j)}$. We can see that the two measures are equivalent up to a (nonlinear) monotonic transformation. This means that they will both have a similar relationship with the ensemble accuracy. As Ent is easier to handle and quicker to calculate, we continue to use it in our experiments.

For the Example in Table 3.4 and using the calculations in the last column (labelled Ent), we can calculate Ent as follows:

$$\begin{aligned} Ent &= \frac{1}{10} \sum_{j=1}^{10} \min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\} \\ &= \frac{1}{10} \times 7 = 0.7 \end{aligned}$$

For Ent the higher the value the more diverse the classifiers.

The measure of difficulty (θ) [39]

For this measure, we define a discrete random variable X which takes values in $\{\frac{0}{L}, \frac{1}{L}, \dots, 1\}$ and denotes the proportion of classifiers in \mathcal{D} that correctly classify an input \mathbf{x} drawn randomly from the distribution of the problem. The measure of difficulty θ is then defined as

$$\theta = Var(X). \quad (3.14)$$

For the Example in Table 3.4 the estimated probability mass function for a variable X with values in $\{\frac{0}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}\}$ is shown in Figure 3.2

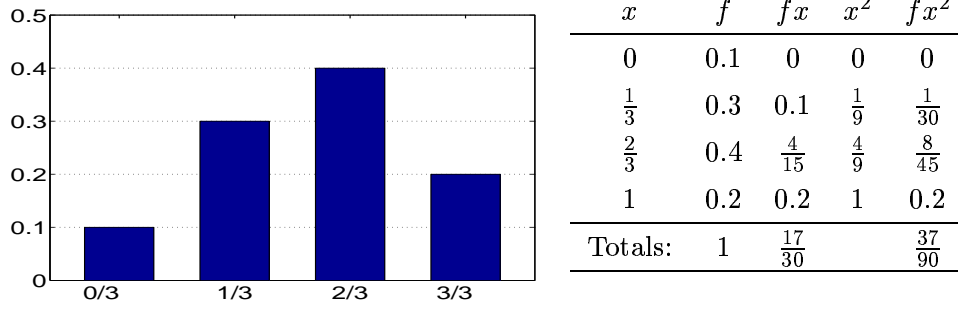


Figure 3.2: AN EXAMPLE OF THE PROBABILITY MASS FUNCTION FOR THE MEASURE OF DIFFICULTY

The variance is then calculated as:

$$\begin{aligned}\sigma^2 &= \frac{\sum fx^2}{\sum f} - \left(\frac{\sum fx}{\sum f} \right)^2 \\ &= \frac{37}{90} - \left(\frac{17}{30} \right)^2 = 0.09\end{aligned}$$

The lower the value of θ , the more diverse the classifier team.

The generalised diversity, (GD) [83], and the coincident failure diversity (CFD) [82]

For this measure of diversity, let Y be a random variable expressing the proportion of classifiers (out of L) that are incorrect on a randomly drawn object $\mathbf{x} \in \mathfrak{R}^n$ and let p_i be the probability that i randomly chosen classifiers are incorrect for a randomly chosen \mathbf{x} , i.e., $p(Y = \frac{i}{L})$. (Note that $Y = 1 - X$, where X is the variable introduced for θ). If we also denote,

$$p(1) = \sum_{i=1}^L \frac{i}{L} p_i, \quad (3.15)$$

$$\text{and } p(2) = \sum_{i=1}^L \frac{i}{L} \frac{(i-1)}{(L-1)} p_i. \quad (3.16)$$

then the generalised diversity measure, GD , is calculated as:

$$GD = 1 - \frac{p(2)}{p(1)}. \quad (3.17)$$

For the Example in Table 3.4, we have $p_0 = 0.2$, $p_1 = 0.4$, $p_2 = 0.3$ and $p_4 = 0.1$ which allows us to calculate GD as,

$$p(1) = \frac{1}{3} \times 0.4 + \frac{2}{3} \times 0.3 + \frac{3}{3} \times 0.1 = 0.43$$

$$\begin{aligned}
p(2) &= \frac{1}{3} \times \frac{0}{2} \times 0.4 + \frac{2}{3} \times \frac{1}{2} \times 0.3 + \frac{3}{3} \times \frac{2}{2} \times 0.1 = 0.2 \\
GD &= 1 - \frac{0.2}{0.43} = 0.538
\end{aligned}$$

For GD the higher the value of the generalised diversity the more diverse the classifiers. The coincident failure diversity, CFD is a modification of GD proposed in [82].

$$CFD = \begin{cases} 0, & p_0 = 1; \\ \frac{1}{1-p_0} \sum_{i=1}^L \frac{L-i}{L-1} p_i, & p_0 < 1 \end{cases} \quad (3.18)$$

Since $p_0 = 0.2 \neq 1$, CFD for the Example in Table 3.4 is,

$$CFD = \frac{1}{0.8} \left(\frac{2}{2} \times 0.4 + \frac{1}{2} \times 0.3 + \frac{0}{2} \times 0.1 \right) = 1.25 \times 0.55 = 0.6875$$

Like the generalised diversity measure, for the coincident failure diversity the higher the value the more diverse the classifiers.

3.2 Limits of the measures

The upper and lower limits for the measures of diversity depend on the number of classifiers, L , and the value of p , the individual classifier accuracy. The limits for the case with two classifiers with equal individual accuracy p have been determined by Kuncheva and Whitaker in [60]. Here we are interested in the situation with three classifiers it is necessary to extend this work with two classifiers to the three-classifier case.

In order to calculate the upper and lower limits for a three classifier case, we must consider the most diverse and least diverse classifier outputs we can have for each particular measure of diversity, for a range of classifier accuracies, p . To simplify the process we assume that all three classifiers have identical accuracy.

3.2.1 The case of Identical Classifiers

The least diverse set of three classifiers we can have is always when all three are identical and the pairwise 2×2 table will be of the following form:

	D_2 correct (1)	D_2 wrong (0)
D_1 correct (1)	p	0
D_1 wrong (0)	0	$1 - p$

So for the measures of diversity we can substitute these values into the formula to get the formula for the identical case.

$$Q_{avID} = \frac{1}{3} \times 3 \begin{cases} \frac{p(1-p)-0}{p(1-p)+0} & \text{if } a, b, c, d < 1 \\ 1, & \text{otherwise} \end{cases}$$

$$\begin{aligned}
&= 1 \quad \forall p \\
\rho_{avID} &= \frac{1}{3} \times 3 \frac{p(1-p) - 0 \times 0}{\sqrt{(p+0)(0+1-p)(p+0)(0+1-p)}} \\
&= \frac{p(1-p)}{\sqrt{p^2(1-p)^2}} = \frac{p(1-p)}{p(1-p)} = 1 \quad \forall p \\
D_{avID} &= \frac{1}{3} \times 3 \times (0+0) = 0 \quad \forall p \\
DF_{avID} &= \frac{1}{3} \times 3 \times (1-p) = 1-p \quad \forall p
\end{aligned}$$

For each of the pairwise tables $a = p$, $b = c = 0$, $d = 1 - p$, and as we are considering three classifiers, $L = 3$. Recall that $l(\mathbf{z}_j)$ is the number of classifiers from \mathcal{D} that correctly recognise \mathbf{z}_j . As the three classifiers are all identical, for p of the \mathbf{z}_j , $l(\mathbf{z}_j) = 3$ and $(3 - l(\mathbf{z}_j)) = 0$ and for the remaining $1 - p$ of them $l(\mathbf{z}_j) = 0$ and $(3 - l(\mathbf{z}_j)) = 3$. So $l(\mathbf{z}_j)(3 - l(\mathbf{z}_j)) = 0$ for all cases. Thus we can calculate the identical case for kw , κ and Ent .

$$kw_{ID} = \frac{1}{9N} \sum_{j=1}^N l(\mathbf{z}_j)(3 - l(\mathbf{z}_j)) = \frac{1}{9N} \times 0 = 0 \quad (3.19)$$

$$\kappa_{ID} = 1 - \frac{\frac{1}{3} \sum_{j=1}^N l(\mathbf{z}_j)(3 - l(\mathbf{z}_j))}{2N\bar{p}(1 - \bar{p})} = 1 - 0 = 1 \quad (3.20)$$

$$Ent_{ID} = \frac{1}{N} \sum_{j=1}^N \min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\} = \frac{1}{10} \times 0 = 0 \quad (3.21)$$

$$(3.22)$$

For the measure of difficulty we need to consider the p.m.f. for the three identical classifiers, shown in Figure 3.3. θ , the variance of X , is then calculated as:

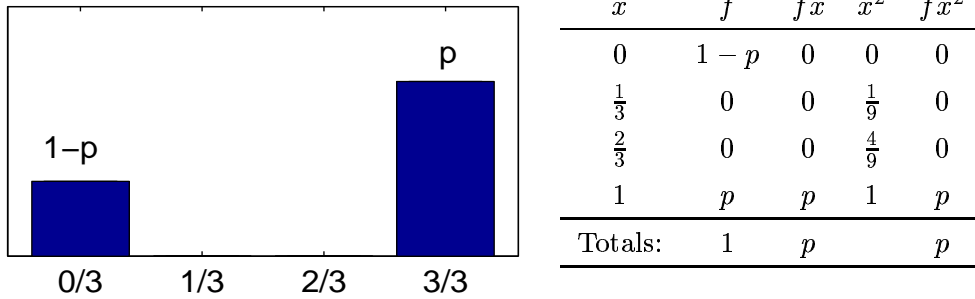


Figure 3.3: THE PROBABILITY MASS FUNCTION FOR IDENTICAL CLASSIFIERS

$$\begin{aligned}
\theta &= \frac{\Sigma fx^2}{\Sigma f} - \left(\frac{\Sigma fx}{\Sigma f} \right)^2 \\
&= \frac{p}{1} - \left(\frac{p}{1} \right)^2 \\
&= p - p^2
\end{aligned} \quad (3.23)$$

For GD and CFD , recall that we are concerned with the probability p_i , that $Y = \frac{i}{L}$, where $Y = 1 - X$, the proportion of classifiers that fail on an object. For the identical case we have :

$$\begin{aligned}
 p_0 &= p, \quad p_1 = 0, \quad p_2 = 0 \text{ and } p_3 = 1 - p \\
 \text{thus } p(1) &= \sum_{i=1}^3 \frac{i}{3} p_i = \frac{1}{3} p_1 + \frac{2}{3} p_2 + \frac{3}{3} p_3 = p_3 = 1 - p, \\
 \text{and } p(2) &= \sum_{i=1}^3 \frac{i(i-1)}{3} p_i = \frac{1}{3} \frac{0}{2} p_1 + \frac{2}{3} \frac{1}{2} p_2 + \frac{3}{3} \frac{2}{2} p_3 = p_3 = 1 - p \\
 \text{so } GD_{ID} &= 1 - \frac{p(2)}{p(1)} = 1 - \frac{1-p}{1-p} = 0
 \end{aligned} \tag{3.24}$$

$$\text{and since } p_0 < 1.0 \tag{3.25}$$

$$CFD_{ID} = \frac{1}{1-p_0} \sum_{i=1}^3 \frac{3-i}{2} p_i = \frac{1}{1-p} \left[\frac{2}{2} p_1 + \frac{1}{2} p_2 + \frac{0}{2} p_3 \right] = 0 \tag{3.26}$$

3.2.2 The case of Highly Diverse Classifiers

The case of the most diverse classifiers is more difficult. The problem must be split into two halves, the case when $0.5 \leq p \leq \frac{2}{3}$ and the case when $\frac{2}{3} \leq p < 1.0$. This is because the possible overlaps of 0/1 outputs are different for either side of $p = \frac{2}{3}$. If we consider the three classifier case with $p = \frac{2}{3}$ the most diverse case is when they each have a different $\frac{1}{3}$ of the data which they misclassify. For $p < \frac{2}{3}$ there is an overlap where two of the three classifiers are incorrect but there is no instance where all three are correct. For $p > \frac{2}{3}$ there are no cases when more than one classifier is incorrect but there are cases when all three are correct. Figures 3.4 illustrates the case when $p < \frac{2}{3}$ and 3.5 illustrates the case when $p > \frac{2}{3}$.

The corresponding pairwise tables for figure 3.4 are:

	$D_2(1)$	$D_2(0)$		$D_3(1)$	$D_3(0)$		$D_3(1)$	$D_3(0)$
$D_1(1)$	$2p - 1$	$1 - p$	$D_2(1)$	$p/2$	$p/2$	$D_1(1)$	$p/2$	$p/2$
$D_1(0)$	$1 - p$	0	$D_2(0)$	$p/2$	$1 - 3p/2$	$D_1(0)$	$p/2$	$1 - 3p/2$

The three-way table can also be determined as:

	$D_2(1), D_2(1)$	$D_2(1), D_2(0)$	$D_2(0), D_2(1)$	$D_2(0), D_2(0)$
$D_1(1)$	0	$2p - 1$	$p/2$	$1 - 3p/2$
$D_1(0)$	$p/2$	$1 - 3p/2$	0	0

D_1	no.	D_2	no.	D_3	no.	kw, κ			Ent
						A	B	$A \times B$	$\min\{A, B\}$
1		0		1		2	1	2	1
\vdots		\vdots		\vdots	$\frac{p}{2}N$	\vdots	\vdots	\vdots	\vdots
\vdots		\vdots		1		2	1	\vdots	\vdots
\vdots		\vdots	$(1-p)N$	0		1	2	\vdots	\vdots
\vdots	pN	\vdots		\vdots		\vdots	\vdots	\vdots	\vdots
\vdots		0		\vdots		1	2	\vdots	\vdots
\vdots		1		\vdots		2	1	\vdots	\vdots
\vdots		\vdots		\vdots	$(1-p)N$	\vdots	\vdots	\vdots	\vdots
1		\vdots		\vdots		2	1	\vdots	\vdots
0		\vdots		\vdots		1	2	\vdots	\vdots
\vdots		\vdots	pN	\vdots		\vdots	\vdots	\vdots	\vdots
\vdots		\vdots		0		1	2	\vdots	\vdots
\vdots	$(1-p)N$	\vdots		1		2	1	\vdots	\vdots
\vdots		\vdots		\vdots	$\frac{p}{2}N$	\vdots	\vdots	\vdots	\vdots
0		1		1		2	1	2	1
N		N		N		Totals:		$2N$	N

where $A = l(\mathbf{z}_j)$ and $B = (3 - l(\mathbf{z}_j))$.

Figure 3.4: THE MOST DIVERSE CLASSIFIERS FOR $p \leq \frac{2}{3}$

We can calculate the p.m.f. of X for θ from this information:

x	f	fx	x^2	fx^2
0	0	0	0	0
$\frac{1}{3}$	$2 - 3p$	$\frac{1}{3}(2 - 3p)$	$\frac{1}{9}$	$\frac{1}{9}(2 - 3p)$
$\frac{2}{3}$	$3p - 1$	$\frac{2}{3}(3p - 1)$	$\frac{4}{9}$	$\frac{4}{9}(3p - 1)$
1	0	0	1	0
Totals:	1	p		$p - \frac{2}{9}$

The corresponding pairwise tables for figure 3.5 are:

	$D_2(1)$	$D_2(0)$		$D_3(1)$	$D_3(0)$		$D_3(1)$	$D_3(0)$
$D_1(1)$	$2p - 1$	$1 - p$	$D_2(1)$	$2p - 1$	$1 - p$	$D_1(1)$	$2p - 1$	$1 - p$
$D_1(0)$	$1 - p$	0	$D_1(0)$	$1 - p$	0	$D_1(0)$	$1 - p$	0

D_1	no.	D_2	no.	D_3	no.	kw, κ			Ent
						A	B	$A \times B$	$\min\{A, B\}$
1		0		1		2	1	2	1
\vdots		\vdots	$(1-p)N$	\vdots		\vdots	\vdots	\vdots	\vdots
\vdots		0		\vdots	$\frac{p}{2}N$	2	1	2	1
\vdots		1		\vdots		3	0	0	0
\vdots		\vdots		\vdots		\vdots	\vdots	\vdots	\vdots
\vdots	pN	\vdots		1		3	0	0	0
\vdots		\vdots		0		2	1	2	1
\vdots		\vdots		\vdots	$(1-p)N$	\vdots	\vdots	\vdots	\vdots
\vdots		\vdots		0		2	1	2	1
\vdots		\vdots	pN	1		3	0	0	0
\vdots		\vdots		\vdots		\vdots	\vdots	\vdots	\vdots
1		\vdots		\vdots		3	0	0	0
0		\vdots		\vdots	$\frac{p}{2}N$	2	1	2	1
\vdots	$(1-p)N$	\vdots		\vdots		\vdots	\vdots	\vdots	\vdots
0		1		1		2	1	2	1
pN		pN		pN		Totals:		$6(1-p)N$	$3(1-p)N$

where $A = l(\mathbf{z}_j)$ and $B = (3 - l(\mathbf{z}_j))$.

Figure 3.5: THE MOST DIVERSE CLASSIFIERS FOR $p \geq \frac{2}{3}$

The three-way table can also be determined as:

	$D_2(1), D_2(1)$	$D_2(1), D_2(0)$	$D_2(0), D_2(1)$	$D_2(0), D_2(0)$
$D_1(1)$	$3p - 2$	$1 - p$	$1 - p$	0
$D_1(0)$	$1 - p$	0	0	0

We can calculate the p.m.f. of X for θ from this information:

x	f	fx	x^2	fx^2
0	0	0	0	0
$\frac{1}{3}$	0	0	$\frac{1}{9}$	0
$\frac{2}{3}$	$3(1-p)$	$2(1-p)$	$\frac{4}{9}$	$\frac{4}{3}(1-p)$
1	$3p - 2$	$3p - 2$	1	$3p - 2$
Totals:	1	p	$\frac{1}{3}(5p - 2)$	

By considering the tables in the two figures 3.4 and 3.5 and in particular the pairwise and three-way tables we can calculate the various diversity measures for the most diverse

case. We denote $\mathcal{M}_{<}$ to be the diversity measure for the case when $p < \frac{2}{3}$ and $\mathcal{M}_{>}$ for the case when $p > \frac{2}{3}$. Also for the pairwise diversity measures we denote $\mathcal{M}_{i,j}$ to be the pairwise diversity between classifier D_i and D_j . Thus, $\mathcal{M}_{<1,2}$ denotes the pairwise diversity between classifier D_1 and D_2 when $p < \frac{2}{3}$. Looking at figure 3.4 gives the information required for $p < \frac{2}{3}$ and figure 3.5 gives the information required for $p > \frac{2}{3}$.

Q for the diverse case

Consider the case for Q when $p < \frac{2}{3}$:

$$Q_{<1,2} = \frac{(2p-1) \times 0 - (1-p)^2}{(2p-1) \times 0 + (1-p)^2} = -1 \quad (3.27)$$

$$Q_{<1,3} = \frac{\frac{p}{2} \times (1 - \frac{3p}{2}) - (\frac{p}{2})^2}{\frac{p}{2} \times (1 - \frac{3p}{2}) + (\frac{p}{2})^2} = \frac{1-2p}{1-p} \quad (3.28)$$

$$Q_{<2,3} = \frac{1-2p}{1-p} \quad (3.29)$$

$$\text{Thus, } Q_{AV<} = \frac{1}{3} \left[1 - + \frac{1-2p}{1-p} + \frac{1-2p}{1-p} \right] = \frac{1-3p}{3(1-p)} \quad (3.30)$$

and similarly the case for Q when $p > \frac{2}{3}$:

$$Q_{>1,2} = Q_{>1,3} = Q_{>2,3} = Q_{<1,2} = -1 \quad (3.31)$$

$$\text{Thus, } Q_{AV>} = -1 \quad (3.32)$$

ρ for the diverse case

Consider the case for ρ when $p < \frac{2}{3}$:

$$\rho_{<1,2} = \frac{(2p-1) \times 0 - (1-p)^2}{\sqrt{(2p-1+1-p)^2(1-p+0)^2}} = \frac{p-1}{p} \quad (3.33)$$

$$\rho_{<1,3} = \frac{\frac{p}{2} - p^2}{\sqrt{p^2(\frac{p}{2} + 1 - \frac{3p}{2})^2}} = \frac{1-2p}{2(1-p)} \quad (3.34)$$

$$\rho_{<2,3} = \frac{1-2p}{2(1-p)} \quad (3.35)$$

$$\text{Thus, } \rho_{AV<} = \frac{1}{3} \left[\frac{p-1}{p} + \frac{1-2p}{2(1-p)} + \frac{1-2p}{2(1-p)} \right] = -\frac{1}{3} \left[\frac{3p^2 - 3p + 1}{p(1-p)} \right] \quad (3.36)$$

and similarly the case for ρ when $p > \frac{2}{3}$:

$$\rho_{>1,2} = \rho_{>1,3} = \rho_{>2,3} = \rho_{<1,2} = \frac{p-1}{p} \quad (3.37)$$

$$\text{Thus, } \rho_{AV>} = \frac{p-1}{p} \quad (3.38)$$

D for the diverse case

Consider the case for D when $p < \frac{2}{3}$:

$$D_{<1,2} = 2(1 - p) \quad (3.39)$$

$$D_{<1,3} = p \quad (3.40)$$

$$D_{<2,3} = p \quad (3.41)$$

$$\text{Thus, } D_{AV<} = \frac{2p + 2 - 2p}{3} = \frac{2}{3} \quad (3.42)$$

and similarly the case for D when $p > \frac{2}{3}$:

$$D_{>1,2} = D_{>1,3} = D_{>2,3} = D_{<1,2} = 2(1 - p) \quad (3.43)$$

$$\text{Thus, } D_{AV>} = 2(1 - p) \quad (3.44)$$

 DF for the diverse case

Consider the case for DF when $p < \frac{2}{3}$:

$$DF_{<1,2} = 0 \quad (3.45)$$

$$DF_{<1,3} = 1 - \frac{3p}{2} \quad (3.46)$$

$$DF_{<2,3} = 1 - \frac{3p}{2} \quad (3.47)$$

$$\text{Thus, } DF_{AV<} = \frac{2 - 3p}{3} \quad (3.48)$$

and similarly the case for DF when $p > \frac{2}{3}$:

$$DF_{>1,2} = DF_{>1,3} = DF_{>2,3} = DF_{<1,2} = 0 \quad (3.49)$$

$$\text{Thus, } DF_{AV>} = 0 \quad (3.50)$$

 KW for the diverse case

For KW we need to consider the total values calculated in the fourth large column in the table in figures 3.4 and 3.5 marked KW at the top. Consider the case for KW when $p < \frac{2}{3}$:

$$KW_{<} = \frac{2N}{9N} = \frac{2}{9} \quad (3.51)$$

and similarly the case for KW when $p > \frac{2}{3}$:

$$KW_{>} = \frac{6(1 - p)N}{9N} = \frac{2}{3}(1 - p) \quad (3.52)$$

κ for the diverse case

For κ we also need to consider the total values calculated in the fourth large column in the table in figures 3.4 and 3.5 marked κ at the top. For this case, the average accuracy of the three classifiers $\bar{p} = p$ since they are all considered to have the same accuracy. Consider the case for κ when $p < \frac{2}{3}$:

$$\kappa_{<} = 1 - \frac{\frac{1}{3}2N}{2Np(1-p)} = -\frac{1}{3} \left[\frac{3p^2 - 3p + 1}{p(1-p)} \right] \quad (3.53)$$

and similarly the case for κ when $p > \frac{2}{3}$:

$$\kappa_{>} = 1 - \frac{\frac{1}{3}6N(1-p)}{2Np(1-p)} = 1 - \frac{1}{p} = \frac{p-1}{p} \quad (3.54)$$

Note that these are the same as the diverse limits we established for ρ .

 Ent for the diverse case

For Ent we need to consider the total values calculated in the fifth large column in the table in figures 3.4 and 3.5 marked Ent at the top. Consider the case for Ent when $p < \frac{2}{3}$:

$$Ent_{<} = \frac{N}{N} = 1 \quad (3.55)$$

and similarly the case for Ent when $p > \frac{2}{3}$:

$$Ent_{>} = \frac{3(1-p)N}{N} = 3(1-p) \quad (3.56)$$

 θ for the diverse case

In order to calculate θ we need to consider the p.m.f. of X the proportion of classifiers which correctly classify an object \mathbf{x} . Consider the case for θ when $p < \frac{2}{3}$:

$$\theta_{<} = \frac{p - \frac{2}{9}}{1} - \left(\frac{p}{1}\right)^2 = p - \frac{2}{9} - p^2 \quad (3.57)$$

and similarly the case for θ when $p > \frac{2}{3}$:

$$\theta_{>} = \frac{\frac{1}{3}(5p-2)}{1} - \left(\frac{p}{1}\right)^2 = \frac{5p}{3} - \frac{2}{3} - p^2 \quad (3.58)$$

 GD for the diverse case

In order to calculate GD and CFD we need to consider, the proportion of classifiers which fail to correctly classify an object \mathbf{x} .

	$p < \frac{2}{3}$	$p > \frac{2}{3}$
p_0	0	$3p - 2$
p_1	$3p - 1$	$3(1 - p)$
p_2	$2 - 3p$	0
p_3	0	0

From this information, we calculate:

$$\begin{aligned} p(1)_{<} &= \frac{1}{3}p_1 + \frac{2}{3}p_2 + p_3 \\ &= \frac{1}{3}(3p - 1) + \frac{2}{3}(2 - 3p) = 1 - p \end{aligned} \quad (3.59)$$

$$p(2)_{<} = \frac{1}{3}p_2 + p_3 = \frac{1}{3}(2 - 3p) \quad (3.60)$$

$$p(1)_{>} = \frac{1}{3}(3(1 - p)) = 1 - p \quad (3.61)$$

$$p(2)_{>} = 0 \quad (3.62)$$

Consider the case for GD when $p < \frac{2}{3}$:

$$GD_{<} = 1 - \frac{p(2)}{p(1)} = 1 - \frac{\frac{1}{3}(2 - 3p)}{1 - p} = \frac{1}{3(1 - p)} \quad (3.63)$$

and similarly the case for GD when $p > \frac{2}{3}$:

$$GD_{>} = 1 - \frac{0}{1 - p} = 1 \quad (3.64)$$

CFD for the diverse case

Since $p_0 < 1$ for both cases we use the formula for CFD . Consider first the case for CFD when $p < \frac{2}{3}$:

$$CFD_{<} = \frac{1}{1 - 0}(p_1 + \frac{1}{2}p_2) = (3p - 1) + \frac{1}{2}(2 - 3p) = \frac{3p}{2} \quad (3.65)$$

and similarly the case for CFD when $p > \frac{2}{3}$:

$$CFD_{>} = \frac{1}{1 - (3p - 2)}(p_1 + \frac{1}{2}p_2) = \frac{1}{3(1 - p)}(3(1 - p) + \frac{1}{2} \times 0) = 1 \quad (3.66)$$

3.2.3 Examining the limits

The upper and lower values established for the measures of diversity are shown in Table 3.5. Recall the example for the non-pairwise diversity measures we considered previously as shown in Table 3.4. We can compare the values obtained for each of the diversity measures with what we now know about the theoretical values we could have obtained. This will give us an insight into how much real-world levels of diversity differ from the theoretical

Table 3.5: LIMITS FOR THE DIVERSITY MEASURES IN TERMS OF p

Diversity Measure	Diverse Value for $p < \frac{2}{3}$	Diverse Value for $p > \frac{2}{3}$	Identical Value $\forall p$
Q	$\frac{1-3p}{3(1-p)}$	-1	1
ρ	$-\frac{1}{3} \left[\frac{3p^2-3p+1}{p(1-p)} \right]$	$\frac{p-1}{p}$	1
DF	$\frac{2-3p}{3}$	0	$1-p$
θ	$p - \frac{2}{9} - p^2$	$\frac{5p}{3} - \frac{2}{3} - p^2$	$p - p^2$
κ	$-\frac{1}{3} \left[\frac{3p^2-3p+1}{p(1-p)} \right]$	$\frac{p-1}{p}$	1
D	$\frac{2}{3}$	$2(1-p)$	0
KW	$\frac{2}{9}$	$\frac{2}{3}(1-p)$	0
Ent	1	$3(1-p)$	0
GD	$\frac{1}{3(1-p)}$	1	0
CFD	$\frac{3p}{2}$	1	0

levels of diversity attainable. The example had classifier accuracy $\bar{p} = 0.57$ and so we need to consider the formulae for $p < \frac{2}{3}$. If we recall the average value of Q for these classifiers was 0.07 suggesting that the ensemble was close to independent. Table 3.6 shows the theoretical limits, the actual value and, if we project the distance between the upper and lower theoretical limits onto the unit scale, the distance from 1, (with identical classifiers at 0 and diverse classifiers at 1).

Table 3.6: EXAMINING THE DIVERSITY VALUES FOUND IN THE EXAMPLE

Diversity Measure	Theoretical Limits	Obtained Value	Projected Distance
Q	1 to -0.55	0.07	0.6
ρ	1 to -0.205	0.053	0.786
DF	0.43 to 0.096	0.2	0.7
θ	0.245 to 0.0023	0.09	0.698
κ	1 to -0.36	0.0498	0.698
D	0 to 0.6	0.46	0.69
KW	0 to 0.22	0.15	0.7
Ent	0 to 1	0.7	0.7
GD	0 to 0.775	0.538	0.694
CFD	0 to 0.855	0.686	0.802

This shows us that the diversity measures consider the ensemble from the example

to be diverse, since the obtained values are all much closer to the diverse, than to the identical end of the range.

Using the information we now have about the theoretical limits, we can illustrate the range of values we can obtain for the different measures of diversity. This will enable us to compare the real-world situation to the theoretical possibilities in terms of range of diversity values obtained. Figures 3.6 and 3.7 show the upper and lower limits for the ten measures of diversity for the case with $L = 3$ and $p \in [0.5, 1.0]$ [99].

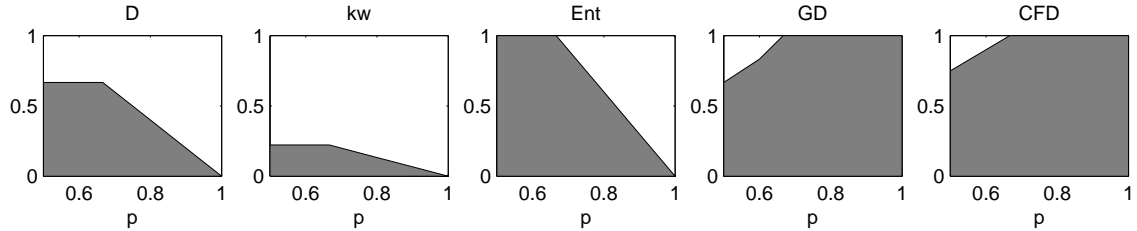


Figure 3.6: THE POSSIBLE RANGE OF VALUES (GREY AREAS) FOR THE FIVE (\uparrow) MEASURES OF DIVERSITY FOR $p \in [0.5, 1.0]$ INDIVIDUAL CLASSIFIER ACCURACY AND $L = 3$ CLASSIFIERS

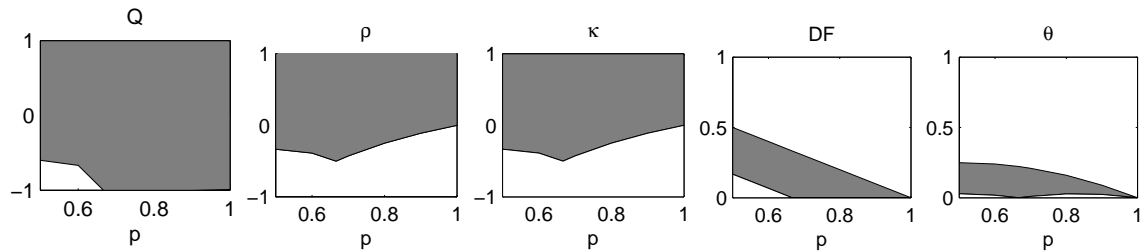


Figure 3.7: THE POSSIBLE RANGE OF VALUES (GREY AREAS) FOR THE FIVE (\downarrow) MEASURES OF DIVERSITY FOR $p \in [0.5, 1.0]$ INDIVIDUAL CLASSIFIER ACCURACY AND $L = 3$ CLASSIFIERS

3.3 Existing empirical studies of Diversity

There are several studies investigating the nature of diversity and its relationship to combination methods [106–108]. Some authors have also used a measure of classifier output correlation to enforce diversity within an ensemble during training of the classifiers [68–70, 90]. A further approach is the ‘overproduce and choose’ strategy which can be used to produce an ensemble of accurate and diverse ensembles. This approach creates an initial large set of classifiers and then uses one of several methods to choose a subset of classifiers which is both accurate and diverse [37, 38, 89].

There is a proven relationship to date due to Tumer and Ghosh [106,107] who showed that under certain assumptions the *average* combination method produces accuracy which is related to the correlation between the classifier outputs. Extending this work they also showed similar relationships for combination using *minimum*, *maximum* and *mean* [108].

Also the concept of negative correlation training of neural networks has enjoyed some interest and has been developed showing promising practical results for both regression and classification [68–70,90]. As with Tumer and Ghosh the *average* combination method is used. Ensembles can also be built using the random subspace method, which modifies the training data by sampling data features to give each classifier in the ensemble a slightly different data set to train on. Ensembles built using the random subspace method and aggregated using *majority vote* are reported to correlate well with a diversity measure based on entropy [17,18].

Masulli and Valentini found that dependence among errors was one of the main factors affecting the effectiveness of error correcting output codes [77]. They analysed the relationship between performance, design and dependence among output errors in ECOC learning machines. They compared the dependence between ECOC-monolithic made of a single multi-layer perceptron and ECOC-PND made up of a set independent and parallel dichotomizers on three data sets from UCI (glass, letter and optdigits) and one synthetic data set. Their results show that dependence among computed codeword bits is significantly smaller for ECOC-PND showing that ensembles of independent dichotomizers are better suited for implementing ECOC classification methods.

3.4 Experimental set-up to investigate the diversity measures

We again used the two databases seen in Chapter 2 from the UCI Repository of Machine Learning Database: The Wisconsin breast Cancer Database(wbc) and the Pima Indian Diabetes Database. We use the same experimental protocol as used in the previous chapter, in summary: We used the first 10 features from wbc so that we could run an exhaustive experiment with all possible partitions. The wbc data has 569 objects, 2 classes and 10 features and is trained using a hold-out (random halves) method. The Pima data has 768 objects, 2 classes and 8 features and is trained using ten-fold cross-validation.

For the wbc data we use all partitions of 10 features for 3 classifiers of the form 4,3,3 (4200 ensembles) and 4,4,2 (3150 ensembles). For each partition we designed one ensemble of three linear classifiers and one ensemble of three quadratic classifiers, resulting in 8400 4,3,3 ensembles and 6300 4,4,2 ensembles. For the Pima data we took all partitions of the form 3,3,2 using 10-fold cross-validation to obtain a total of 560 ensembles.

We then considered:

1. The range of values for the measures of diversity.
2. The correlation between each measure of diversity and all other measures of diversity.
3. The correlation between each of the methods of combination and each of the measures of diversity.

The correlation coefficient used was Pearson's Product Moment correlation coefficient.

3.5 Diversity Measures Results

3.5.1 Overall Diversities

Examining the breast cancer data results, we found that the minimum observed value of the individual classifier accuracy, p , was 0.6258, the maximum was 0.9579 and the overall mean was 0.8927. For the Pima data, the minimum was 0.6328, the maximum was 0.7734 and the overall mean was 0.6962. Tables 3.7 and 3.8 show the observed range of values for the ten measures of diversity compared with their theoretical limits for wbc and Pima respectively. Based on the observed minimum and maximum accuracies mentioned previously, we have taken $p \in [0.6, 0.95]$ for wbc and $p \in [0.6, 0.8]$ for Pima. The theoretical limits were deduced from the graphs shown previously in Figures 3.6 and 3.7. The graphical representation is used to illustrate how much the observed and theoretical ranges differ from each other. The larger rectangle represents the length of the theoretical range of diversities possible. The black section of the rectangle corresponds to the observed range of values we found in our experiments. This idea corresponds to the unit length we discussed earlier in reference to Table 3.6. Here however we have scaled it so that we can compare not only the theoretical and observed values for each diversity measure, but also compare the ranges of the different diversity measures to each other. For example, kw has a much smaller range of theoretical (and observed) values than does Q .

Considering the breast cancer data shown in Table 3.7, Q , ρ and κ can all take negative values when classifiers are negatively correlated. Given that none of these measures has any observed negative values, we can conclude that the classifiers are not very diverse for the breast cancer data. The measures where low values indicate high diversity, (\downarrow), except DF and θ , have high values, toward the right end of the range, as shown in Table 3.7. The measures where high values indicate high diversity, (\uparrow), except for GD and CFD , have low values. This suggests, that the classifiers are less diverse than they could theoretically be. For the Pima data we see that the range of diversity for each measure is considerably less than for the breast cancer data, illustrated by the narrower black bands in Table 3.8. The observed values do show that the classifiers are even less diverse than for the breast cancer data, with not even DF and θ indicating particularly diverse classifiers.

Table 3.7: THE OBSERVED RANGE OF VALUES FOR THE DIVERSITY MEASURES COMPARED WITH THE THEORETICAL LIMITS POSSIBLE FOR THE OBSERVED VALUES OF p FOR THE BREAST CANCER DATA

Diversity Measure	Theoretical Limits for $\bar{p} \in [0.6, 0.95]$	Observed Range	Graphical representation
Q (\downarrow)	$[-1.00, 1.00]$	$[0.35, 0.97]$	
ρ (\downarrow)	$[-0.5, 1.00]$	$[0.19, 0.70]$	
DF (\downarrow)	$[0.00, 0.40]$	$[0.05, 0.10]$	
κ (\downarrow)	$[-0.50, 1.00]$	$[0.10, 0.70]$	
θ (\downarrow)	$[0.00, 0.24]$	$[0.05, 0.09]$	
D (\uparrow)	$[0.00, 0.66]$	$[0.05, 0.28]$	
kw (\uparrow)	$[0.00, 0.22]$	$[0.02, 0.09]$	
Ent (\uparrow)	$[0.00, 1.00]$	$[0.07, 0.42]$	
GD (\uparrow)	$[0.00, 1.00]$	$[0.27, 0.74]$	
CFD (\uparrow)	$[0.00, 1.00]$	$[0.43, 0.86]$	

() theoretical; () observed range of values.

It is interesting to note that even though the measures do not indicate that all the classifiers are identical or close to identical, the average accuracy of the team was quite similar to the average best individual accuracy, especially for the wbc data. Thus a range of values of diversity did not span a similar range of improvement/degradation of team accuracy. This is an early indication of the lack of any strong, exploitable relationship between diversity measures and team accuracy in real-life classification problems.

3.5.2 Relationships among the diversity measures

For the case of correct/incorrect (1/0) classifier outputs (oracle-type outputs), kw differs from the averaged disagreement measure D by a coefficient [62]. Also for the case with $L = 3$ classifiers, kw and Ent differ by a coefficient (Appendix A.1, proposition 3). This in turn means that the disagreement measure and Entropy differ by a coefficient for the three classifier case, with correct/incorrect, outputs, i.e.,

$$kw = \frac{L-1}{2L} D \quad (1/0 \text{ outputs})$$

Table 3.8: THE OBSERVED RANGE OF VALUES FOR THE DIVERSITY MEASURES COMPARED WITH THE THEORETICAL LIMITS POSSIBLE FOR THE OBSERVED VALUES OF p FOR THE PIMA DIABETES DATA

Measure	Theoretical Limits for $\bar{p} \in [0.6, 0.8]$	Observed Span	Graphical representation
Q (\downarrow)	[-1.00,1.00]	[0.57,0.78]	
ρ (\downarrow)	[-0.5,1.00]	[0.30,0.46]	
DF (\downarrow)	[0.00,0.40]	[0.16,0.19]	
κ (\downarrow)	[-0.50,1.00]	[0.29,0.46]	
θ (\downarrow)	[0.00,0.22]	[0.11,0.13]	
D (\uparrow)	[0.00,0.62]	[0.23,0.30]	
kw (\uparrow)	[0.00,0.22]	[0.08,0.10]	
Ent (\uparrow)	[0.00,1.00]	[0.34,0.45]	
GD (\uparrow)	[0.00,1.00]	[0.38,0.49]	
CFD (\uparrow)	[0.00,1.00]	[0.54,0.65]	

() theoretical;() observed range of values.

$$\begin{aligned}
 \Rightarrow kw &= \frac{1}{3}D & (1/0 \text{ and } L = 3) \\
 kw &= \frac{2}{9}Ent & (1/0 \text{ and } L = 3) \\
 \Rightarrow D &= \frac{2}{3}Ent & (1/0 \text{ and } L = 3)
 \end{aligned}$$

The following results are an expansion of our study into combination methods and diversity measures [99]. We were interested in how the different diversity measures might be related to each other. In order to investigate these relationships we calculated the correlation between their values and illustrated them as shown in Figure 3.8. Since we have two types of diversity measure, those measuring similarity (\downarrow) and those measuring diversity (\uparrow) we would naturally expect some of the measures to be highly negatively correlated. Since we are interested in *any* relationship and not whether it is positive or negative, we have taken the absolute values of the correlation coefficient. The intensity of the colour in Figure 3.8 is determined by this correlation. The stronger the correlation the lighter the colour.

To further aid analysis of these relationships we used a clustering program to illustrate

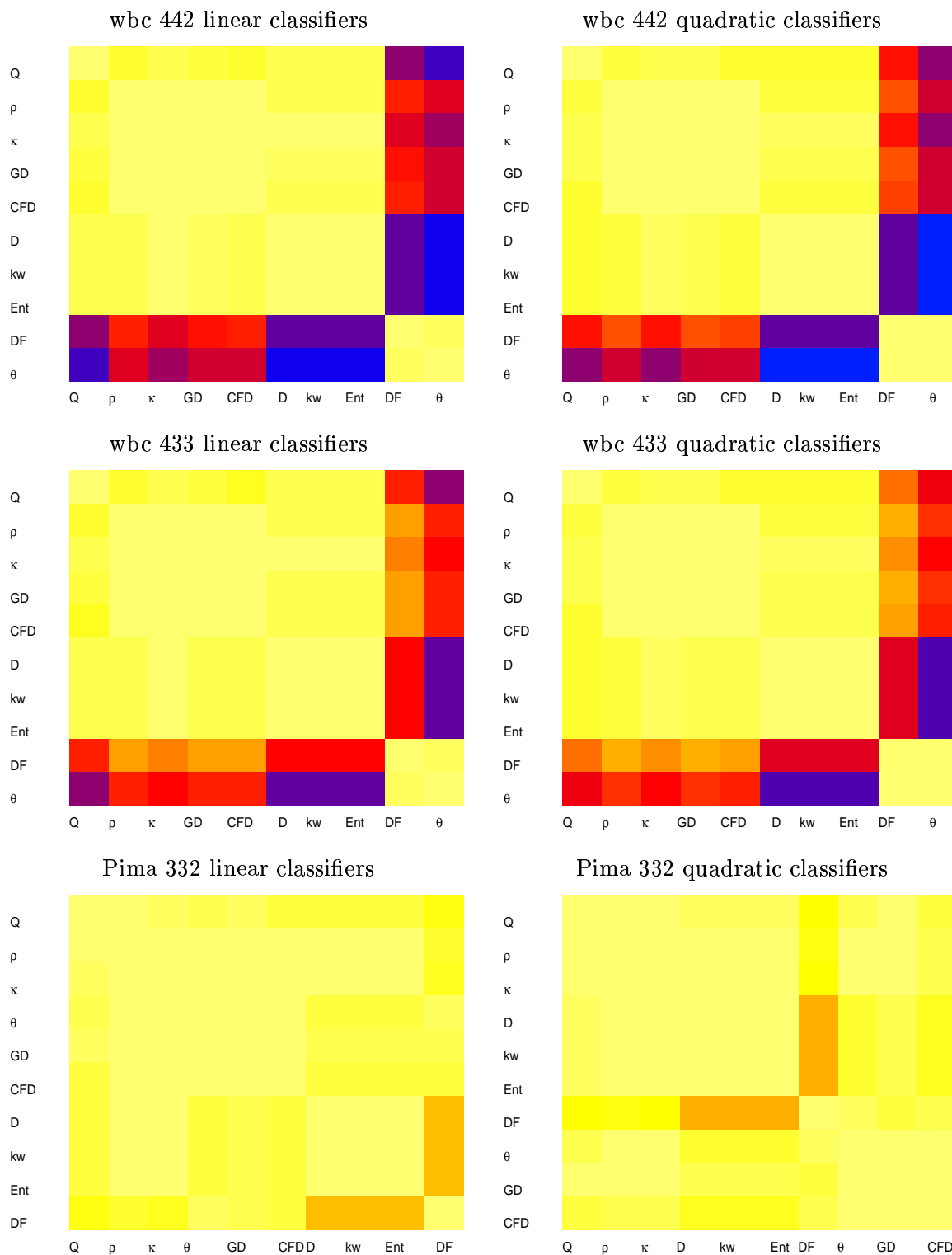


Figure 3.8: THE OVERALL CORRELATION BETWEEN THE DIVERSITY MEASURES. THE STRONGER THE CORRELATION THE LIGHTER THE COLOUR.

the strength of the relationships between the different diversity measures. Figure 3.9 shows the dendrograms formed when we cluster the diversity measures using average-linkage relational clustering¹. The lower the branches joining the different diversity measures the stronger the relationship between them.

Looking at the colours in Figure 3.8 and the height of the branches in Figure 3.9 and comparing them with the corresponding figures for the combination methods (2.5 and 2.6) we see that the colours are much lighter and the branches are much lower for the diversity measures showing that the diversity measures are more correlated with each other than the combination methods are. The colours are also lighter for the diversity measures with the Pima data set than with the breast cancer data indicating that the diversity measures were more correlated on this data set.

Looking at both the shade diagrams and the dendrograms we can see that the relationships are more complicated than for the combination methods with quite different clusters for the two data sets. We can see for both data sets that D , kw and Ent are identical on the basis of the correlation coefficients. Looking at figures for the breast cancer data it is easier to identify clusters but for the Pima data the diversity measures are so closely correlated that it is very difficult to split them into consistent clusters. It appears that ρ , κ , GD and CFD are fairly closely correlated to each other, but it is interesting to note that GD is more correlated to κ (for wbc) and to θ (for Pima) than to CFD which is actually a more similar measure from a derivation point of view. DF only has correlation with θ and even that is not very strong for the Pima data. DF measures the proportion of examples which a pair of classifiers both misclassify. Based on the correlation colours and the cluster dendrograms it seems that this is different to the other diversity measures.

3.5.3 Relationship with accuracy

One of the key questions when we consider diversity is ‘how can we make the best use of diversity?’ One approach is to try to force an ensemble to be diverse. Negative correlation can be enforced into an ensemble to increase the accuracy of a multiple classifier system. Liu and Yao propose a negative-correlation training method for ensembles of Neural Networks with considerable success [68, 69, 71, 117].

Liu and Yao have carried out a series of studies into using negatively correlated neural networks on two data sets from the UCI machine learning repository [68–70, 117]. Using Australian credit card assessment data and the Wisconsin breast cancer data (which we have also used) they have shown that an ensemble of negatively correlated neural networks can perform significantly better than other systems for regression tasks without any noise. Their algorithm enforces negative correlation within the ensemble by encouraging different

¹The clustering routine and the dendrogram drawing routine are from the package PRTOOLS for Matlab [26]

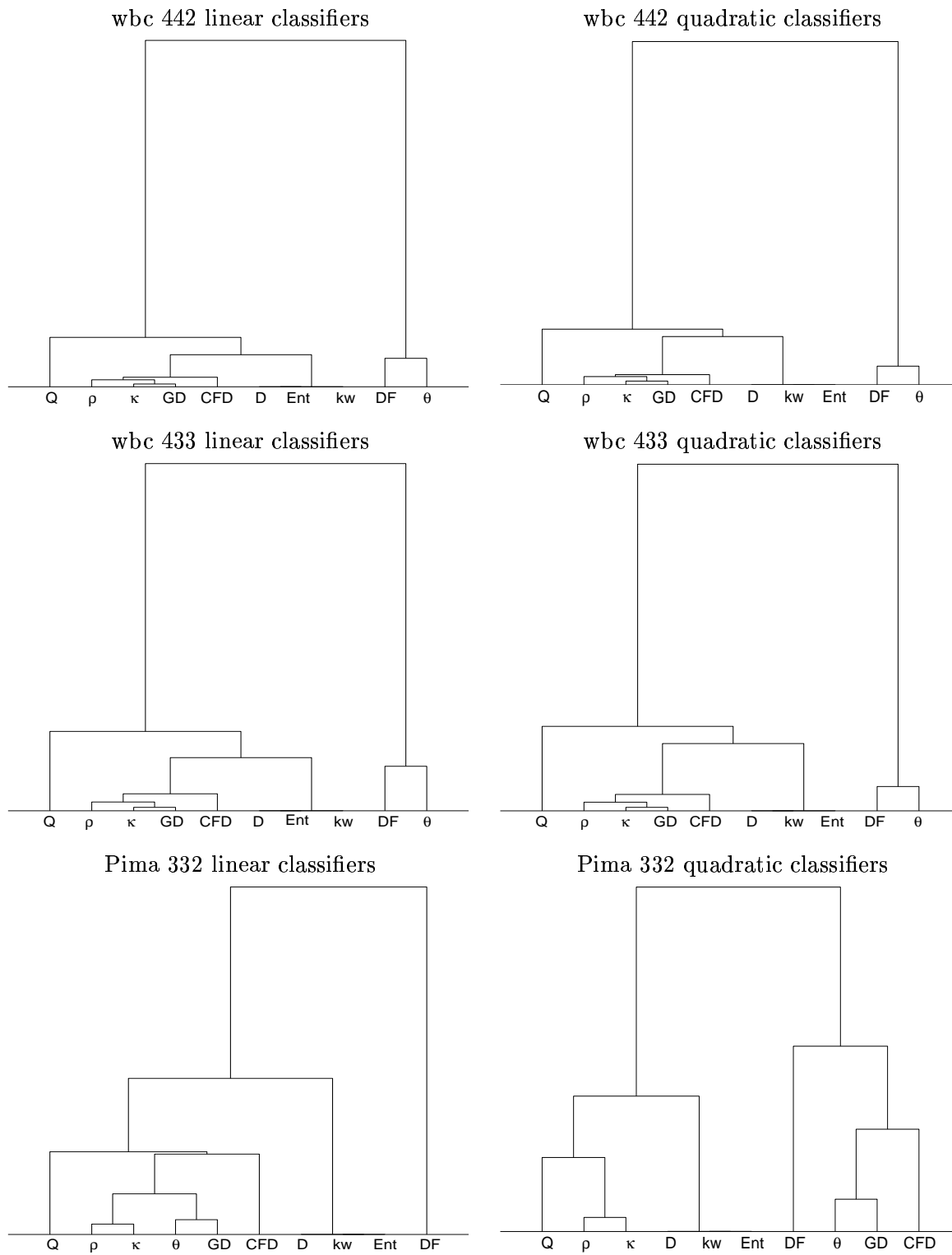


Figure 3.9: THE CLUSTER DENDROGRAMS FOR THE DIVERSITY MEASURES

individual networks to learn different parts or aspects of the training data, and then by using a novel correlation penalty term in the error function to encourage this specialisation.

The networks are trained simultaneously to allow interaction between different networks. They compare their algorithm with thirteen other algorithms and show that theirs is better than nearly all of them including CART, logistic discriminant, radial basis function and Naive Bayes [68, 69]. In further work they consider two approaches to breast Cancer diagnosis, a monolithic, feed-forward, evolutionary artificial neural network, and a set of several simultaneously trained, feed-forward networks to create an ensemble. They found on the highly non-linear, Chlorophyll-a prediction problem that negative correlation learning is statistically significantly better than independent training [117]. In the same study, using the breast cancer data again, they initially found that combining using simple average *reduced* the benefits of training with negative correlation. On analysing the causes they found that this was due to the fact that some of the constituent networks are more important than others as they have all learned different parts of the data. They found that by changing their combination method to combining by applying a ‘winner-takes-all’ approach and for each area only using the network with the highest activation dramatically improved the testing error.

Another way of using diversity is in the ‘overproduce and choose’ strategy where a large set of classifiers is created and then the most accurate and diverse subset is chosen from them. Giacinto and Roli have produced several studies following this strategy [37, 38]. They choose to use this overproduce and choose approach because direct generation of accurate and diverse ensembles is too difficult for the current state of the classification field of research. It also allows the exploitation of all ways of creating diverse candidates for the ensemble by varying net type, initial random weights, network architecture and training data. Their data consists of multi-sensor remote sensing images relating to an agricultural area near the village of Feltwell (UK) with the task of assigning one of five agricultural classes to each of 5238 testing pixels given 5124 training pixels and 582 validation pixels. They used several different overproduction phases to get different initial large sets of different types of classifiers. Their approach to choosing the subset of classifiers involves examining the number of coincident errors made by the classifiers. They cluster the classifiers into subsets by putting those with highly correlated coincident errors in the same cluster so that those in different clusters are error-diverse. At each stage of clustering a candidate ensemble is created by taking one classifier from each cluster and combining by majority voting using the validation set. Once the clustering is completed all of the candidate ensembles are compared and the one with the highest performance is chosen as the final ensemble. They proved that their approach is optimal and compared it with another overproduce and choose strategy and found it to be preferable. In a later study with Vernazza, they compared their approach to a further five strategies on the Feltwell data again, using three different diversity measures, Generalised Diversity, the Q -statistic and their own Compound Diversity which is in fact 1–Double Fault [89]. Using several

initial large sets from the overproduce stage, they found that it was not possible to identify the best ‘choose’ strategy since it varied depending on the task at hand.

In order to investigate how we can make use of diversity we look at the inter-relationships between the combination methods from the previous chapter and the diversity measures introduced in this chapter.

3.5.4 Relationships between the combination methods and the diversity measures

For each combination method we have a column of values, the accuracies (in the range $[0 - 1]$). There is one value for each of the different possible ensembles and there is one column of accuracies for each of the possible partition types (4,3,3 etc.). For each diversity measure we also have a column of values, the diversities (range depending on the particular diversity measure, shown in Figures 3.6 and 3.7). Again there is one value for each of the different possible ensembles and a different column for each partition type. Thus each particular row for the combination methods and diversity measures corresponds to the same particular set of three classifiers. Therefore, we can compare the combination methods and diversity measures by comparing these values. We do this by calculating the Pearson’s product moment correlation between these corresponding values. Figure 3.10 illustrates these correlations for both the breast cancer and Pima data.

As before the intensity of the colour in the Figure (3.10) is determined by the correlation. The stronger the correlation the lighter the colour. The darker colours in these shade graphs show that there is less correlation between the combination methods and the diversity measures than either the combination methods or the diversity measures show amongst themselves. In fact there is very little consistent correlation between the two. $D/kw/Ent$ showed a slight correlation with MAX/MIN , θ some correlation to MAJ and AVR , and DT and DF some correlation to MAJ .

Table 3.9 shows the correlation coefficients obtained when combining all of the data for both wbc and Pima and comparing the diversity measure values and the combination method accuracies. From Table 3.9, DF and θ are the only measures which show correlation of size 0.5 or more with most of the combination methods. The correlation with BKS and WER is just under 0.5 for DF and is just under 0.5 with WER for θ . Overall, θ shows stronger correlation with the combination methods than DF does. They show negative correlation values, which means that if the combination method has high accuracy then DF and θ have low value, which reflects high diversity since they are of the (\downarrow) form.

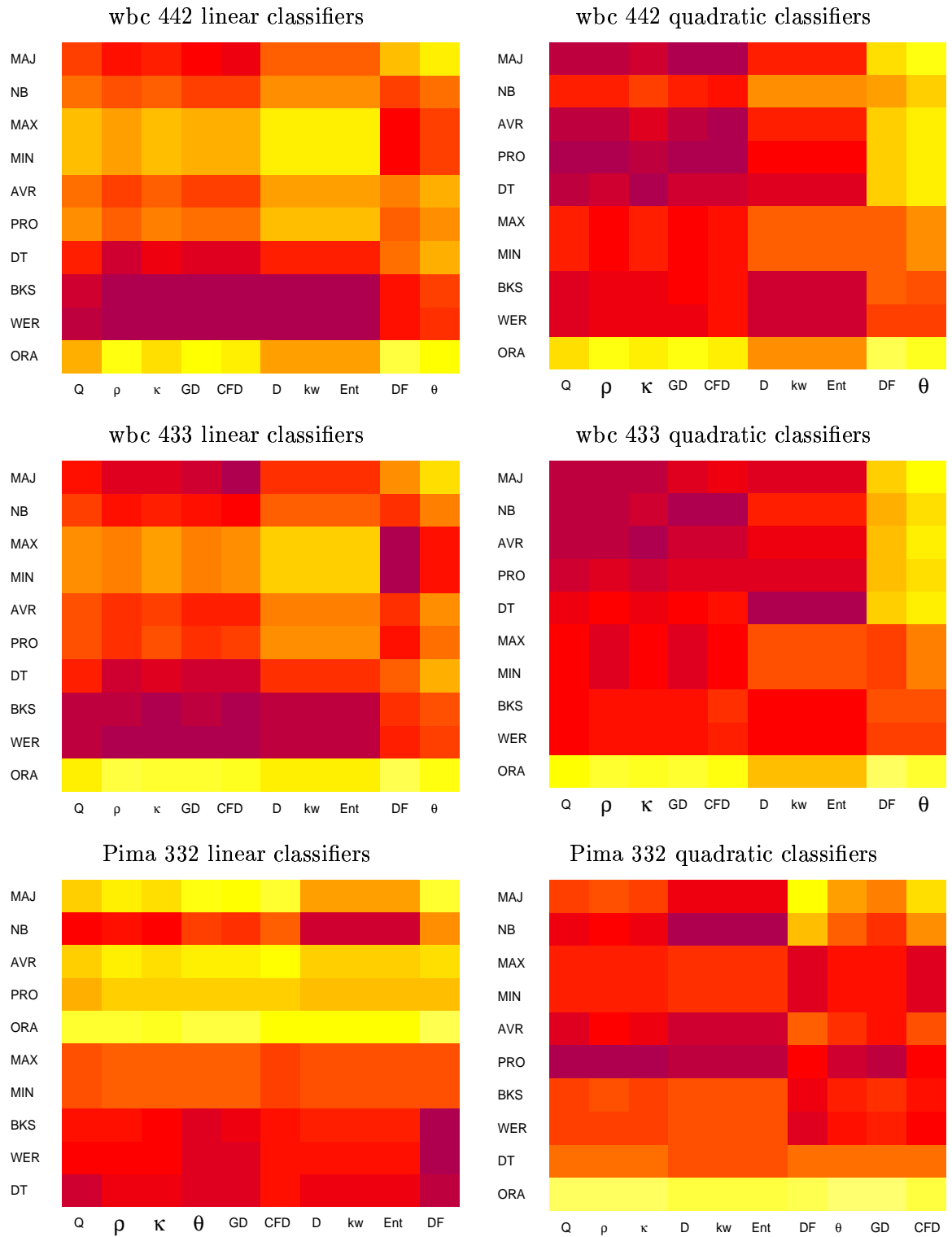


Figure 3.10: THE CORRELATION BETWEEN THE COMBINATION METHODS AND THE DIVERSITY MEASURES. THE STRONGER THE CORRELATION THE LIGHTER THE COLOUR.

Table 3.9: THE CORRELATION COEFFICIENTS FOR THE RELATIONSHIPS BETWEEN THE COMBINATION METHODS AND THE DIVERSITY MEASURES. BOLD NUMBERS ARE THOSE WITH ABSOLUTE VALUE OF 0.5 OR GREATER.

	Q	ρ	$D/kw/Ent$	DF	κ	θ	GD	CFD
<i>MAJ</i>	-0.0009	-0.1725	-0.0582	-0.8984	-0.1442	-0.9199	0.2139	0.3248
<i>NB</i>	0.1461	0.0047	-0.2524	-0.7239	0.0430	-0.7660	0.0323	0.1216
<i>BKS</i>	-0.0902	-0.2060	0.0966	-0.4965	-0.2020	-0.5108	0.2343	0.2776
<i>WER</i>	-0.0868	-0.1977	0.1014	-0.4633	-0.1955	-0.4747	0.2242	0.2661
<i>MAX/MIN</i>	0.1725	0.0017	-0.2430	-0.7690	0.0506	-0.7994	0.0200	0.0811
<i>AVR</i>	0.0286	-0.1418	-0.0865	-0.8854	-0.1086	-0.9071	0.1768	0.2682
<i>PRO</i>	0.0346	-0.1388	-0.1050	-0.8692	-0.0972	-0.8905	0.1680	0.2428
<i>DT</i>	-0.0315	-0.2244	-0.0055	-0.8819	-0.1888	-0.9047	0.2539	0.3384
<i>ORA</i>	-0.4198	-0.6278	0.3822	-0.9646	-0.5817	-0.9384	0.6403	0.6886

3.6 Diversity Measures Conclusions

In this chapter we introduced several diversity measures and studied the relationships between them and to the combination methods introduced in Chapter 2. We took a breast cancer data-set of 10 feature values for 569 patients and using all partitions of the form (4,4,2) and (4,3,3) for two types of classifier (linear and quadratic), conducted a set of four enumerative experiments. We also took a diabetes data set of 8 feature values for 768 patients and conducted a set of ten-fold cross-validation experiments using all possible partitions of the form (3,3,2).

We then considered the ranges of the diversity measures for the classifiers produced compared with the theoretical range and their implications for the accuracy of the ensemble. In the previous chapter we studied the accuracy of ensembles formed from these classifiers and found that they did not improve considerably over the average single best individual classifier. This is probably due to the fact that the classifiers produced were not particular diverse.

Next we studied the correlation amongst the diversity measures. We found that for (0/1) classifier outputs with an ensemble of three classifiers D , kw and Ent are identical up to a coefficient. We also found that ρ , κ , GD and CFD are fairly, consistently correlated whilst DF was not strongly correlated with any of the other measures.

We then proceeded to look at the relationship between the diversity measures and the combination methods introduced in Chapter 2. We found that there was very little

consistent correlation between the two. On combining all data, both wbc and Pima with all partition types, we found that DF and θ had the strongest relationship with the combination methods.

It is the fact that there is no simple, clear relationship between diversity measures and combination methods which makes the explicit use of diversity in multiple classifier systems such a thorny subject. In order to use diversity we would like to see a consistent, positive relationship showing that highly diverse ensembles have high accuracy on combination. Unfortunately we have not found this.

Directly calculating the accuracy for the chosen combination methods currently makes more sense than calculating diversity and trying to predict the accuracy, with the measures currently at our disposal. This is true even if the measure of diversity is easier to calculate than some combination methods, the ambiguous relationship between diversity and accuracy discourages optimising the diversity. It may be possible to create a hybrid measure which in some way combines accuracy and diversity into a single measure to be able to use diversity. For now, it is better to use diversity as mentioned in section 3.5.3 by trying to enforce diversity in the ensemble or using diversity to select an ensemble when following an ‘overproduce and choose strategy’. We shall consider this latter option in Chapter 5.

Chapter 4

Perturb and Combine Ensemble Construction Methods

So far we have considered the various ways of combining ensembles of given classifiers. We have also investigated how the diversity of an ensemble of classifiers can be measured and how this may be related to the ensemble accuracy. Now we will consider ways in which we can try to actively alter the diversity of the constituent classifiers in an ensemble. Recall figure 2.2, which showed what we can change in a multiple classifier system. Part *D* showed that we can alter the classifiers in an ensemble by modifying the training data on which they are built. This chapter deals with trying to create an ensemble of diverse classifiers by manipulating the training data in one of a various number of ways. These include Bagging, Boosting and Arcing algorithms [9, 20, 22]. These are usually referred to as Perturb and Combine methods because we first perturb the data to get different training sets and then we combine the classifiers built on them [8]. We hope that by doing this we can improve the ensemble performance (accuracy on combination) through producing more diverse classifiers.

4.1 Bias and Variance

There are many different definitions of bias and variance with regards to classifier combination problems [24, 50, 95]. Freund and Schapire [32] found that it is not always necessarily easy or useful to use the bias-variance decomposition with boosting algorithms in classification. Here we consider Breiman's definition [9]. Consider taking a large number of replicated training sets of size N from the same underlying distribution, and consider the average performance of the algorithms on them. Breiman shows that the average error can be decomposed into a noise term, a bias term and a variance term:

$$\overline{PE}(f) = \sigma^2 + \beta^2(f) + V(f) \tag{4.1}$$

where f is the classifier produced by the algorithm, $\overline{PE}(f)$ is the average prediction error, σ^2 is the noise term, $\beta^2(f)$ is the bias term and $V(f)$ is the variance term.

Neural nets and decision trees tend to have low bias, but their problem is a high variance, they are unstable since they are sensitive to small changes in the learning set, or in the construction. For these methods the problem is how to reduce the variance. There are several methods that have been devised to address this problem.

4.2 Bagging

Breiman devised a method of manipulating the data to try to reduce the variance for unstable classifiers which he called bagging [7].

Suppose we have a training set \mathbf{T} and we could obtain a large number of independent samples of this training set $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \dots$ each with the same number of objects, N , and coming from the same underlying distribution. If we take the i^{th} sample we can construct a predictor function to act as our classifier $D_{\mathbf{T}_i}(\mathbf{x})$. Recalling the bias-variance decomposition of the error we can average all of the predictors to define a new predictor:

$$\overline{D}(\mathbf{x}) = \text{avg}_i D_{\mathbf{T}_i}(\mathbf{x}) \quad (4.2)$$

According to Breiman, this function has the same bias as $D_{\mathbf{T}}(\mathbf{x})$ but zero variance, resulting in a lower prediction error [9].

Unfortunately, we cannot usually obtain the large number of independent copies \mathbf{T}_i we require to take advantage of this approach. In order to overcome this problem we use *Bootstrap Approximation*.

In Bootstrap sampling we sample from Z with replacement, for bagging we assume that all objects in Z have an equal likelihood of being chosen. Typically the size of the bootstrap sample is taken to be the same as that of Z , i.e., N . This can be carried out an unlimited number of times to obtain as many training sets as we wish. Each replicate contains on average 63.2% of the original training set [21]. Each of the bootstrap samples is then used to construct a different classifier which can be combined to give an ensemble. Bagging is useful as it is very easy to implement and construction of each bootstrap sample and classifier can be run in parallel thus computer run time is kept to a minimum. It is also resistant to errors due to noisy data [20]. However, it is not always as accurate as some other methods [5, 20, 30].

Skurichina and Duin investigated how the random subspace method and bagging compared when datasets have redundant features [101]. They considered one artificial, and five real two-class datasets from the UCI machine learning repository which they adapted to have high redundancy in the feature space. They found that the performance of bagging was unaffected by the redundancy in the dataset but rather on the data-dimensionality

and its relation to the training sample size. In highly redundant feature spaces where many of the features are noise, bagging was superior to the random subspace method but in datasets where the discrimination information is spread over all the features, the random subspace method was superior.

Breiman has looked into new variants of bagging aimed at improving the performance by reducing both bias and variance in regression problems [12]. This is achieved by initially performing bagging and then altering the output values based on the outcomes from bagging. Next bagging is carried out on these altered output values. This process is then repeated until their stopping rule is satisfied.

4.3 Arcing and Boosting

Building a highly accurate classifier can be a difficult problem, however finding moderately accurate classifiers is a comparatively easy task. Boosting is based on the observation that finding many rough classifiers can be a lot easier than finding a single, highly accurate classifier. Boosting refers to a general and provably effective method of producing an accurate classification rule by combining moderately inaccurate classifiers [92].

To understand the idea of boosting we need to introduce the concept of *weak* and *strong* learners. As defined by Breiman in [8], a *weaklearner* is a computationally efficient algorithm which is only slightly better than random guessing for *any* distribution. While a *stronglearner* is accurate over the whole input space, not just the training set and so has low test error. Breiman then defines a boosting algorithm as being any algorithm which can take a *weaklearner* and boost it into being a *stronglearner*.

Boosting was developed in answer to the question ‘Does the existence of a computationally efficient *weaklearner* imply the existence of an efficient *stronglearner* that can generate arbitrarily accurate hypothesis?’ [8]. Schapire answered this question in the affirmative by developing the first provable polynomial-time boosting algorithm [93]. Later Freund described a simpler and more efficient boosting algorithm which he called *boost-by-majority* [29]. This had certain practical drawbacks, in that it requires prior knowledge of the amount they believe the classification algorithm to be better than random guessing. Together, Freund and Schapire then went on to develop the AdaBoost algorithm. Whilst not quite as efficient as *boost-by-majority*, AdaBoost is more practical since it is adaptive in nature and does not require the prior knowledge referred to above [31].

Breiman determines that AdaBoost and Freund and Schapire’s other boosting algorithms do not in fact strictly fulfil the boosting assumptions [8]. He says that to satisfy the strict definition of a *weaklearner* the classes must not have any overlaps. He believes that these boosting assumptions are restrictive, since in virtually all real data situations there is some overlap between classes and no *weaklearners* exist. He prefers the term

‘arcing’ which he believes is less restrictive. Arcing stands for **A**daptive **R**esampling and **C**ombining algorithms. It refers to algorithms which

1. assign weights to the training data
2. build classifiers on the training data taking the weights into account,
3. increase the weight for those data that were misclassified by the previous classifier,
4. combine the ensemble of classifiers built to produce an overall classification rule.

Freund and Schapire acknowledge that under the strictly defined rules of boosting there may be cases when there are no weak learners and that their boosting algorithms would indeed be called ‘arcing’ algorithms according to Breiman’s terminology [32]. The term boosting is commonly used to encompass these ideas, and in particular in association with the now renowned AdaBoost algorithm. To prevent confusion we will use the term Boosting in its broadest sense throughout the rest of this work.

4.4 Which method to use?

According to Breiman [8] the main contribution of both bagging and boosting on the reduction of the error is through reduction of the variance and in this, boosting is usually the better method. However, Freund et. al.’s experiments [32, 95] showed that boosting can decrease both variance and bias in some cases but can also increase the variance whilst reducing the bias sufficiently to still ensure that the final error is reduced.

We are particularly interested in the AdaBoost boosting algorithm [31] which has had considerable success with artificial and real-world data problems [5, 8, 30, 85]. As well as being an off-the-shelf algorithm which only involves entering the data and hitting the start button, AdaBoost can usually take a good but not exceptional classifier such as CART and turn it into a procedure with performance close to the lowest achievable test set error rates. In fact, Breiman (at the NIPS workshop 1996 as quoted in [34]) referred to AdaBoost in combination with decision trees as the “best off the shelf classifier in the world”. We will describe this algorithm in more detail below. It has been found that in circumstances without noise, boosting is clearly superior to bagging and that AdaBoost produces much more diverse ensembles than either bagging or randomising [20]. It is this diversity which can lead to greater improvement in accuracy for AdaBoost [103]. However bagging outperforms boosting when substantial classification noise is introduced. Unfortunately, it has also been observed that boosting can be paralysed [113], i.e., no further improvement is achieved when adding new classifiers to the team.

For the remainder of this thesis we will be working predominantly with the boosting algorithm, called AdaBoost.

4.5 AdaBoost

AdaBoost was originally designed to rapidly drive the *training* error to zero. The fact that it is also extremely good at reducing the testing error is a fortunate by-product [9]. The name AdaBoost comes from the phrase **Ad**aptive **B**oosting.

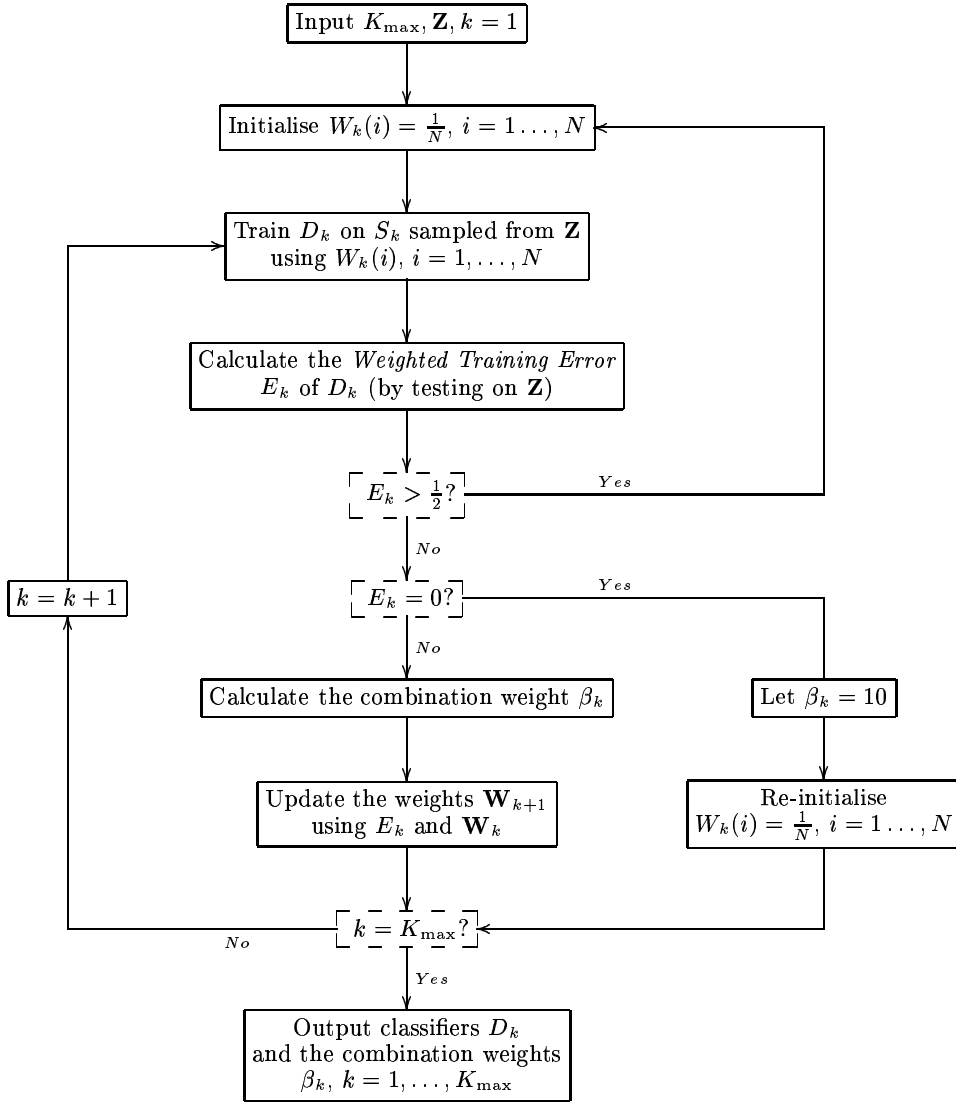
AdaBoost constructs classifiers by modifying the training set based on the previous classifier's performance. It does this by getting the new classifier to put more emphasis on those objects which the previous classifier found difficult to classify accurately. This is achieved by maintaining a distribution of weights over the training set, which can be modified as required on each iteration. Thus if the current classifier finds a certain object difficult to classify then that object will have greater weight for the next iteration. Conversely, if the current classifier finds a certain object easy to classify then that object will have less weight in the next iteration.

4.5.1 The AdaBoost algorithm

Some implementations of AdaBoost use a resampling method [9, 22–24] and others use reweighting [5]. There is some disagreement about which to use and some studies have therefore compared both [9, 34]. These implementations differ according to whether you resample from the original training set or attach weights to each data point and re-use the whole training set to build the next classifier. The choice of implementation does not affect AdaBoost too much although boosting with reweighting is a more direct implementation of the theory [5]. Research by Breiman suggests that there is very little difference in the results obtained using the two methods [9]. We have used a resampling method because this allows for any type of basic classifiers to be used. The reweighting method requires some modification of the underlying classifiers to allow them to accommodate the weights as input and so cannot be used with any type of classifier like the resampling method.

For the resampling implementation, each weight determines the probability of its associated object being selected for the training set for an individual component classifier [24]. Initially all weights are set equal. On each round if a training object is not accurately classified then its chances of being selected again for a subsequent training set are increased by increasing the value of its associated weight [94]. In this way the next classifier is forced to concentrate on the more difficult examples in the training set. In Figure 4.1 we show the basic algorithm for AdaBoost using the resampling implementation.

We require as input, the training set Z and K_{\max} the maximum number of iterations we wish the algorithm to carry out. We initialise the weights associated with each element in the training set $W_k(i)$ $i = 1, \dots, N$ to be equal (i.e. $\frac{1}{N}$). At each iteration k , a bootstrap sample S_k is taken from the training set with replacement using the weights to determine each object's chance of being selected. On the first iteration, this process is identical to



Key:

k the current iteration; K_{\max} the number of iterations; \mathbf{Z} the training set;
 S_k the k^{th} training set; D_k the k^{th} classifier trained; E_k the k^{th} training error;
 β_k the k^{th} combination weight; $W_k(i)$ the k^{th} weight for object i ;
 $\mathbf{W}_k = \{W_k(1), \dots, W_k(N)\}$ the k^{th} set of weights used.

Figure 4.1: THE ADABOOST ALGORITHM: THE RESAMPLING IMPLEMENTATION

the bootstrap sample taken in bagging. As in bagging the size of the sample set is usually taken to be N , the same as the original training set Z . A classifier D_k is trained on this training set S_k and is then tested on the original training set Z to obtain the weighted training error, E_k :

$$E_k = \sum_{i=1}^N d(i)W_k(i) \quad (4.3)$$

$$\text{where } d(i) = \begin{cases} 1 & \text{if } \mathbf{Z}_i \text{ is misclassified by } D_k \\ 0 & \text{if correctly classified} \end{cases}$$

If E_k is greater than $\frac{1}{2}$ then classifier D_k is considered to be too inaccurate (worse than random guessing for two class case). If $E_k = 0$ we also encounter a problem since we need to use it as the denominator in the equation for the combining weights, β_k , at the following step. Earlier implementations of AdaBoost [30] stopped when $E_k > \frac{1}{2}$ (error greater than 50%) was reached and did not specify what to do with $E_k = 0$. Subsequently Breiman [8,9] suggested re-starting with re-initialised weights after either $E_k = 0$ or $E_k > \frac{1}{2}$ and ignoring that iteration. We decided that it was not necessary to ignore the iteration when $E_k = 0$ (100% accurate individual classifier on the training set) since this gave us a good classifier to add to our ensemble. Instead we decided that giving a suitably large value to β and re-initialising the weights was a better approach. Thus, when $E_k > \frac{1}{2}$ the weights $W_k(i)$ are reinitialised to $1/N$ and we ignore that iteration. When $E_k = 0$ we include that classifier in our collection, assign a value of 10 to β ¹ and re-initialise the weights.

When we obtain a classifier D_k with error $0 < E_k < \frac{1}{2}$ we calculate the combination weight β_k as follows:

$$\beta_k = \frac{1 - E_k}{E_k}. \quad (4.4)$$

Next we update the weights for each object $W_{k+1}(i)$, using the current weights W_k and error E_k as

$$W_{k+1}(i) = \frac{W_k(i)\beta_k^{d(i)}}{\sum_{j=1}^N W_k(j)\beta_k^{d(j)}}, \quad (4.5)$$

If we have not reached the required number of classifiers we repeat the process until we have obtained K_{\max} classifiers. Once we have obtained the required number we output the classifiers and the combination weights D_k, β_k $k = 1, \dots, K_{\max}$.

The final decision for classification of a new object, \mathbf{x} , is made by weighted voting between the K_{\max} classifiers. First, all classifiers label \mathbf{x} and then for all D_k that gave label ω_t , we calculate the support for that class by

$$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x})=\omega_t} \ln(\beta_k). \quad (4.6)$$

The class with the maximal support is chosen for \mathbf{x} .

4.5.2 Optimality of the combiner for AdaBoost

The final support for each class is given by equation 4.6. If we re-call that $\beta_k = \frac{1-E_k}{E_k}$ then 4.6 can also be written as:

$$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x})=\omega_t} \ln\left(\frac{1-E_k}{E_k}\right). \quad (4.7)$$

¹10 was chosen after examining the sizes of β for various errors, E_k

The following theorem shows the rationale for this combiner, showing why it could be used as the support for each class, ω_t . It is described by Freund and Schapire in [31].

Theorem 1 *Given an ensemble of classifiers $\mathcal{D} = \{D_1, \dots, D_L\}$ with corresponding errors $\{E_1, \dots, E_L\}$.*

Let $D_i(\mathbf{x}) = s_i$ be the output of classifier D_i for object \mathbf{x} where $s_i \in \Omega$.

Denote $\mathbf{s} = [s_1, \dots, s_L]$ and assume conditionally independent outputs, i.e.,

$$P(\mathbf{s}|\omega_i) = \prod_{j=1}^L P(s_j|\omega_i) \quad (4.8)$$

Then a set of Bayes-optimal discriminant functions are given by

$$g'_i(\mathbf{x}) = \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} \log \frac{1 - E_j}{E_j}, \quad (4.9)$$

where $I(\omega_i)$ is the index set of all classifiers that assigned object \mathbf{x} to class ω_i , i.e., $D_j(\mathbf{x}) = \omega_i$.

Proof

The set of Bayes-optimal discriminant functions are given by:

$$g_i(\mathbf{x}) = \log(P(\omega_i) \cdot P(\mathbf{s}|\omega_i)) \quad i = 1, \dots, c$$

Taking assumption 4.8 into account,

$$\begin{aligned} g_i(\mathbf{x}) &= \log(P(\omega_i)) + \log(P(\mathbf{s}|\omega_i)) \\ &= \log(P(\omega_i)) + \log\left(\prod_{j=1}^L P(s_j|\omega_i)\right) \\ &= \log(P(\omega_i)) + \log\left(\prod_{j \in I(\omega_i)} P(s_j = \omega_i) \prod_{j \notin I(\omega_i)} P(s_j \neq \omega_i)\right) \\ &= \log(P(\omega_i)) + \log\left(\prod_{j \in I(\omega_i)} (1 - E_j) \cdot \prod_{j \notin I(\omega_i)} E_j\right) \\ &= \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} \log(1 - E_j) + \sum_{j \notin I(\omega_i)} \log(E_j) \end{aligned}$$

We can form an equivalent set of discriminant functions $g'_i(\mathbf{x})$ by adding a term which does not depend on the class label i ,

$$\begin{aligned} g'_i(\mathbf{x}) &= \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} \log(1 - E_j) + \sum_{j \notin I(\omega_i)} \log(E_j) - \sum_{j=1}^L \log(E_j) \\ &= \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} \log(1 - E_j) + \sum_{j \notin I(\omega_i)} \log(E_j) \end{aligned}$$

$$\begin{aligned}
& - \sum_{j \in I(\omega_i)} \log(E_j) - \sum_{j \notin I(\omega_i)} \log(E_j) \\
& = \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} (\log(1 - E_j) - \log(E_j)) \\
& = \log(P(\omega_i)) + \sum_{j \in I(\omega_i)} \log \frac{(1 - E_j)}{E_j}
\end{aligned}$$

■

4.5.3 Boosting the margins

One of the most unexpected phenomena seen in boosting experiments is that after the training error reaches zero the test error continues to decrease. In fact, even long after the training error has reached zero, even if the ensemble becomes very large, the test error does not usually increase due to overtraining as might be expected [8, 85, 95]. The main exceptions to this are the empirical results of Wickramaratna et. al., who found that boosting strong learners (radial basis function classifiers) led to performance degradation as the classifiers were forced to concentrate on outliers and the harder examples [113].

To explain this positive feature of boosting, Schapire et. al. used the concept of *margins* [32, 94, 95] originally suggested by Vapnik [110] and further developed in the context of support vector machines in his work with Cortes [16]. The main idea is that it is not enough to consider the training error in analysing performance, it is also necessary to consider how *confident* the classifiers are in their classification. They develop a measure of confidence, the *margin*, which allows them to prove that any improvement in the margin on the *training* set will *guarantee* an improvement in the upper bound of the testing error.

Suppose that we have an ensemble of base classifiers and their associated combination weights, normalised so that they sum to unity. For a particular example we have to consider the sum of weights for each class label. Schapire et.al. define the classification margin as follows [95]:

Definition 1 *The classification margin for an example is the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label.*

The margin is therefore a number in the range $[-1, 1]$ and to be correctly classified the margin of an example must be positive. The larger the margin, the more confident the classification. So a large, positive margin indicates a confident, correct classification. Schapire et. al. found that both bagging and boosting tend to result in an increase in the margin of examples and that they converge to a situation in which most examples have large positive margins [95]. Boosting, by its very nature, is aggressive in its tackling of examples with small margins since it actively focuses on these examples in subsequent rounds. This concept of margins is a possible explanation for why AdaBoost continues to

drive the testing error down even after the training error has reached zero. Schapire et al.'s experiments show that maximising the margins usually results in better generalisation error. Long after the correct classification has been reached the margins are still being increased giving more and more confidence to the classification and resulting in better generalisation ability.

The margins can be used in conjunction with algorithms other than AdaBoost as in the work by Hoche and Wrobel. They study the margins of training examples in order to *actively* determine the best number of features to include to get the right balance between speed and accuracy [42]. They develop an algorithm which monitors the average of the training sample margins as learners are built. When the actual improvement in the margin decreases below their expected improvement they add the next feature.

Kleinberg considers that boosting algorithms are enforcing uniformity by de-emphasising easy examples [49]. By combining the weak learning iteration part of the AdaBoost algorithm with a stochastic-discriminant algorithm they confirmed this theory experimentally.

4.6 Existing empirical studies about AdaBoost

There are many experimental studies investigating various aspects of AdaBoost. Figure 4.6 shows some of these studies and the aspect of AdaBoost with which they are concerned.

4.6.1 AdaBoost and modifications

Schapire [92] gives an overview of boosting. He highlights some of the main features of AdaBoost and shows some of the more important empirical results to date. These results show that AdaBoost generally performs as well as, if not significantly better than other methods. Schapire also highlights that one of the problems with AdaBoost, its overemphasis of weight on certain examples, can in fact be turned to our advantage as a method of identifying outliers. Since AdaBoost concentrates weight on the harder examples, and outliers are hardest to classify, then those with the largest weight are often outliers.

Kuncheva and Whitaker compared three different versions of AdaBoost an aggressive version, a conservative version and an inverse version [61]. They also looked at the possibility of studying diversity, in this case the Q statistic, in conjunction with AdaBoost to identify the paralysis stage found by Wickramaratna in [113]. The aggressive version increased weights on incorrectly classified examples and decreased weights on those correctly classified examples. The conservative version actively changes only one of these, either increasing weights on the incorrectly classified examples or decreasing weights on the correctly classified examples. The inverse version is opposite to the aggressive version, decreasing weights on incorrectly classified examples and increasing weights on correctly classified examples. This third approach may seem odd but it was included to see whether

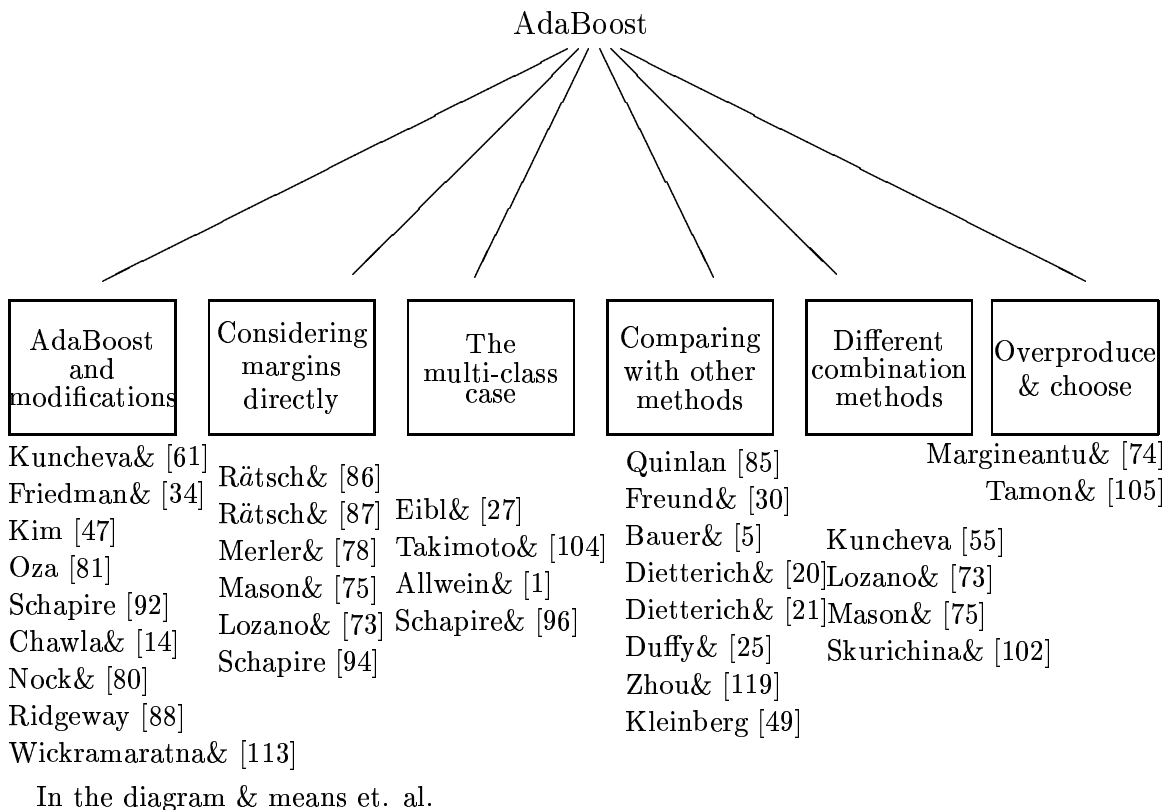


Figure 4.2: THE VARIOUS ASPECTS OF, AND STUDIES INTO, ADABOOST

actively avoiding overtraining by too much emphasis on incorrect examples would lead to better results. They found that in fact this was not the case, with the conservative approach having the most accurate results. They also found that the relationship between Q and accuracy was not particularly clear and could probably only be used to give a guide to the general trend in performance.

Friedman, Hastie and Tibshirani found that AdaBoost fits an underlying model, that of additive logistic regression [34]. They also prove that after each weight update in the AdaBoost algorithm the weighted misclassification error of the most recent classifier is 50%. This may be another reason why AdaBoost does not overfit more often. They use the reweighting implementation version of AdaBoost and determined that there is little connection between this deterministic reweighting version of AdaBoost and other randomized ensemble methods such as bagging and randomized trees. They compare four versions of AdaBoost using eight datasets from UCI and a decision tree as the base classifier.

- Discrete AdaBoost-Freund and Schapire's [30] AdaBoost where the classifiers, $D(\mathbf{x}) : \mathbb{R}^n \rightarrow \{-1, +1\}$.
- Real AdaBoost-Schapire and Singer's [96] two algorithms using confidence rated predictions where $D(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ with the sign of $D(\mathbf{x})$ giving the classification and the size of $|D(\mathbf{x})|$ giving the confidence in the prediction. There are two algorithms because we need one for the two-class case and the other for the multi-class-case and are used with datasets as appropriate.
- LogitBoost-fits additive logistic regression models by stage-wise optimisation of the Bernoulli log-likelihood. Again they have two versions, one for the two-class case and one for the multi-class case.
- Gentle AdaBoost-works like Real AdaBoost but uses Newton steps rather than exact optimisation at each step.

They found that the Real AdaBoost, LogitBoost and Gentle AdaBoost algorithms were on a par with each other, with the Discrete AdaBoost algorithm being slightly worse. In most of their experiments the error rate of Discrete AdaBoost was twice that of the other algorithms. They then proceed to introduce a slight modification to the algorithms to speed up computation running time. They do this by considering the weight of each training observation and only submitting those observations with weight greater than a certain threshold to the base classifier at each round. The weights are still updated for all of the training observations though, so that a particular observation can be discarded from some of the iterations but can be brought back in again at a later stage if its weight increases. Experiments using this show that very large computation reductions can be achieved using this simple adaptation. Another interesting point that they highlight is with the resampling implementation of AdaBoost. The common practice is to use the original sample size N to determine the bootstrap size. They identify that there is in fact no evidence to suggest that this is optimal in all or in any situation. This is something which we will look at in more detail later on.

As has been found in other studies, one of AdaBoost biggest weaknesses is with noisy data, when it can concentrate too much effort on outliers and mislabelled data. Kim develops a new boosting algorithm, Averaged Boosting, which is more robust to noise than AdaBoost [47]. The difference between them is that AdaBoost uses the *product* of the classifiers and their coefficients whereas Averaged Boosting uses the *average of the product* of classifiers and their coefficients when updating the weights. The averaging means that not all the correctly classified examples are ignored and therefore not all the effort is concentrated on a few outliers and mislabelled examples in noisy situations. Kim carried out experiments on fourteen datasets from UCI using decision trees as the base

classifier. The results showed that Averaged Boosting was more robust than AdaBoost to noise and also outperforms bagging in low noise situations. Also, Averaged Boosting was comparable to bagging in high noise cases.

Oza also looked at using averaging to increase stability and improve boosting performance [81]. The algorithm AveBoost takes into consideration all of the previous base classifiers distributions when calculating the next iteration's weight distribution rather than just the immediately preceding distribution. This is achieved by initially calculating the distribution as for AdaBoost but then each element is averaged with the distributions from all previous iterations. Their experiments showed that this leads to results that are on a par with or better than those for AdaBoost on 9 datasets from UCI.

Chawla et. al. noted that both boosting and bagging can have limitations on very large datasets [14]. This is because computation time becomes prohibitive. One solution to this is to take a subset of the data to be representative of the whole, but how do we ensure that vital information is not lost in the discarded data. They bypass this problem by partitioning the large datasets into random and disjoint subsets which are then distributed to different processors. As each processor is working independently the computation time is minimised and, hopefully we can create diverse classifiers. The classifiers produced are then combined by majority vote ensuring that we are utilising all available information. Their experiments were on three smaller (data sizes: 6435, 10992, 20000) datasets and one larger datasets (training size:209529, testing size:17731). Their results showed that this is a potentially very useful approach for very large datasets which cannot be practically handled by a single processor.

Nock and Sebban investigate AdaBoost and give a theorem proving the efficiency of AdaBoost and an explicit version of the theorem of the upper bound of its training error [80]. They prove that optimising each weak hypothesis on a criterion which they call Z (and which is not accuracy) optimises the overall hypothesis. Using this idea they introduce their own boosting algorithm which they apply to the feature selection problem. It follows a feed-forward search method and adds the feature at each stage which increases the accuracy most when run using the boosting algorithm. If none of the potential features increase, they stop and use the current subset. They found that using this feature selection with boosting approach was better than using the whole feature set with neural networks on 13 out of 19 datasets, mostly from UCI.

Ridgeway also looks at an alternative area for boosting to be used [88]. By viewing AdaBoost as a solution to the problem of minimising a loss function, boosting can become applicable to a variety of other problems such as non-linear regression, robust non-linear regression and non-linear exponential family and survival regression. He considers boosting to be equivalent to gradient-based functional optimisation algorithms that are able to fit non-linear functions to data.

4.6.2 Considering margins directly

Rätsch, Onoda and Müller found that AdaBoost overfits with data where there are high noise levels since it asymptotically achieves a hard margin [86]. AdaBoost concentrates too much on a few hard-to-learn patterns and in a noisy situation this is clearly suboptimal behaviour. They propose a solution by introducing a note of mistrust in the data so that AdaBoost can ignore outliers and mislabelled data. They introduce several regularisation methods and generalisations of AdaBoost to achieve this. This soft approach is similar to, and inspired by, the route which people working with support vector machines have followed. Rätsch continues his work with AdaBoost in conjunction with Warmuth by developing the AdaBoost* algorithm [87]. This algorithm is able to explicitly maximise the minimum margin of the training examples up to a given precision. Thus it maximises the margin on separable classes and minimises the overlap between any two classes on inseparable cases. They compare AdaBoost* with standard AdaBoost using C4.5 decision trees and find that their modifications do indeed lead to an increase in the margins.

Merler et. al. adapt AdaBoost in their experiments to deal with problems where there is a cost-sensitive issue [78]. They consider the two-class case for cancer diagnosis. Obviously a false negative diagnosis has much more serious implications than a false positive diagnosis. Thus they modify AdaBoost to update the weights differently for incorrect classification of different classes. They are actively increasing the margins of positive examples without modifying the margins of negative examples.

Mason, Bartlett and Baxter decided that since AdaBoost implicitly seems to be maximising the margins it may be possible to improve performance by trying to explicitly optimise the margins [75]. They develop the Direct Optimization Of Margins (DOOM) algorithm to do this. It works by using AdaBoost to generate a set of base classifiers before DOOM finds the optimal combination weights. They carried out experiments on ten datasets from UCI and on all but one dataset the combination produced by DOOM had lower testing error than using AdaBoost's weighted combination. Their investigation shows that this improved testing performance is achieved by sacrificing the training error rate. Also their plots of the margins suggest that the size of the minimum margin is not the critical factor in the generalisation performance.

4.6.3 The multi-class case

The version of AdaBoost which we have described and which we will be studying further is the two-class version. However there are other versions which are extensions to the multi-class case. Some of these rely on reduction of the problem to a series of binary problems. AdaBoost.M1 is a version which deals directly with the multi-class case but it requires the base classifiers to have accuracy of greater than $\frac{1}{2}$ [31]. This is reasonable for the

two class case, where this is simply a requirement that the base classifiers be better than random guessing, but as the number of classes grows it becomes more and more difficult to achieve. To deal with this problem Eibl and Pfeiffer have modified AdaBoost.M1 by changing a single line of code [27]. This has resulted in an algorithm which guarantees to minimise the upper bound for their performance measure as long as the base classifier is better than random guessing, i.e., better than $\frac{1}{c}$ with c classes. They also modify the stopping criterion to be softer.

Schapire and Singer examined several generalisations for multi-class problems [96]. Their experimental results with these improved boosting algorithms showed that it is possible to achieve dramatic improvements in the training error when there is a reasonably large amount of data. On small, noisy datasets, however, the rapid decrease in the training error is often linked to overfitting with resulting degradation in the generalisation error.

Takimoto and Maruoka also developed an information-based boosting algorithm which is able to tackle the multi-class problem without having to reduce it to a set of binary problems [104]. Unfortunately in experiments, they found that their algorithm's performance was worse than other boosting algorithms such as AdaBoost.

Allwein, Schapire and Singer consider the problem of the multi-class case with margin-based classifiers, and look at how it might be possible to reduce it to a set of binary problems [1]. They use AdaBoost and support vector machines to build the base classifiers using eight datasets from UCI and combine their outputs using decoding techniques which are similar in nature to error-correcting output codes. Interestingly they also prove that having mostly large margins for the training set implies that the generalisation error has improved bounds independent of the number of rounds of boosting. That is the bounds are tighter than those given by Schapire et. al. in [95].

4.6.4 Comparing with other methods

Quinlan carried out his investigation into bagging and boosting with C4.5 decision trees in 1996 which has now become a benchmark study into AdaBoost's performance [85]. His experiments involved twenty-seven datasets from the UCI machine learning repository. They compared the reweighting implementation of AdaBoost with bagging using ten base classifiers in the ensemble. In this early investigation the AdaBoost algorithm stops when the training error reaches zero. He found that both bagging and AdaBoost substantially decrease the testing error with boosting generally better than bagging. However boosting can produce severe degradation with some datasets. He then changed the voting weights to a confidence measure in the classification of the training example and this led to some improvement.

Following on from Quinlan's work Freund and Schapire carried out experiments com-

paring bagging with AdaBoost [30]. They also compared their performances on twenty-seven datasets from UCI. They also looked in more detail at an algorithm which combines AdaBoost with a nearest-neighbour classifier. Their results showed that boosting is better with simple classifiers and is possibly helpful when observed examples have varying degrees of hardness, or when the learning algorithm is sensitive to changes in the training set. They deduced that this was because the pseudo-loss version of AdaBoost concentrates not only on the hard to classify examples but more specifically on the incorrect labels which are hardest to discriminate. They were particularly surprised by the continued reduction in the testing error after the training error reaches zero.

Bauer and Kohavi also compare bagging and AdaBoost and some of their variants in real-world experiments designed to look at the bias-variance decomposition [5]. They found that bagging and its variants always improved the performance even if only very slightly. They also found that boosting did not always improve performance and that it depended on the data. However, when boosting did improve it did so significantly. They also found that boosting results showed a higher variance but lower bias than bagging and that AdaBoost does not deal well with noisy data.

Dietterich has also performed several investigations into AdaBoost, comparing it with other methods of constructing ensembles such as bagging and randomization. Randomization involves computing the twenty best splits at each internal node of a decision tree and then choosing one randomly. In experiments comparing bagging, boosting and randomization using C4.5 trees, he found that if there is little or no noise, boosting performs best and randomization may be slightly better than bagging [20]. However, in high noise cases (20%) bagging is much better than boosting and sometimes better than randomization. Also in large data sets bagging does not alter the training sample much and so there is not much improvement over the single C4.5 tree. In further experiments he identifies three reasons why an ensemble may work better than a single classifier [21]. These are:

Statistical: if we have a set of decision rules with similar accuracy then we can average them to reduce the risk of choosing the ‘wrong’ rule.

Computational: decision rules can get stuck in local optima, so by starting from different points we can provide a better approximation to the ‘true’ decision rule.

Representational: often the true hypothesis cannot be represented by our available rules, thus a weighted sum of decision rules can expand the space of representable functions.

Dietterich determined that bagging and randomization both predominantly tackle the statistical and to some extent the computational problem. AdaBoost however, tackles the representational problem and this is the reason he believes it to be fundamentally different

in approach to bagging and randomization. We also know that AdaBoost tends to overfit in high noise cases but it does not overfit as often as we might expect. Dietterich believes this is due to its stage-wise nature and lack of backfitting; it does not return and modify existing hypotheses or combination weights as it progresses. His experiments confirm these conclusions.

Duffy and Helmbold use different terminology, they use the phrase ‘leveraging algorithm’ to describe any learning algorithm which produces a combined hypothesis by iteratively calling a black-box learning routine to produce the individual hypotheses to be combined [25]. AdaBoost is therefore a leveraging algorithm and they introduce their own new leveraging algorithm in this paper. Their analysis suggests that their algorithm is likely to be better than AdaBoost on noisy data which is a known problem for AdaBoost. On a set of twelve, small, two-class datasets from UCI their algorithm and AdaBoost were comparable. They also carried out a second set of experiments involving UCI’s LED artificial data, which allows the addition of attribute noise, and with the mushroom and chess datasets where they flipped some labels to introduce noise. They found that for between 10 and 20% noise their algorithm is significantly better than AdaBoost.

Zhou, Wu and Tang have designed an overproduce and choose strategy called GASEN which is based on their theory that if there are many neural networks available it might be better to use only a subset [119]. Overproduce and choose strategies are based on the idea that it may be better to build a large number of classifiers and then select in some way the ‘best’ subset of them. The difficulty is thus how to define what we mean by ‘best’ and how to choose amongst all possible subsets. Zhou et. al. show equations which prove that it may be better to use a subset of neural networks and which could be used to identify which neural networks to omit from the ensemble. However, this method would be prohibitively computationally expensive for real-world applications. GASEN is one possible algorithm designed to help with this problem. Their experiments compare GASEN with bagging and AdaBoost using resampling on ten datasets from UCI. They found that GASEN not only generates smaller neural network ensembles but also has a stronger generalisation ability.

Kleinberg compared stochastic discrimination to boosting and bagging with three different underlying weak learning algorithms FindAttrTest, FindDecRule and C4.5 [49]. Experiments with 17 datasets from UCI showed that stochastic discrimination outperformed all the methods on 14 datasets. Also on 5 out of 7 datasets from Statlog stochastic discrimination outperformed 23 different combination methods.

4.6.5 Different combination methods

Kuncheva also looked at using AdaBoost to build the base classifiers and then combining them in an alternative way [55]. She looked at various combination methods comparing fuzzy methods (fuzzy integral and decision templates) with some of the non-fuzzy methods we have already seen (majority vote, minimum, maximum, average, product, naive Bayes) and weighted majority the standard method used with AdaBoost. Experiments carried out using three datasets available from UCI, two from ELENA and the author's own artificial dataset, found that the simple combining methods of minimum, maximum, product and average gave particularly poor results. Majority vote and naive Bayes had erratic performances and weighted majority, although stable, was slightly worse than the fuzzy combination methods.

Lozano and Koltchinskii also use AdaBoost to create a set of classifiers before combining them using their own set of weights rather than those provided by AdaBoost [73]. They use an independent validation set in order to actively search for the set of weights which minimises the error on combining. Their algorithm involves considering the margin of those examples in the validation set. It is called DOOM-LP since it is directly optimising the margins as DOOM does, but the solution also involves solving a sequence of linear programming problems. Experiments showed that with eight out of ten datasets from UCI, DOOM-LP improved over using AdaBoost's normal combination weights. It is therefore on a par with DOOM's performance, but at a lower implementation cost.

Skurichina and Duin study how different combination methods affect the performance of both bagging and boosting [102]. They carried out experiments using two artificial and one real dataset. Their results show that bagging is useful with unstable classifiers with critical training sample sizes, if the sample size is too small or too large bagging does not work as well. Also combining the classifiers from bagging with weighted majority vote, weighted average or product all produce better results than using simple majority vote. With boosting, the larger the training sample size the better and it does not depend so much on the instability of the base classifiers. Also boosting has generally better performance than bagging while the combining rule used is less important than in bagging. They conclude that the combining rule which is best is data-dependent and depends upon the training sample size but it is important. Also simple majority voting, which is usually used with bagging, tends to be the worst choice.

4.6.6 Overproduce and choose

Margineantu and Dietterich have taken this use of AdaBoost one stage further in an overproduce and choose strategy [74]. AdaBoost is used to build a large set of classifiers before a pruning algorithm is used to choose a subset of classifiers to form an ensemble.

This approach is aimed at reducing the memory requirement and the computation costs without a serious loss of performance. They compared five different pruning algorithms on ten datasets from UCI using the resampling version of AdaBoost with C4.5 decision trees as the base classifiers. The combination weights from AdaBoost are used to combine the final, pruned ensemble as usual. Their results show that despite reducing the number of classifiers in the ensemble it is possible to obtain a nearly comparable performance to that obtained using the entire set. They also introduce the kappa-error diagram which gives a way of visualising the trade-off between accuracy and diversity and which we shall be considering further in the next chapter.

Tamon and Xiang's work [105] follows on from Margineantu and Dietterich's work with pruning algorithms using one in particular - kappa pruning [74]. They use the reweighting version of AdaBoost building C4.5 decision tree classifiers. Once a classifier has been pruned from the ensemble their modification involves distributing the weight associated with it between all the remaining classifiers. They view this as a clustering-like process, in that the weight is distributed based upon the proximity of the other classifiers to the pruned classifier. Thus all the weights of the original ensemble are incorporated in the weighted voting to give the final hypothesis. Experiments on eight datasets show that this weight shifting can help improve the kappa pruning error rates sometimes. They then proceed to look more theoretically at how we can prune a subset of classifiers from the whole set in such a way as to minimise the training error of the ensemble. They do this by establishing a matrix where entry i, j is the margin of h_i on example j and trying to develop an heuristic which can reduce the number of hypotheses (rows) without reducing the sum of the margins in each column below a certain threshold. They achieve a semi-feasible solution by turning this problem firstly into an integer programming problem and then into a linear program problem.

4.6.7 Summary of Existing Studies of AdaBoost

AdaBoost often has significantly better performance than other boosting methods. It can also be used to identify outliers since it places additional weight on those cases most difficult to classify which often turn out to be outliers [93]. After normalisation and updating of weights the weighted error of the most recent classifier on the data is 50% making each classifier likely to be different from the previous classifier [34].

AdaBoost is weakest on noisy data where it often performs considerably worse than other methods of classification. Some modifications to bring an averaging element into the AdaBoost algorithm were found to make it more robust with noisy data [47, 81]. AdaBoost also has problems with very large data sets. These can be dealt with by using a set of processors each working on a disjoint partition of the data to build an ensemble of

classifiers separately before combining them using majority vote [14]. An alternative approach to deal with the noisy data problem is to build a level of mistrust into the AdaBoost algorithm [86]. This led to explicitly maximising the minimum margin of the training examples up to a certain precision [87].

The margins can also be used to deal with cost issues such as a difference in the consequences of false positive and false negative results. This is done by actively increasing the margins on either positive or negative examples as required [78]. This led to the DOOM algorithm which seeks to directly optimise the margins [75].

For the multi-class case there has been some work in breaking the problem down to a series of binary problems [1] as well as algorithms directly dealing with the multi-class format [31, 104]. An original requirement of the AdaBoost algorithm was that the base classifiers had to have error of less than 50% [31], which is obviously restrictive for the multi-class case as the number of classes grows. Consequently there have been some modifications of the algorithm to deal with this [27].

There are many investigations comparing AdaBoost to other classification strategies [5, 20, 21, 25, 30, 49, 85, 119]. They found that bagging always improves the performance and that boosting is generally better than bagging. However it can produce severe degradation with some datasets. Boosting is better with simple classifiers and on data with little noise. For high noise cases bagging is much better than boosting and is also sometimes better than randomisation.

There are also several studies into using alternative combination methods after AdaBoost has produced an ensemble of classifiers rather than the usual weighted votes [55, 73, 103]. They show that fuzzy combination methods [55] or actively searching for an alternative set of weights which minimise the error [73] both produce better results than the standard weighted majority using weights provided by the AdaBoost algorithm. Skurichina concluded that the best combination method was not only data-dependent but also depended on the sample size of the training data [103].

Further studies have also shown that it is possible to use AdaBoost to build a large set of classifiers and then use a pruning algorithm to reduce the size of the ensemble, without dramatically altering the generalisation error [74, 105]. We shall be looking at this *Overproduce and Choose* approach in the next chapter.

4.7 Experimental set-up into how AdaBoost affects Classifier Diversity

This section is based on a study presented at IPMU 2002 [98]. We are interested to see whether there is a link between diversity and accuracy of ensembles built by AdaBoost. We try to establish whether or not we could use the change in diversity as classifiers are

added to get the most benefit from AdaBoost.

For our experiments we used the Pima Indian diabetes database and the Haberman survival database, both taken from the UCI repository of machine learning database. Table 4.1 shows a summary of the data sets. The data sets are described in more detail in Appendices B.9 and B.3.

Table 4.1: SUMMARY OF THE DATA SETS

Name	No. Classes	Size of data set	No. Features
Pima Indian diabetes	2	768	8
Haberman Survival	2	306	3

By using AdaBoost with 10-fold cross-validation we were able to produce ensembles of 100 classifiers. We used three types of classifiers: linear, quadratic and neural networks. The neural network consists of a single hidden layer of 15 neurons with a maximum of 300 training epochs.

As each new classifier was added to the ensemble we considered

1. the training accuracy of the ensemble
2. the testing accuracy of the ensemble
3. the training diversity of the ensemble using the Q -statistic introduced earlier (3.1.1).

For each classifier type and dataset, with both training and testing sets, we plotted the average change in the error against the number of classifiers. We also plotted the average change in training Q against the number of classifiers. We were interested in finding out whether Q can be used as a stopping criterion for AdaBoost. Therefore we identified the minimum training Q in each experiment. Our experiments showed that rather than rapidly reaching zero our training error actually fluctuated in value and so we also monitored the minimum training error. We considered these as potential stopping criteria, called ‘min Q ’ and ‘min TRE ’ respectively, and recorded their corresponding testing error.

As we have already seen, AdaBoost is not particularly robust to noisy data [20, 21, 25, 47, 86, 92] and we have subsequently found that Pima has some outliers which can lead to AdaBoost becoming overtrained [9]. It is not known whether Haberman also has significant outliers but if it does this may be the reason that the training error did not rapidly reach zero in our experiments as we would have expected.

4.8 AdaBoost and Classifier Diversity Results

Figures 4.3, 4.4 and 4.5 show how the average training and testing errors change, and how the average training Q changes as AdaBoost adds classifiers to the ensemble. The top, left-hand graph in each figure shows the change in error versus the number of classifiers for the Haberman data. The top, right-hand graph shows the change in error versus the number of classifiers for the Pima data. The bottom, left-hand graph shows the change in Q for the Haberman data and the bottom, right-hand graph shows the change in Q for the Pima-Indian data. The thick, blue lines indicate the training run and the thin, red lines the testing run. Figure 4.3 shows the results for linear classifiers, Figure 4.4 the results for quadratic classifiers and Figure 4.5 the results for neural networks.

Looking at the graphs we can see that the testing error is lower for the Pima data than the Haberman data although the difference is marginal for quadratic classifiers. Recall that AdaBoost is supposed to rapidly drive the training error to zero, however we see that with linear classifiers the training error is almost horizontal. For quadratic classifiers the training error is almost horizontal for the Haberman data and initially decreases before levelling off with the Pima data. We see slightly more of a decrease with the neural network classifiers before starting to level off. This is similar to the paralysis observed by Wickramaratna [113].

When we consider the testing error we see that for linear classifiers and for quadratic classifiers with Haberman data the graph is close to the training error and is almost horizontal. For quadratic and neural networks with Pima, the graph follows the same shape as the training error but is higher in value. For neural networks with the Haberman data we see that the testing error is considerably higher and fluctuates erratically suggesting overfitting.

If we look at the lower, Q , graphs we see that for linear classifiers and quadratic classifiers with Haberman data there is almost no change in the value of Q with it being very close to 1. With quadratic classifiers and Pima data there is an initial decrease before the Q value levels off at approximately 0.8. These results suggest that the classifiers that are being added to the ensemble are all very similar and there is therefore not much to be gained by boosting them. This was reflected in the horizontal training error graphs.

For the neural network classifiers we see that the Q value rapidly decreases as the first 20 classifiers are added to the ensemble. After this point it continues to decrease but quite slowly as the rest of the classifiers are added until it reaches a value of approximately 0.2. This suggests that the neural network classifiers are more diverse in the ensemble than with linear and quadratic classifiers, and the ensemble becomes more diverse as the ensemble grows. This seems to explain why the neural network training error reduced whilst the quadratic and linear classifiers training error did not.

From Figures 4.3, 4.4 and 4.5 we can say that neither linear nor quadratic classifiers benefit from boosting using AdaBoost, but neural networks may be more suitable. We could have obtained the same level of testing error by using the first few classifiers for both linear and quadratic classifiers.

Ideally we would like diversity to be more closely related to the training error. We had hoped that the ‘elbow’ point on the Q graph would correspond to a similar turning point on the training error graph which would be where the error would slow down its decreasing. Then we could identify a suitable point at which to stop the AdaBoost algorithm. Unfortunately there does not appear to be a close enough relationship between the training error and Q to do this. Previous studies suggest that other diversity measures would not be much different from Q [92].

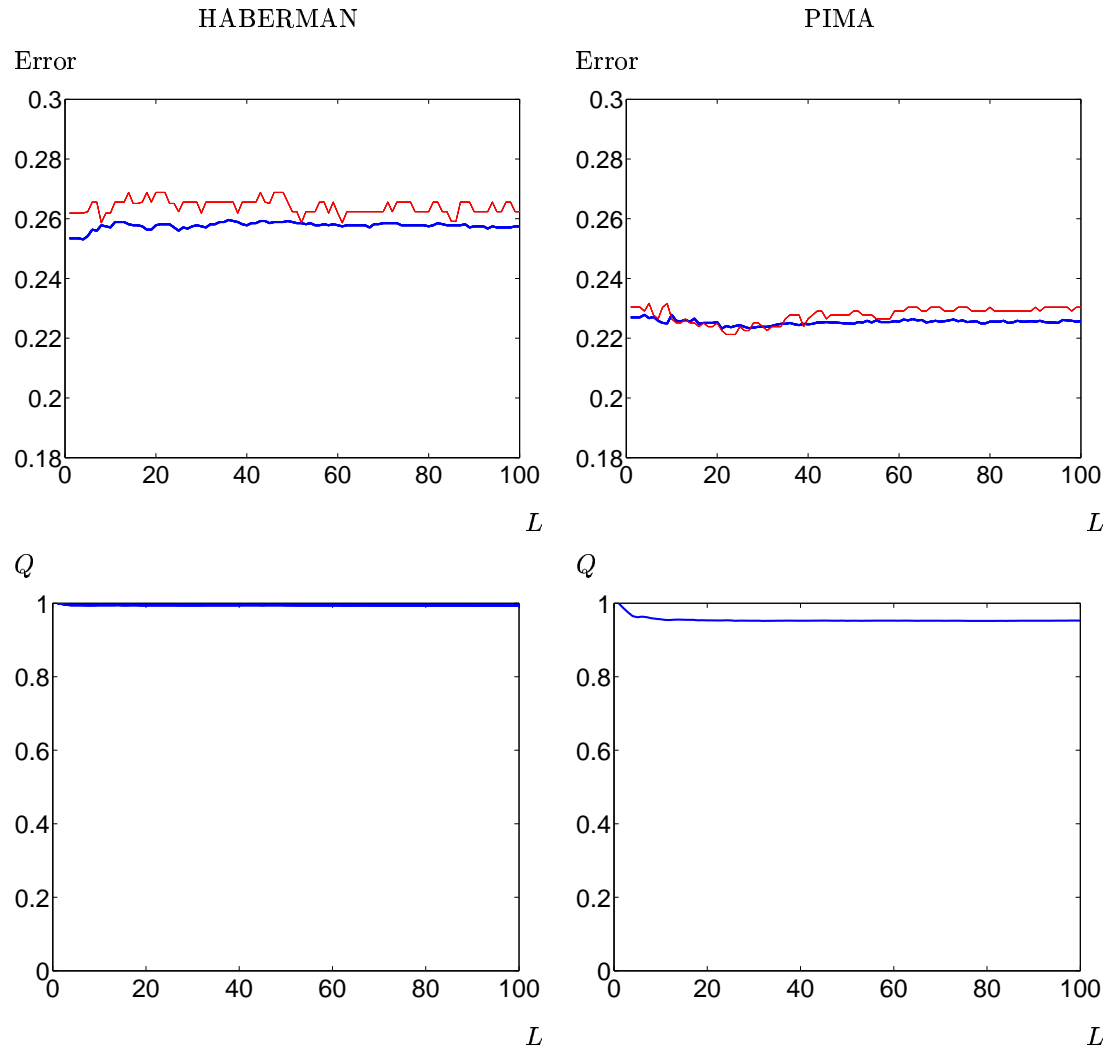


Figure 4.3: CHANGE IN THE AVERAGE ERROR AND VALUE OF Q AS WE ADD LINEAR CLASSIFIERS TO THE ENSEMBLE. THE THICK, BLUE LINE IS THE TRAINING DATA AND THE THIN, RED LINE IS THE TESTING DATA

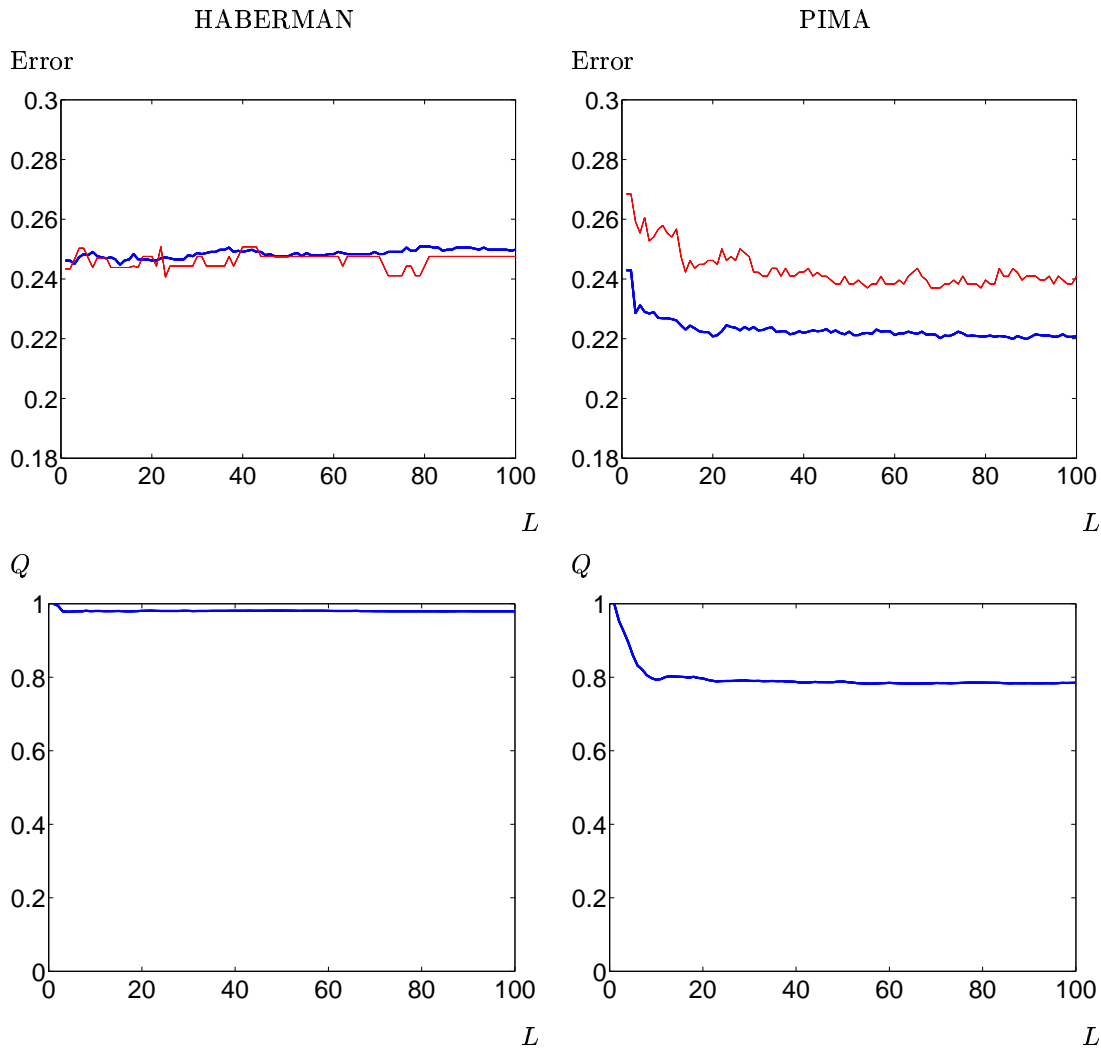


Figure 4.4: CHANGE IN THE AVERAGE ERROR AND VALUE OF Q AS WE ADD QUADRATIC CLASSIFIERS TO THE ENSEMBLE. THE THICK, BLUE LINE IS THE TRAINING DATA AND THE THIN, RED LINE IS THE TESTING DATA

Figure 4.6 shows the average training and testing errors versus the average training Q for all data sets and all base classifiers. The blue line in the upper graph shows the training error and the red line in the lower graph shows the testing error. The ensemble consisting of a single classifier is at the far right at $Q=1$. As each classifier is added to the ensemble we plot the new value of Q and the error corresponding to it. The lines join these plots together and show how Q and the error changes as the ensemble grows.

The lines have a general trend going to the left and downwards indicating that both the value of Q and the errors decrease as the ensemble grows. The black circles and arrows indicate where the minimum value of Q occurs and the pink circles and arrows indicate where the final $N=100$ classifiers occurs. These show that there is not much difference in the error observed at $\min Q$ and at $N=100$, but the $\min Q$ could reduce the number of

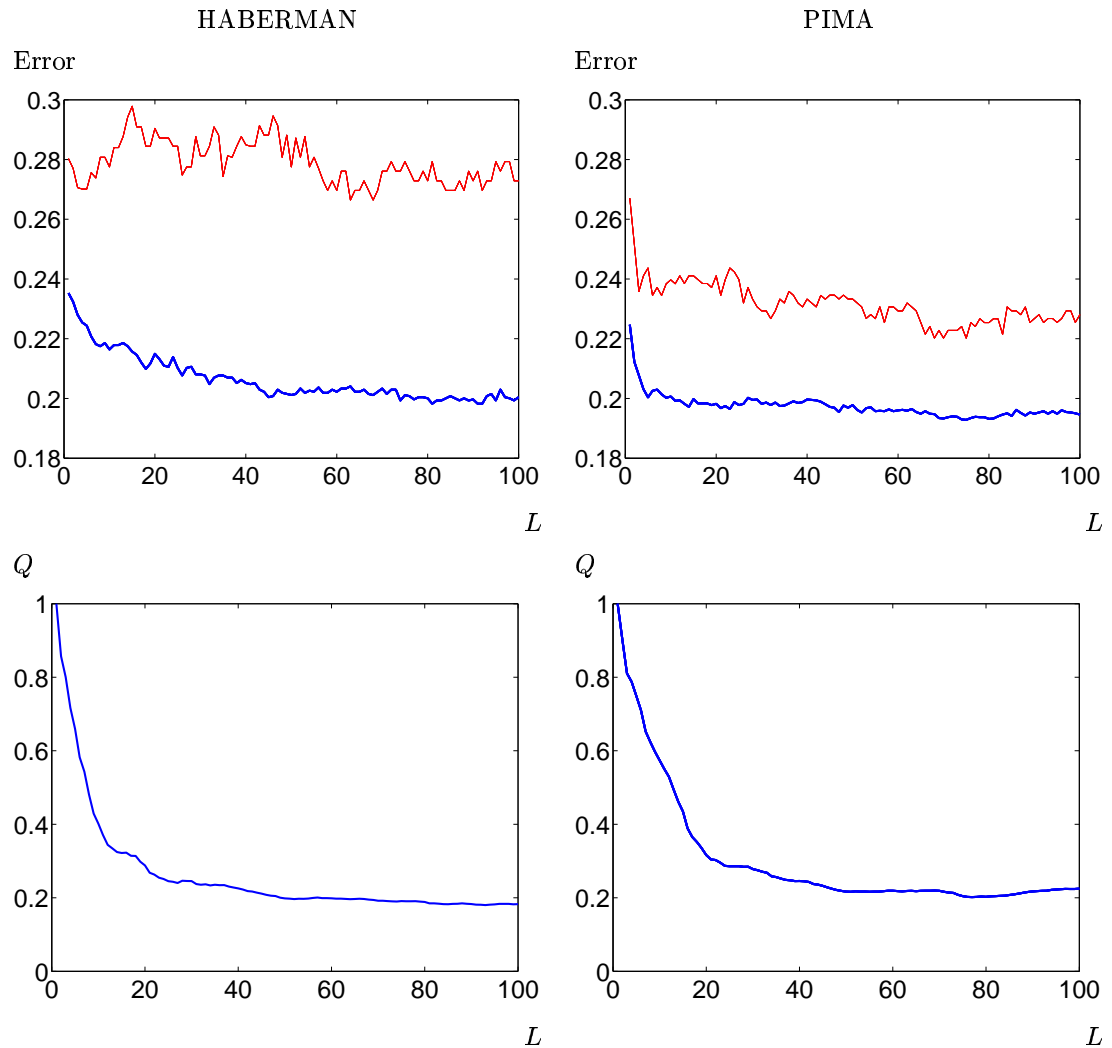


Figure 4.5: CHANGE IN THE AVERAGE ERROR AND VALUE OF Q AS WE ADD NEURAL NETWORKS TO THE ENSEMBLE. THE THICK, BLUE LINE IS THE TRAINING DATA AND THE THIN, RED LINE IS THE TESTING DATA

iterations taken. The minimum testing error actually occurs before the $\min Q$ is reached and is not related to the value of training Q .

We can see that there is little improvement in generalisation error gained by stopping the algorithm at the $\min Q$ point, however there is a computational advantage in that we terminate at an earlier point reducing the number of iterations.

We now examine whether the computational advantage of terminating at an earlier point using $\min Q$ is consistent in all cases of our experiments.

Table 4.2 shows the average number of iterations taken, (the number, L , of classifiers in the ensemble) and the corresponding test error obtained using three criteria to determine when to terminate AdaBoost, averaged over 10 independent runs. The stopping criteria

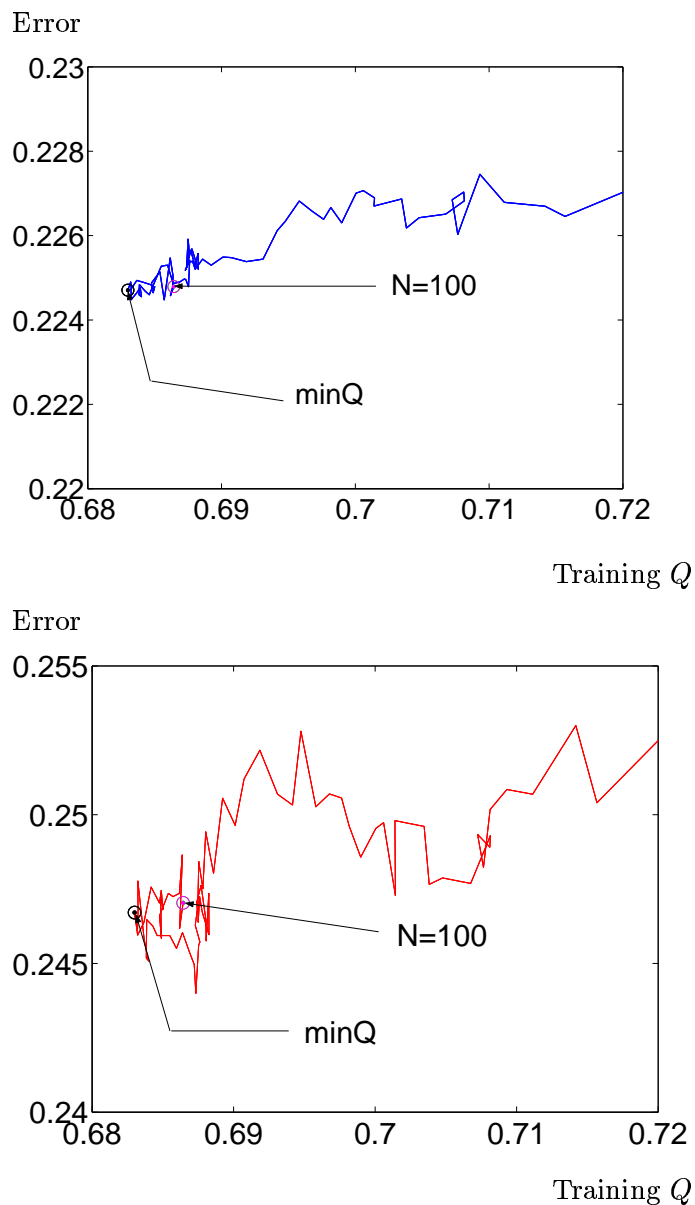


Figure 4.6: AVERAGE ERROR VERSUS AVERAGE TRAINING Q FOR ALL DATA SETS AND BASE CLASSIFIERS. THE BLUE LINE IN THE UPPER GRAPH IS THE TRAINING ERROR AND THE RED LINE IN THE LOWER GRAPH IS THE TESTING ERROR. THE BLACK CIRCLES INDICATE THE MINIMUM Q AND THE PINK CIRCLES INDICATE THE POINT WHERE ALL 100 CLASSIFIERS HAVE BEEN BUILT

used were: using all 100 classifiers (All), the minimum training error ($\min TRE$) and minimum Q ($\min Q$). The best performances for each case are shown in bold.

The results show that all three criteria give a similar performance. Using all 100 classifiers and $\min Q$ are on a par with each other, with $\min TRE$ slightly worse than both of them. However, $\min Q$ and $\min TRE$ both use less members in the ensemble, especially for linear

and quadratic classifiers. $\min TRE$ in particular uses considerably less than the full 100 classifiers.

4.9 AdaBoost and Classifier Diversity Conclusions

In this chapter we have investigated the diversity of classifier ensembles built using the AdaBoost algorithm. We carried out experiments with two datasets- Haberman survival data with 3 features, 2 classes and 306 patients and Pima Indian diabetes data with 8 attributes, 2 classes and 768 patients. We used ten-fold cross validation, building 100 classifiers each time of three types: linear, quadratic and neural networks to produce a set of results.

Our experiments show that there is no benefit in boosting either linear or quadratic classifiers, but there is a benefit in boosting neural networks using the AdaBoost algorithm. For neural networks we saw that AdaBoost is increasing the diversity of the ensemble (Q

Table 4.2: TESTING ERRORS USING VARIOUS STOPPING CRITERIA

Case	Stopping Criterion	mean L	mean Test Error
Pima Linear Classifiers	All	100	0.2304
	$\min TRE$	11	0.2330
	$\min Q$	42.5	0.2304
Pima Quadratic Classifiers	All	100	0.2410
	$\min TRE$	34.5	0.2436
	$\min Q$	56	0.2462
Pima Neural Networks	All	100	0.2281
	$\min TRE$	67.3	0.2228
	$\min Q$	63.7	0.2254
Haberman Linear Classifiers	All	100	0.2624
	$\min TRE$	17.4	0.2656
	$\min Q$	37.9	0.2619
Haberman Quadratic Classifiers	All	100	0.2476
	$\min TRE$	9.9	0.2503
	$\min Q$	21.4	0.2476
Haberman Neural Networks	All	100	0.2729
	$\min TRE$	68.3	0.2872
	$\min Q$	65.1	0.2771

was decreasing) as classifiers are added to it. This may be due to the unstable nature of neural networks and the changing weights from AdaBoost.

It is possible that future work may consider using the minimum Q either on its own or together with the minimum training error to determine a termination point for AdaBoost in order to increase generalisation performance. We have also found that a reasonably high accuracy can be obtained after the first few iterations. It may also be of use therefore, to try to enhance the performance of a small ensemble further by applying different combination methods on a reasonably accurate and diverse ensemble. Having established from our results that neural networks are suitable for use with AdaBoost, and from other papers that so are decision trees (e.g. [22]), we will only use these, discarding linear and quadratic classifiers from our further work.

Chapter 5

Improving AdaBoost

Once again recall figure 2.2, which showed what we can change in a multiple classifier system. As we have seen in the previous chapter we can alter AdaBoost by manipulating the combination methods (figure 2.2 part A), or by using different types of classifier (figure 2.2 part B). It has been shown that neural networks and decision trees are the most practical base classifier to use with AdaBoost [22]. Also, with the possible exception of fuzzy methods [55], the use of weighted majority to combine the classifiers produced by AdaBoost is on a par with other combination methods available.

There are many works taking AdaBoost as a starting point, trying to improve it and modify it to different applications [25, 47, 74, 75, 78, 80, 81, 86–88, 105]. In this chapter we are interested in whether we can improve on the performance of standard AdaBoost with resampling by modifying some of its characteristics.

The problem of complexity of the base classifier has been considered for decision trees by pruning, but the analogy in neural networks, examining the best number of hidden layers, number of neurons etc., has not. It is important to know whether complex or simple Neural Networks have higher ensemble accuracy on combination. Drucker [22] suggests that the number of hidden neurons is not crucial when used with AdaBoost, as the boosting algorithm compensates for any deficiencies in the constituent networks. This is supposedly due to the fact that AdaBoost only requires the classifiers to have error of less than $\frac{1}{2}$. He decides on the architecture to use by running a few boosting rounds and comparing the error on a validation set with the training error, if they are similar then the architecture is assumed to be reasonable. We feel that it may be beneficial to investigate whether or not the choice of architecture has a greater impact on the ensemble performance than Drucker believes.

The first part of this chapter therefore parallels those studies where the complexity of decision trees in relation to AdaBoost has been studied, for example whether or not to prune the trees [114]. Pruning using a separate pruning set can improve the generalisation

of a single tree [79]. The question is whether the ensemble performance will be improved when the base classifiers are pruned. Dietterich [20] and Windeatt and Ardesir [114] found that there was no significant difference between using a pruned tree and using the full tree.

Recall that decision stumps are the simplest type of decision tree with only one split at the root node partitioning the data into two disjoint classification regions [19]. It is often possible to use an exhaustive search method to identify the best critical value for decision stumps, which would not be possible with more complex trees. It has been found that ensembles using stumps may work very well with certain distributions such as additive logistic models [34]. However, they may not work well if the distribution is not of the desired type and so it may be preferable to use more complex base classifiers in order to have a wider class of distributions for which we are guaranteed a good performance [6]. The case for decision stumps is that they are easy to implement and for most real-world cases they are powerful enough. However in less-common situations dependencies in the data may require more powerful and therefore more complex decision trees [33].

Clearly there would be an optimal pruning value (tree size) for every problem. We would expect a similar argument to hold for the neural network size. We are interested in whether AdaBoost benefits from a small, large or randomly mixed complexity of the constituent neural networks. The hope is that instead of trying to match the neural networks to the data, there could be a more general recommendation about the neural network complexity. In the first part of this chapter we consider modifying the complexity of the neural network base classifier and the size of the training sets sampled for each new classifier.

Overproduce and choose is a strategy which is becoming widely used in association with combining methods [37, 89, 117]. It involves building a large number of base classifiers and then discarding some of them according to one of a variety of algorithms. This is done in the hope of producing a smaller ensemble of classifiers which does not significantly increase the ensemble error but takes considerably less computation time. In the second part of this chapter we consider whether we can improve on the AdaBoost ensemble's performance by following an overproduce and choose strategy, which reduces the size of the ensemble based on a combination of the error and diversity of the component classifiers.

5.1 Modifying AdaBoost

Drucker found that boosting neural networks outperforms boosting decision trees [22]. We consider the resampling implementation of AdaBoost (discussed in more detail in section 4.5.1) using Neural Networks as our base classifiers for our first set of experiments. The standard resampling implementation of AdaBoost, like that of Bagging, takes boot-

Table 5.1: DATA USED IN THE EXPERIMENTS

Data	Size	# Classes	# Features	Training Size N	Testing Size	Random Range
Phoneme	5404	2	5	540	4864	270 - 1620
Pima	768	2	8	691	77	345 - 2073
Glass	214	6	9	192	22	96 - 576

strap samples of the same size as that of the training data given. However as voiced by Friedman et al. [34] there is no obvious reason to take this value and there is no evidence that this is an optimal choice in all or any instance. It has been found that increasing the sample size (so it is greater than the training set size) for Bagging leads to a decrease in performance [103]. This is due to the classifiers being built on increasingly similar sample sets and therefore becoming less diverse. In this part of the chapter we consider the effect of using various values for the sample size and the number of neurons in the hidden layer.

5.1.1 Experimental Set-up to investigate modifying the Training Set Size and the Neural Network size

For our experiments we used the Phoneme database taken from the ELENA database¹, the Pima Indian diabetes database and the Glass Identification database, the latter two taken from the UCI Repository of Machine Learning Database.² Details of these sets of data are shown in Table 5.1.

For each data set we ran various versions of AdaBoost and monitored the effects on the accuracy. We call bootstrap size N with 15 hidden neurons our “standard set-up”, because this is the architecture we have been using for all our experiments with AdaBoost and there seems to be no standard in the literature. Each experiment was carried out using ten-fold cross validation so the results shown are averaged over ten runs. For the larger Phoneme database we used reverse-cross validation, i.e., we used one tenth of the data for training and kept the remaining nine tenths for testing. With the smaller Pima and Glass databases we used standard cross-validation training on nine tenths and testing on the remaining one tenth of the data. The fifth and sixth columns of Table 5.1 show the sizes of the training and testing sets supplied to AdaBoost. The seventh column shows the range of values that the sample size could take when random sample sizes were used.

Multilayer Perceptron with a single hidden layer and 100 training epochs was used for all the experiments, using the routine TRAINLM from the neural network toolbox of Matlab 6.5 release 13. For the Glass dataset we found that the computation time was

¹Available via anonymous ftp at [ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases](ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases).

²Available at <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

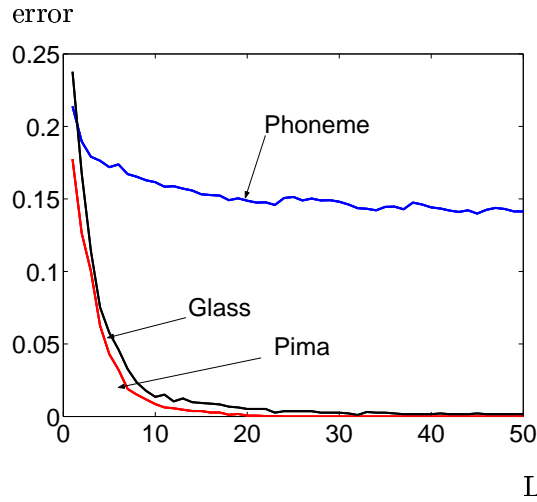


Figure 5.1: THE AVERAGE TRAINING ERROR FOR PHONEME, PIMA AND GLASS WITH 15 NEURONS AND SAMPLE SIZE N

excessive. Consequently, we only have results for the Glass data with bootstrap samples of size $2N$ and less, and number of hidden neurons of size 20 and less. We call this the restricted case when we compare with Phoneme and Pima (Res. Ph. and Res. Pim. respectively).

5.1.2 Training Errors

Figure 5.1 shows the average training error for Pima, Glass and Phoneme data. Glass and Pima show the shape we would expect from AdaBoost, with the training error rapidly going to zero. However, Phoneme shows a less typical shape. It gently decreases but does not go below 0.15. This is the same sort of shape as we found in the previous chapter in Figures 4.3, 4.4 and 4.5 when we were using different parameters with the neural networks. This shape may be due to the fact that Phoneme is a much larger data set and we use a smaller proportion of the data for training, leading to less accurate classifiers. We also see different results for Pima in these experiments than in the experiments in Chapter 4. We have found out that this is because the Matlab version we were using to provide our neural networks was upgraded between the two sets of experiments. Also we modified the parameters from a maximum of 300 training epochs in the experiments in Chapter 4 to a maximum of 100 training epochs in the experiments in this chapter. This seems to have solved the training error problems we had in Chapter 4 as the training error now rapidly reaches zero. Unfortunately, the testing errors are not as low so it seems that the new neural networks are overfitting.

5.1.3 Varying the Sample Size

In this subsection we consider the impact on AdaBoost's performance of modifying the size of the bootstrap sample taken to train the classifier at each iteration. Firstly, we consider using a fixed value for the size of the set AdaBoost samples from the training set based on the original training set size N : $\frac{1}{2}N$, N , $2N$, $3N$, $4N$, $5N$. Afterwards we consider randomly selecting the size of the set AdaBoost samples from the training set from within $\{\frac{1}{2}N, \frac{1}{2}N + 1, \dots, 3N - 1, 3N\}$.

Figure 5.2 shows results when modifying the sample size used to build the classifier at each iteration. In all the graphs the thicker, black line indicates the results using the standard bootstrap sample of same size as the original training set, denoted N , with our standard value of 15 neurons in the hidden layer. The x-axis in each plot shows the number of classifiers. The y-axis in each plot is the testing error obtained through 10-fold cross-validation. The lines indicate using various multiples of the training size N value, $\frac{1}{2}N$ to $5N$. The graphs for Phoneme and Pima are split into two parts for readability.

Examining all of the graphs we find that using $\frac{1}{2}N$ or $2N$ have about the same if not slightly higher error rates than using N for Phoneme and Pima, but show slightly lower error rates for Glass after 20 classifiers have been added to the ensemble. Using $3N$ gives a lower testing error rate than N , with Pima and a similar testing error rate to N with Phoneme.

If we used between 15 and 20 classifiers with the Pima dataset we would have considerably better performance with $3N$ or $4N$ rather than N or $5N$ bootstrap sample size. But for less than 10 classifiers $5N$ is the best choice. Similarly for Glass data, N is the best choice for less than 20 classifiers but $\frac{1}{2}N$ is the best choice for more than 20 classifiers. This suggests that there is no clear reason why one sample size is better than another since the pattern does not show an increase or decrease in performance as the sample size increases. We believe therefore that the optimal sample size is data dependent to some extent. In order to optimise AdaBoost it may be a good idea to conduct some preliminary experiments to find the best sample size to use as well as balancing computation time against accuracy gained.

If we recall figure 4.5 for Pima we note that this shows a different line for 15 neurons and N sample size than we see in figure 5.2. This is partly because we are averaging over 10 runs so we will always get a slightly different result if we do not use exactly the same initial conditions since neural networks are unstable. However, the main reason that there is a difference is that there was an upgrade to Matlab between the two sets of experiments and we are therefore using different neural network programs thus producing different results.

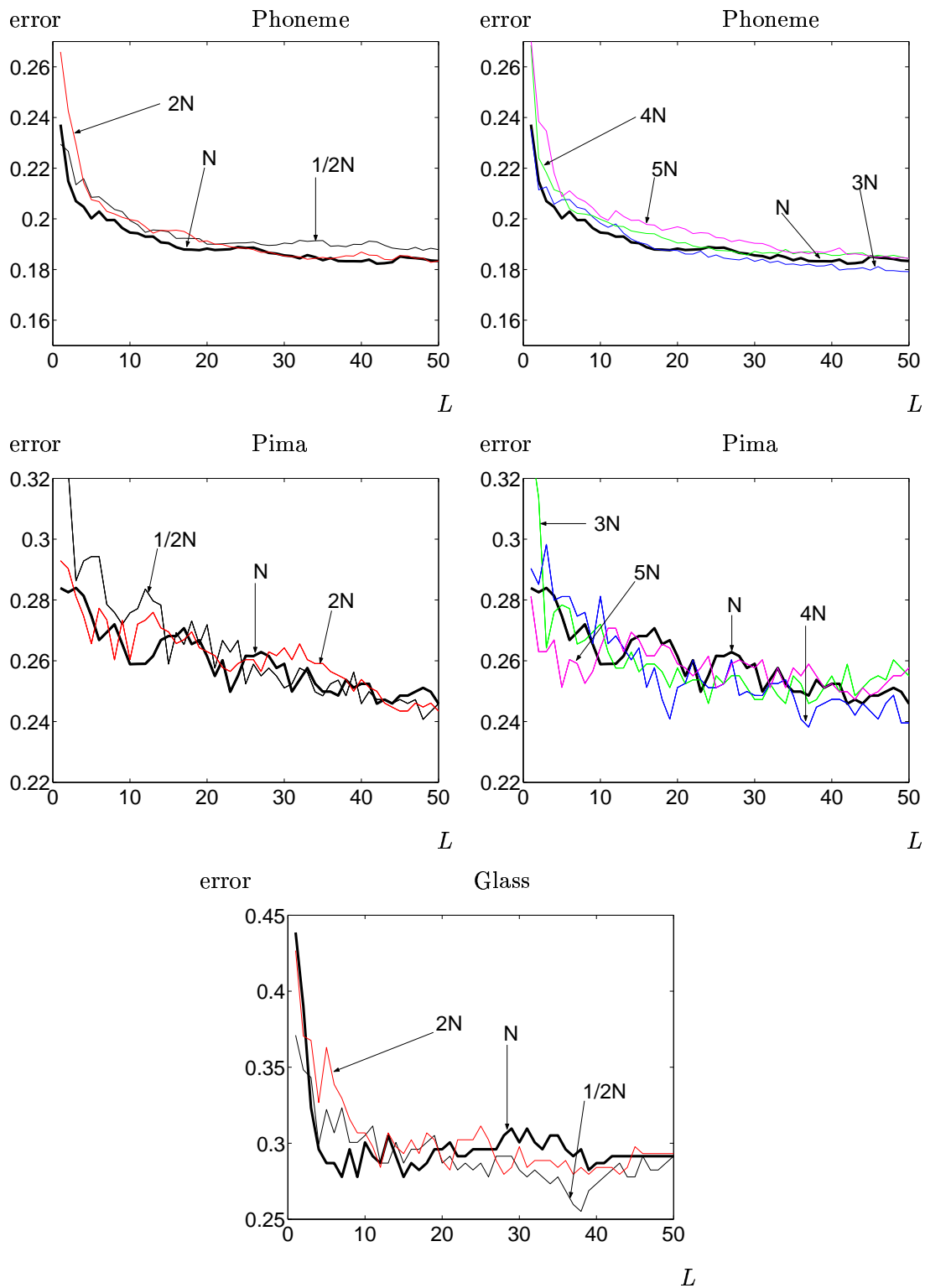


Figure 5.2: THE EFFECT ON THE TESTING ERROR OF VARYING THE SAMPLE SIZE

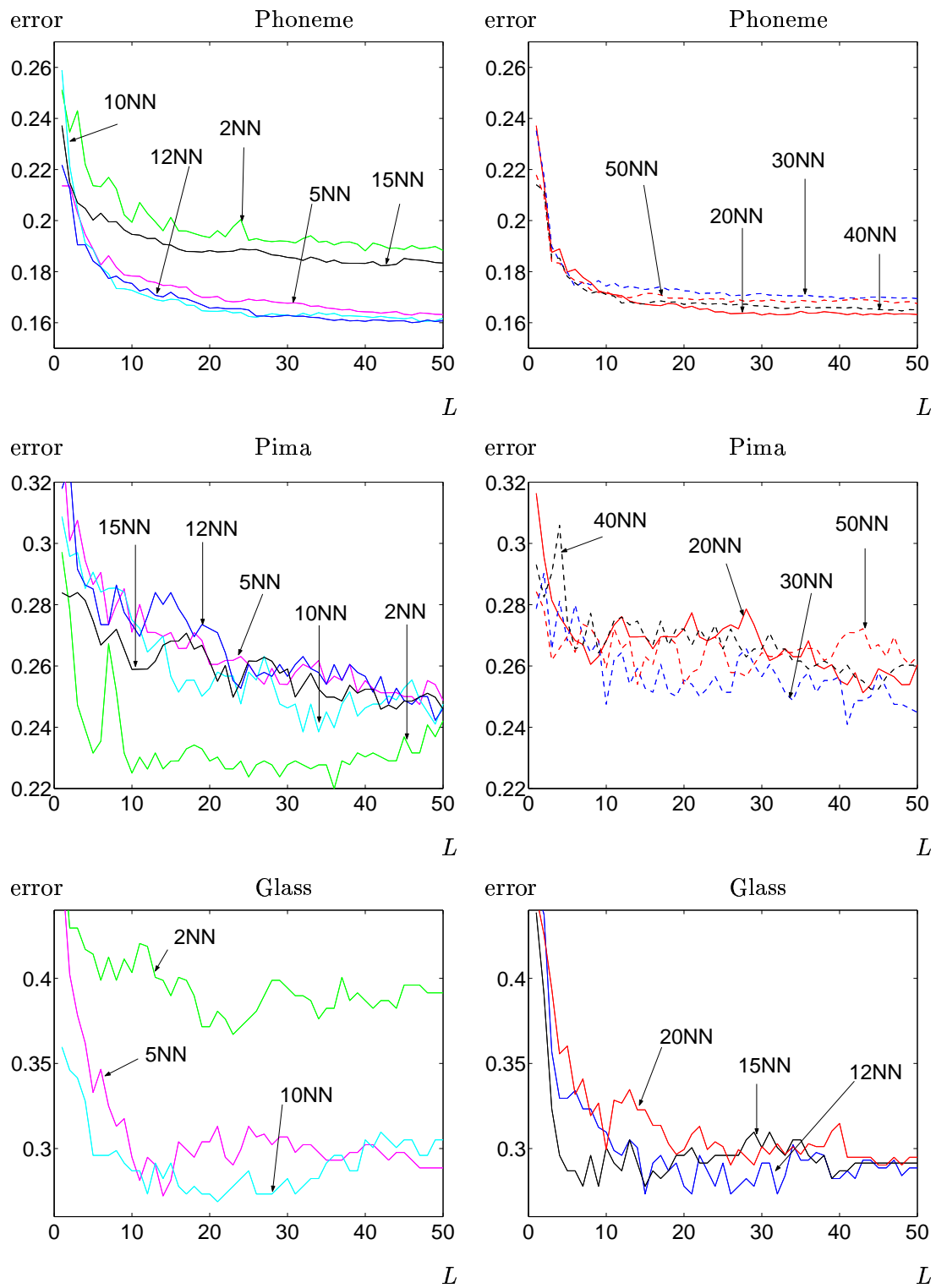


Figure 5.3: THE EFFECT ON THE TESTING ERROR OF VARYING THE NUMBER OF NEURONS IN THE HIDDEN LAYER

5.1.4 Modifying the Number of Neurons Used

In this subsection we consider the impact on AdaBoost's performance of modifying the number of neurons in the hidden layer of the neural network classifiers being built by AdaBoost. Firstly, we consider using a fixed value for the number of neurons: 2, 5, 10, 12, 15, 20, 30, 40, 50 neurons. Then we consider the effect of randomly selecting the number of neurons in the hidden layer from within $\{2, 3, \dots, 24, 25\}$ neurons.

Figure 5.3 shows results when modifying the number of neurons in the neural network's hidden layer. The lines indicate different choices of the number of neurons ranging from 2 to 50. The standard sample size N was used in all these experiments. The results for each data set are again shown in two subplots for readability.

For the Phoneme data 2 neurons is a poor choice and 10 and 12 neurons gives the best results but there is not very much difference between the performance with 5,10,12,15,20 and 40 neurons. For the Pima data using 2 neurons is much better than any of the others with 10 and 15 neurons the next best numbers of neurons to use. For the Glass dataset 2 neurons is the worst and 5,10,12,15,20 neurons are fairly similar in performance with 10 and 12 neurons being marginally better than the others.

Examining all the graphs as a whole shows that the best value is data dependent and there is no clear overall, choice. It also shows that a variation in the number of neurons does affect the performance of the ensemble. This is contrary to Drucker's suggestion in [22] that any deficiencies in the performance of the networks due to the architecture will be made up for by the boosting algorithm. Our results suggest that it would be sensible to conduct a preliminary experiment to identify which number of neurons would produce the lowest testing error.

5.1.5 Varying Both Sample Size and Number of Neurons

Here we randomly select both the size of the bootstrap sample from $\{\frac{1}{2}N, \dots, 3N\}$ and the number of neurons from $\{2, 3, \dots, 24, 25\}$.

Figure 5.4 shows results when randomly modifying the sample size, the number of neurons in the neural network's hidden layer or both. The graphs represent randomly changing the sample size, randomly changing the number of neurons in the hidden layer or randomly changing both, throughout the ensemble's growth.

For the Phoneme dataset using N and 15 neurons is the worst choice, with randomly changing the sample size, number of neurons or both all showing a similar error rate, lower than our standard set-up. The results for the Pima and Glass datasets are more complicated. It seems that the best structure to use with these data sets depends on the number of classifiers you want in the ensemble. For less than 10 classifiers using randomly changing sample size and number of neurons is best with the Pima data set but

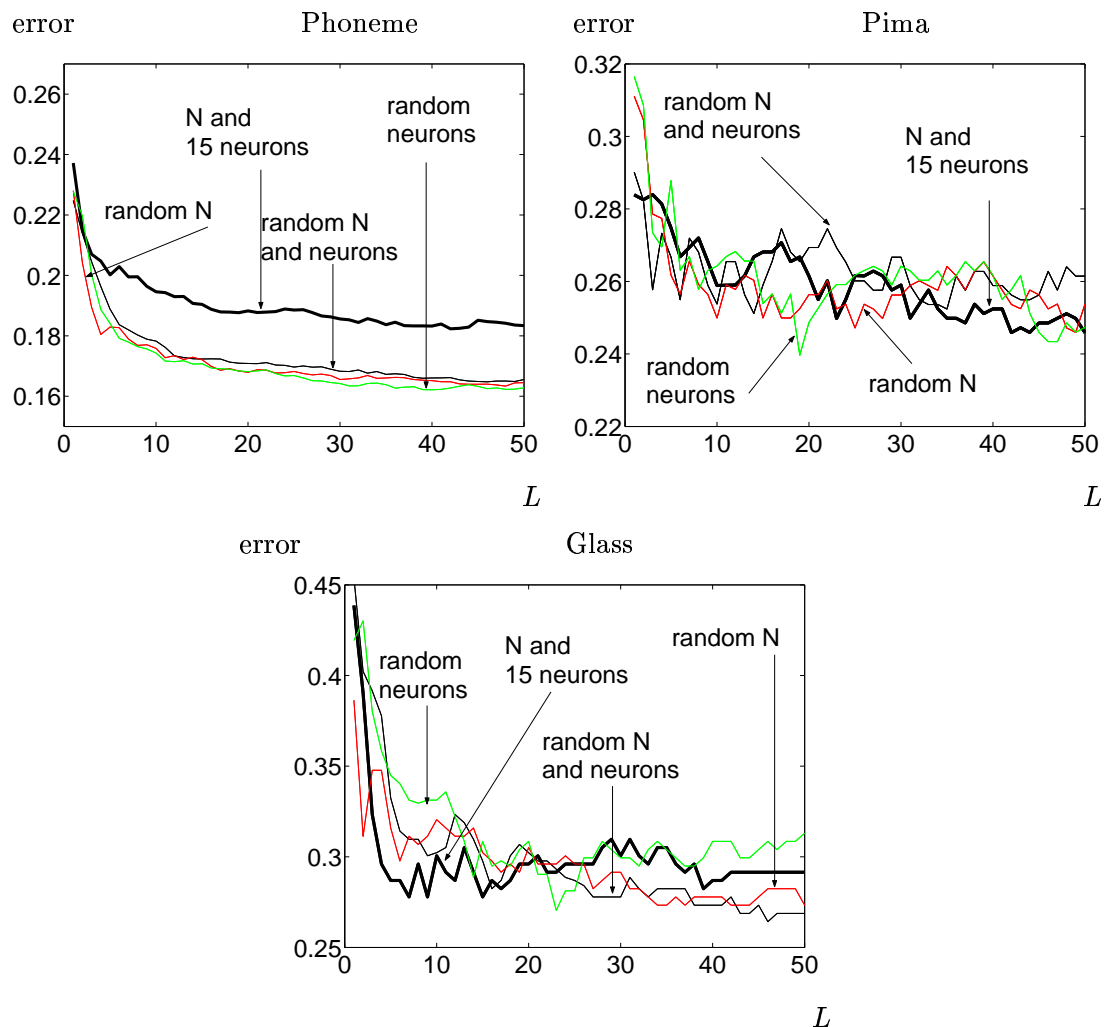


Figure 5.4: THE EFFECT ON THE TESTING ERROR OF RANDOMLY VARYING THE SAMPLE SIZE, OR THE NUMBER OF NEURONS IN THE HIDDEN LAYER, OR BOTH

for between 15 and 30 classifiers it is considerably worse than all the other methods. For the Glass results after 25 classifiers have been added, randomly changing the sample size and randomly changing both the sample size and the number of neurons are the better options with randomly changing both having the slightly lower testing error. Using our standard of sample size N and 15 neurons is the best choice up to about 20 classifiers after which its performance deteriorates considerably and is the worst of them all between 25 and 35 classifiers.

Examining all three graphs shows that there is again no ‘right’ choice and so considerations such as time must also come into play. Since the more parameters we modify, the more time is taken, random modification of N would seem to be a good compromise between performance and computation time.

Table 5.2: BEM PROCEDURE COMPARING THE ADABOOST IMPLEMENTATIONS

Classifier D_i	Test Error						Wins/Successes					
	$\frac{1}{2}N$	N	$2N$	$3N$	$4N$	$5N$	$\frac{1}{2}N$	N	$2N$	$3N$	$4N$	$5N$
1	0.229	0.237	0.266	0.235	0.269	0.271	1	0	0	0	0	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
50	0.188	0.183	0.183	0.179	0.184	0.184	0	0	0	1	0	0
Successes ($Y_{i,j}$) =							1	16	0	33	0	0

We now proceed to statistically compare the various different values for the bootstrap sample size and the number of hidden neurons. We do this by adapting the BEM³ procedure for comparing competing pattern recognition algorithms used by Alsing et al in [2]. By comparing the testing errors (to find the minima) given by the classifiers for different implementations of AdaBoost we can select the best version for each iteration. Table 5.2 shows the procedure for calculating the number of successes/wins for each bootstrap sample size with the Phoneme data. For each classifier the method with the lowest error wins and gets 1, in the event of a tie between k methods each method gets $\frac{1}{k}$. The total value of successes for each method is then calculated. The conditional probability of procedure $proc_i$ being the best for data set Ψ_j is calculated as:

$$P(proc_i|\Psi_j) = \frac{Y_{i/j}}{v_j} \quad (5.1)$$

where $Y_{i/j}$ is the number of successes for procedure i on dataset j , and v_j is the size of dataset j . The total probability of procedure $proc_i$ being the best for all the data sets is then calculated as:

$$P(proc_i) = \sum_j P(proc_i|\Psi_j)P(\Psi_j) \quad (5.2)$$

where $P(\Psi_j)$ is the probability of dataset j (for comparing Phoneme and Pima, $P(Ph) = P(Pim) = 0.5$, and for all three $P(Ph) = P(Pim) = P(Glass) = \frac{1}{3}$).

Tables 5.3 and 5.4 show the number of successes for each bootstrap sample size and each number of hidden neurons. The bold numbers refer to the maxima for each dataset and each modification (Bootstrap size or number of neurons).

First we consider the overall probabilities calculated by Equation (5.2) for the Phoneme and Pima datasets together, then we consider all three datasets (recall that we exclude bootstrap sample sizes $> 2N$ and hidden neurons > 20 as we do not have this information for the Glass dataset). These results are shown as the final two rows of Tables 5.3 and 5.4. The results show that the best performance for the Phoneme data is with sample size $3N$ and 12 hidden neurons. For the restricted version (sample size $< 3N$, < 30 neurons)

³Bechofer, Elmaghraby, and Morse

N is the best sample size and again 12 hidden neurons. Similarly with Pima, $4N$ works best but N is best if we restrict it to less than $3N$. With the number of hidden neurons we can see that the results for Pima are biased very much in favour of 2 neurons. With the Glass dataset, $\frac{1}{2}N$ and 10 hidden neurons are the best. If we consider the results with both Phoneme and Pima we find that $3N$ and 2 neurons are the winners. For all three datasets with the restricted range of values we find that N is the preferred sample size and 2 hidden neurons is best. The fact that the number of hidden neurons that wins is 2 is hardly surprising considering the results for Pima. The fact that 2 neurons did so

Table 5.3: SUCCESSES, Y_i/j , FOR DIFFERENT BOOTSTRAP SAMPLE SIZES

Data	Bootstrap Sample Size					
	$\frac{1}{2}N$	N	$2N$	$3N$	$4N$	$5N$
Phon.	1	16	0	33	0	0
Ph.Res.	1	36	13	X	X	X
Pima	4	4	1	11	19	11
Pim.Res.	16	20	14	X	X	X
Glass	28	16	6	X	X	X

Total Probabilities of being the best:

Ph&Pim	0.05	0.20	0.01	0.44	0.19	0.11
All	0.30	0.48	0.22	X	X	X

Table 5.4: SUCCESSES, Y_i/j , FOR DIFFERENT NUMBERS OF HIDDEN NEURONS

Data	Hidden Neurons								
	2	5	10	12	15	20	30	40	50
Phon.	0	1	11	22	0	3	1	8	4
Ph.Res.	0	1	12	23	2	12	X	X	X
Pima	48	0	0	0	0	0	1	1	0
Pim.Res.	49	0	0	0	0	1	X	X	X
Glass	0	5	27	17	1	0	X	X	X

Total Probabilities of being the best:

Ph&Pim	0.48	0.01	0.11	0.22	0	0.03	0.02	0.09	0.04
All	0.326	0.04	0.26	0.26	0.02	0.086	X	X	X

well suggests that AdaBoost may be adversely affected by outliers within the Pima data. This has been suggested previously as a reason for the poorer performance of AdaBoost on this data [9]. Our results suggest that it would be sensible to conduct a preliminary experiment to identify which number of neurons would produce the lowest testing error.

5.2 Kappa-error diagrams and Pareto-optimal sets

Recall the studies using overproduce and choose strategies with AdaBoost building the initial ensemble of base classifiers, which we discussed in the previous chapter (Chapter 4.6.6). This section looks in more depth at one ‘choose’ method in particular, the method introduced by Margineantu and Dietterich: Kappa-Error Convex Hull Pruning [74]. Here we hope to test the hypothesis that using Kappa-Error Convex Hull Pruning produces a considerably smaller ensemble of classifiers without significantly increasing the generalisation error.

5.2.1 Kappa for class label outputs

If we have two classifiers D_a and D_b on our data set of N examples we can develop a contingency matrix, \mathcal{C} . In this table cell \mathcal{C}_{ij} contains the number of examples x for which $D_a(x) = i$ and $D_b(x) = j$. If the two classifiers are identical then only the diagonal will contain non-zero values, and if they are very different the values off the diagonal will be large. If we define Θ_1 to be the probability that the two classifiers agree and Θ_2 the probability that they agree by chance,

$$\Theta_1 = \frac{\sum_{i=1}^c \mathcal{C}_{ii}}{N} \quad (5.3)$$

$$\text{and, } \Theta_2 = \sum_{i=1}^c \left(\sum_{j=1}^c \frac{\mathcal{C}_{ij}}{N} \cdot \sum_{j=1}^c \frac{\mathcal{C}_{ji}}{N} \right) \quad (5.4)$$

then we can use the κ_E -statistic ⁴,

$$\kappa_E = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \quad (5.5)$$

to define a measure of agreement. This, as the 0/1 version κ , is a measure of diversity of (\downarrow)-type, the higher the value the less diverse (see Chapter 3, Sections 3.1 and 3.1.2 for more information). $\kappa_E = 0$ when the two classifiers are independent and $\kappa_E = 1$ when the two classifiers are identical. Negative values can occur when there is negative correlation between the two but this rarely occurs in practice [74].

Figure 5.5 shows an example of how to calculate κ_E . Note that Θ_1 is calculated as the sum of the diagonal divided by the number of examples. Θ_2 is calculated by multiplying

⁴Recall the measure of interrater agreement, denoted κ , introduced in section 3.1.2. It is a special case of κ_E when applied to 0/1 outputs.

the sum of the first row, divided by the total number of examples by the sum of the first column, divided by the total number of examples, repeating this for the second row and column and so on and finally totalling them all up to give the summation. In the example we have $c = 3$ classes, and $N = 10$ examples and four equally accurate but different classifiers. Note, to illustrate the range of κ_E we have deliberately designed classifiers D_1 and D_2 to have identical outputs on this data ($D_1 = D_2$) and D_3 to have different outputs. D_4 is such that it is independent of D_3 , they give the same classification on half of the sample objects and different classifications on the other half. Here we see that κ_E for D_1 and D_2 is 1 since they are identical, is -0.18 for D_1 and D_3 suggesting they are quite diverse, 0.26 for D_3 and D_4 so they are slightly similar and is 0.39 for D_1 and D_4 so they are even more similar.

5.2.2 Kappa-error Diagrams

Using κ_E we are able to follow the ideas of Margineantu and Dietterich to draw a Kappa-Error diagram which allows us to visualise an ensemble of classifiers [74]. A Kappa-error diagram is a scatterplot where each point represents a pair of classifiers. The x coordinate is calculated as the κ_E value for the pair of classifiers and the y coordinate is the average of their errors.

Margineantu and Dietterich found that the κ_E -error diagram for AdaBoost and that for Bagging showed the different nature of the two ensemble methods. The classifiers produced by Bagging have a much tighter cluster than those produced by AdaBoost. This is not particularly surprising as Bagging trains classifiers on a sample drawn from the same uniform distribution, whereas AdaBoost modifies the distribution samples are drawn from. Thus AdaBoost produces a more diverse set of classifiers. The classifiers with lower accuracy tend to have higher diversity (and of course lower weight in the AdaBoost weighted voting combining method) which compensates for the weaker accuracy. This gives more evidence to how and why AdaBoost outperforms Bagging.

If AdaBoost produces very many classifiers then by careful selection of a subset of them we may be able to improve the ensemble performance and reduce the computation time; an overproduce and choose strategy. Margineantu and Dietterich used their κ_E -error diagrams to give a pruning method in order to reduce the size of an ensemble classifiers produced by AdaBoost. They produce a subset of the AdaBoost ensemble by calculating the convex hull of the of the points in the κ_E -error diagram and using the corresponding classifiers in their subset. Since the preferred pairs of classifiers are found in the bottom left of the plot the convex hull is a reasonable choice. This does however, have one drawback in that there is no control over how many classifiers end up in the final set. A convex set of points is such that if two points are in the set then so are all points on the line segment

joining them.

Definition 2 *The Convex Hull is the intersection of all convex sets containing a subset, A of a real vector space.*

True Class	Z	D_1	D_2	D_3	D_4
ω_1	z_1	ω_1	ω_1	ω_3	ω_3
ω_2	z_2	ω_2	ω_2	ω_2	ω_2
ω_3	z_3	ω_2	ω_2	ω_3	ω_3
ω_1	z_4	ω_1	ω_1	ω_2	ω_1
ω_1	z_5	ω_3	ω_3	ω_1	ω_1
ω_2	z_6	ω_2	ω_2	ω_1	ω_3
ω_2	z_7	ω_1	ω_1	ω_2	ω_1
ω_2	z_8	ω_2	ω_2	ω_2	ω_2
ω_1	z_9	ω_1	ω_1	ω_3	ω_1
ω_3	z_{10}	ω_2	ω_2	ω_3	ω_2
Accuracy		0.6	0.6	0.6	0.6

$$\mathcal{C}(D_1, D_2) = \begin{matrix} & \omega_1 & \omega_2 & \omega_3 \\ D_1 & \begin{matrix} \omega_1 & \omega_2 & \omega_3 \end{matrix} \end{matrix}$$

ω_1	4	0	0
ω_2	0	5	0
ω_3	0	0	1

$$\mathcal{C}(D_1, D_3) = \begin{matrix} & \omega_1 & \omega_2 & \omega_3 \\ D_1 & \begin{matrix} \omega_1 & \omega_2 & \omega_3 \end{matrix} \end{matrix}$$

ω_1	0	2	2
ω_2	1	2	2
ω_3	1	0	0

$$\mathcal{C}(D_3, D_4) = \begin{matrix} & \omega_1 & \omega_2 & \omega_3 \\ D_3 & \begin{matrix} \omega_1 & \omega_2 & \omega_3 \end{matrix} \end{matrix}$$

ω_1	1	0	1
ω_2	2	2	0
ω_3	1	1	2

$$\mathcal{C}(D_1, D_4) = \begin{matrix} & \omega_1 & \omega_2 & \omega_3 \\ D_1 & \begin{matrix} \omega_1 & \omega_2 & \omega_3 \end{matrix} \end{matrix}$$

ω_1	3	0	1
ω_2	0	3	2
ω_3	1	0	0

Note: $D_1 = D_2$ so we only need to calculate D_1 's relationship with D_3 and D_4 .

$$\begin{aligned} (D_1, D_2) : \Theta_1 &= \frac{\mathcal{C}_{11} + \mathcal{C}_{22} + \mathcal{C}_{33}}{10} = \frac{4 + 5 + 1}{10} = \frac{10}{10} = 1, \\ \Theta_2 &= \frac{4}{10} \times \frac{4}{10} + \frac{5}{10} \times \frac{5}{10} + \frac{1}{10} \times \frac{1}{10} = \frac{42}{100}, \kappa_E = \frac{1 - 0.42}{1 - 0.42} = 1 \\ (D_1, D_3) : \Theta_1 &= \frac{2}{10} = 0.2, \Theta_2 = \left(\frac{\mathcal{C}_{11} + \mathcal{C}_{12} + \mathcal{C}_{13}}{10} \times \frac{\mathcal{C}_{11} + \mathcal{C}_{21} + \mathcal{C}_{31}}{10} \right) + \\ &\quad \left(\frac{\mathcal{C}_{21} + \mathcal{C}_{22} + \mathcal{C}_{23}}{10} \times \frac{\mathcal{C}_{12} + \mathcal{C}_{22} + \mathcal{C}_{32}}{10} \right) + \left(\frac{\mathcal{C}_{31} + \mathcal{C}_{32} + \mathcal{C}_{33}}{10} \times \frac{\mathcal{C}_{13} + \mathcal{C}_{23} + \mathcal{C}_{33}}{10} \right) \\ &= \left(\frac{0 + 2 + 2}{10} \times \frac{0 + 1 + 1}{10} \right) + \left(\frac{1 + 2 + 2}{10} \times \frac{2 + 2 + 0}{10} \right) + \left(\frac{1 + 0 + 0}{10} \times \frac{2 + 2 + 0}{10} \right) \\ &= \left(\frac{4}{10} \times \frac{2}{10} \right) + \left(\frac{5}{10} \times \frac{4}{10} \right) + \left(\frac{1}{10} \times \frac{4}{10} \right) = \frac{32}{100}, \kappa_E = \frac{0.2 - 0.32}{1 - 0.32} = -0.18 \\ (D_3, D_4) : \Theta_1 &= \frac{5}{10} = 0.5, \Theta_2 = \frac{2}{10} \times \frac{4}{10} + \frac{4}{10} \times \frac{3}{10} + \frac{4}{10} \times \frac{3}{10} = \frac{32}{100}, \kappa_E = \frac{0.5 - 0.32}{1 - 0.32} = 0.26 \\ (D_1, D_4) : \Theta_1 &= \frac{6}{10} = 0.6, \Theta_2 = \frac{4}{10} \times \frac{4}{10} + \frac{5}{10} \times \frac{3}{10} + \frac{1}{10} \times \frac{3}{10} = \frac{34}{100}, \kappa_E = \frac{0.6 - 0.34}{1 - 0.34} = 0.39 \end{aligned}$$

Figure 5.5: EXAMPLE OF CALCULATING KAPPA_E

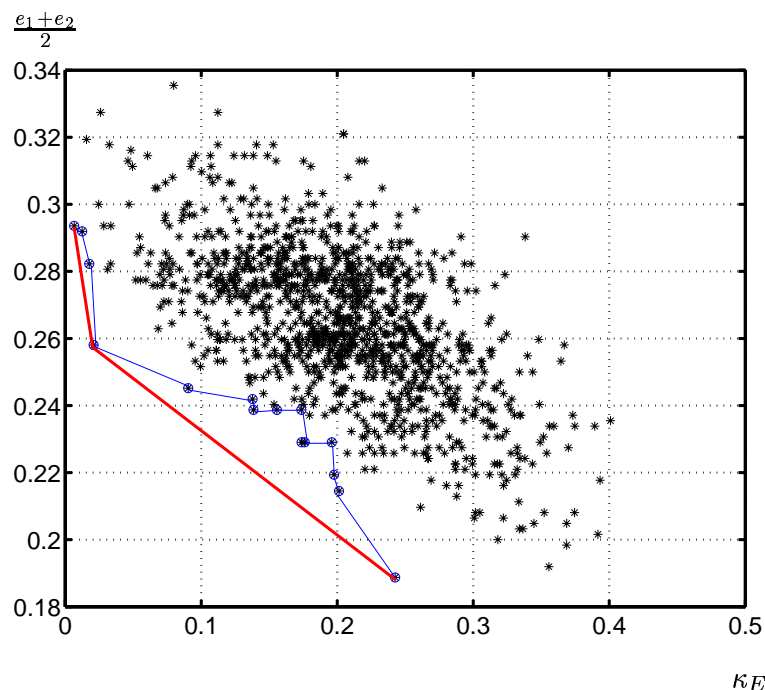


Figure 5.6: AN EXAMPLE OF A KAPPA-ERROR PLOT, SHOWING THE CONVEX HULL (THICK, RED LINE) AND THE PARETO OPTIMAL SET (THIN, BLUE LINE)

Figure 5.6 shows an example of a Kappa-error diagram. It consists of a single run of AdaBoost on the Liver data⁵, building 50 decision tree classifiers. The convex hull and the Pareto-optimal set (see next subsection) are also shown.

5.2.3 Pareto-Optimal Sets

In practice the convex hull often discards a large proportion of the classifiers originally built by the AdaBoost algorithm. Also it can be overly sensitive to noise since even small variations in κ_E and the average error can change the whole shape of the convex hull and therefore alter the chosen set of classifiers. By calculating the Pareto-optimal set we can achieve a better balance between ensemble accuracy and size of the final set.

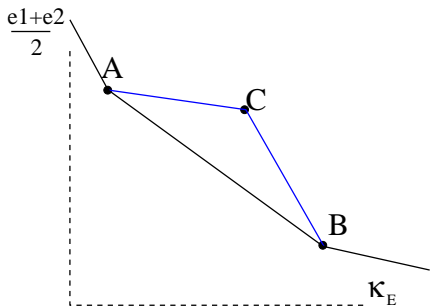
Definition 3 *The Pareto-optimal set $S^* \subseteq S$ contains all non-dominated alternatives.*

Let $A = \{a_1, \dots, a_m\}$ be a set of alternatives (a pair of classifiers for our case), These alternatives are characterised by a set of criteria $C = \{C_1, \dots, C_M\}$ (In our case the criteria are κ_E and error).

An alternative a_i is called non-dominated if there is no other alternative $a_j \in S$, $j \neq i$, so that a_j is better than a_i on *all* criteria. Therefore, for our case, the Pareto optimal set

⁵from UCI Machine Learning repository

will be a superset of the convex hull. Figure 5.7 shows the difference between the convex hull and the Pareto optimal set.



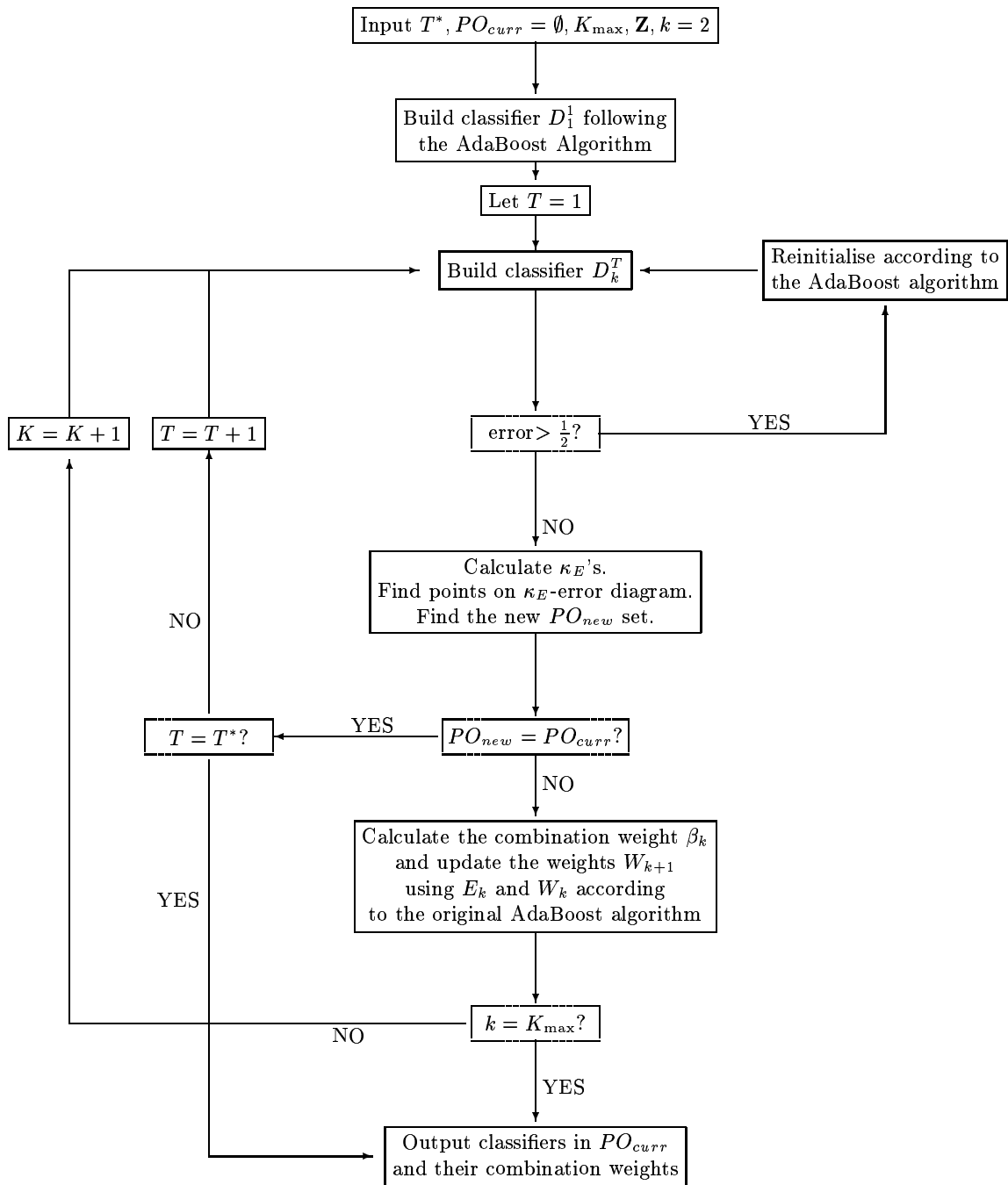
If points A and B are in the convex hull then point C is not since it is behind segment AB. However, we can see that C has lower error than A and is more diverse than B. Thus C is not worse than A on all of the criteria, nor is it worse than B on all of the criteria. Therefore, C is “non-dominated” and so would be included in the Pareto-optimal set.

Figure 5.7: PARETO OPTIMALITY [56]

5.3 AdaBoost with Pareto Optimality

Figure 5.8 shows how we have modified the AdaBoost algorithm to take account of Pareto optimality. The input is the basic training set, Z , the maximum number of classifiers built, K_{\max} , the Pareto optimal set (initially empty), PO_{curr} , and the maximum number of iteration permitted with no change in the Pareto optimal set, T^* . The first classifier is built and then at each subsequent iteration the classifiers are built according to the original AdaBoost algorithm. The classifier’s error is then calculated. If the error is greater than $\frac{1}{2}$ we reinitialise according to the original AdaBoost algorithm. If the error is less than $\frac{1}{2}$, the pairwise values of κ_E between the new classifier and all the existing classifiers are calculated as the points on a kappa-error diagram. The Pareto optimal set is then calculated. If there is no difference between this new Pareto set and the current Pareto set then we discard this classifier and build another one. This cycle, with no change in the Pareto optimal set, can be repeated up to T^* times before we exit the loop and output the classifiers in the current Pareto optimal set as our choice of ensemble. If however, the new Pareto set is different from the current Pareto set then we update the current set and update the weights according to the original AdaBoost algorithm. This can continue until a maximum number of K_{max} classifiers has been built.

We can see that if the classifiers do not alter the Pareto set very much then we can produce quite a small ensemble. This can reduce the computation time, hopefully without dramatically altering the amount of information contained within the ensemble.



Key:

k the current iteration;

S_k the training set;

$W_k(i)$ the weight for object i ;

β_k the combination weights;

PO_{curr} the current Pareto optimal set;

K_{max} the number of iterations;

D_k the classifier trained;

$W_k = \{W_k(1), \dots, W_k(N)\}$ the set of weights used;

T^* max no. iterations with no change in the Pareto optimal set;

PO_{new} the new Pareto optimal set;

Figure 5.8: THE ADABOOST ALGORITHM WITH PARETO-OPTIMAL SETS

5.3.1 Experimental set-up to investigate AdaBoost with Pareto Optimality

We are interested in how using the Pareto set compares with using the full ensemble produced by AdaBoost on both accuracy and the number of iterations. For our experiments we use ten different data sets from UCI machine learning repository. These are Ecoli, German, Glass, Ionosphere, Liver, Pima, Sonar, Vehicle, Votes and Wisconsin Breast Cancer. Table 5.5 shows a summary of the datasets. The data sets are described in more detail in Appendix B.

Table 5.5: SUMMARY OF THE DATA SETS

Name	No. Classes	Size of data set	No. Features	appendix
Ecoli	8	336	7	B.4
German	2	1000	23	B.11
Glass	7	214	9	B.2
Ionosphere	2	351	34	B.6
Liver	2	345	6	B.5
Pima	2	768	8	B.9
Sonar	2	208	60	B.1
Vehicle	4	846	18	B.10
Votes	2	435	16	B.7
wbc	2	569	30	B.8

In these experiments we used decision trees. We chose K_{\max} (the maximum number of classifiers built) to be 50. One set of experiments was carried out using T^* (the number of iterations with unchanged Pareto sets before exiting the algorithm) to be 5, another set was carried out using $T^* = 10$ and a control set was carried out using the full 50 classifiers built using the original AdaBoost algorithm. Each run of experiments was repeated 100 times and averaged to give a general view of the performances for comparison. The Pareto version with $T^* = 5$ is referred to as $P5$ and with $T^* = 10$ as $P10$.

5.3.2 AdaBoost with Pareto Optimality results

Table 5.6 shows the error averaged over 100 runs for basic AdaBoost, AdaBoost with Pareto for 5 repeats and AdaBoost with Pareto for 10 repeats. As we can see, AdaBoost is more accurate on five of the ten datasets with P5 the best on four and P10 the best on one dataset. There are no significant differences between the errors. However if we consider the size of the ensembles we could dramatically reduce the size of ensemble needed

to achieve these levels of error and so reduce the computation time involved.

Table 5.6: THE AVERAGE ERROR FOR ADABOOST, PARETO WITH 5 REPEATS AND PARETO WITH 10 REPEATS. THE NUMBERS IN BOLD SHOW THE LOWEST ERROR.

Data	AdaBoost	P5 mean error	P10 mean error
Ecoli	0.1418	0.1515	0.1515
German	0.2798	0.2836	0.2881
Glass	0.2350	0.2168	0.2314
Ionosphere	0.0708	0.0642	0.0750
Liver	0.2889	0.2997	0.3057
Pima	0.2652	0.2756	0.2710
Sonar	0.1748	0.1862	0.1776
Vehicle	0.2539	0.2578	0.2498
Votes	0.0495	0.0470	0.0523
wbc	0.0381	0.0354	0.0374

The top part of figure 5.9 illustrates the percentage error change of the Pareto set runs over the AdaBoost runs. As we can see the error has increased by up to 7% and reduced by nearly 10% depending on which dataset has been used. Note that for Ecoli, the percentage change in error for both P5 and P10 is identical and so are both represented by the same dot in the figure.

The lower part of figure 5.9 shows the average range and mean of the number of runs taken by the two Pareto runs. Naturally Pareto with $T^* = 5$ generally uses less classifiers in the ensemble than Pareto with $T^* = 10$ since it exits the algorithm after only 5 iterations with no change in the Pareto set rather than the 10 iterations for P10.

If we look at the number of runs we see that for German, Liver and Pima with P5 and P10 and for Votes with P5 the average number of classifiers in the ensemble is less than 40. There is a small increase in error corresponding to these of 2 to 4% for P5 and 3 to 6% for P10 with the exception of Votes which as well as a lower average in the number of runs reduces the error by about 5%. For the other datasets where P5 was the best algorithm, Glass and Ionosphere, there tends to be a higher but still quite wide range in the average number of runs. For wbc there is very little variation in the number of runs, using between 46 and 50 classifiers in the ensemble, but there was still a quite dramatic reduction in the error for P5 and a slight reduction for P10.

The results suggest that again the choice of algorithm is data-dependent but that using the AdaBoost with P5 is preferable to AdaBoost with P10. The choice of whether to use basic AdaBoost or AdaBoost with P5 will depend on several factors. The data set used

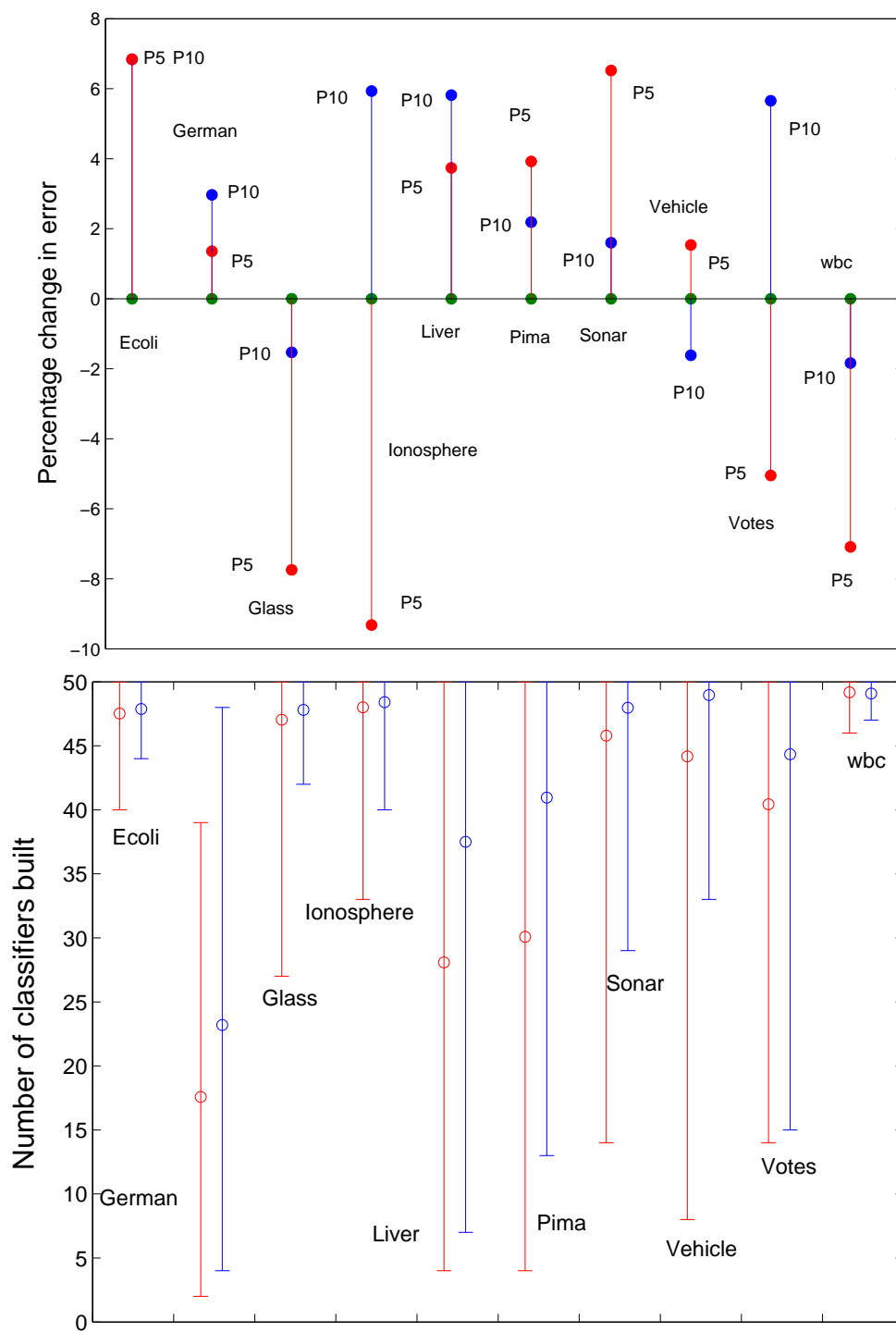


Figure 5.9: COMPARING THE PERCENTAGE CHANGE IN ERROR FROM THE STANDARD ADABOOST AND THE AVERAGE NUMBER OF RUNS WHEN USING PARETO WITH 5 REPEATS (RED DOTS/LINES ON LEFT) AND PARETO WITH 10 REPEATS (BLUE DOTS/LINES ON RIGHT)

will affect the choice. If there is time, a preliminary comparison between the two on the specific data may be beneficial. The trade-off between accuracy and expedience is another consideration, if the computation time is the most important factor then P5 would be the preferable choice.

5.4 Improving AdaBoost Conclusions

In this chapter we considered several possible ways of improving AdaBoost. Firstly we conducted an investigation into how modifying the parameters of AdaBoost using resampling and neural networks affected the classifier ensemble's performance. We carried out experiments using three datasets, Phoneme (5 attributes, 2 classes and 5404 instances), Pima Indian diabetes (8 attributes, 2 classes and 768 instances) and Glass (9 attributes, 7 classes and 214 instances). Our results suggest that we may be able to improve the performance of AdaBoost by using 10 or 12 neurons in the hidden layer of our neural network rather than using 15 as we currently do. This would also have the added advantage of taking less computation time. The size of bootstrap sample does not make much difference to the error rate. Keeping the bootstrap sample the same size as the training data should not make much difference with most data sets. The results also show that the best value to use for bootstrap size or number of hidden neurons can be very data specific. Some preliminary experiments to find the best value to use may be beneficial with some datasets such as we found with the Pima data. Also it may be a good idea to use a random modification of the sample size in some instances to avoid performance degradation as the ensemble grows as we found with the Glass data (see Figure 5.4).

The second part of our investigation involved using Pareto optimality to add classifiers built by AdaBoost to the ensemble only if they were in the Pareto optimal set. We carried out experiments with ten datasets from the UCI machine learning repository. We compared standard AdaBoost to AdaBoost with Pareto 5, and to AdaBoost with Pareto 10. Pareto 5 and 10 allow 5 or 10 cycles, respectively, without change in the Pareto set, before exiting the algorithm. Our results showed that there is no significant difference between the error rates for AdaBoost, Pareto 5 and Pareto 10. However the number of classifiers in the ensemble can be reduced. In particular the Pareto 5 approach can reduce the number of classifiers in the ensemble without necessarily increasing the error. Pareto 5 may therefore be better than standard AdaBoost but it depends upon the data set.

Overall, we have found that there are no hard and fast rules when it comes to algorithms since they tend to behave differently with different datasets. Thus there is really no option but to carry out preliminary experiments to guide further choices.

Chapter 6

Conclusions

6.1 Main Investigations and Findings of this Thesis

In our first investigation we studied the relationships between nine combination methods. Two data sets were used. We considered the overall accuracies of the combination methods, their improvement over the single best classifier, and the correlation between the ensemble outputs using the different combination methods. We found some interesting relationships and correlations amongst the combination methods. In particular, maximum is equivalent to minimum for the two-class case, average has a close relationship with product, behaviour-knowledge space is correlated with Wernecke's method and majority vote is correlated with naive Bayes.

Next we introduced ten diversity measures. Using the same two data sets, we studied the relationships between the diversity measures. Then we looked at their relationship to the combination methods previously studied. The ranges of the ten diversity measures for three classifiers were derived. They were compared with the theoretical ranges and their implications for the accuracy of the ensemble were studied. We found that for (0/1) classifier outputs with an ensemble of three classifiers the disagreement measure, Kohavi-Wolpert variance and entropy measure are identical up to a coefficient. We also found that the correlation coefficient, measure of interrater agreement, generalised diversity and coincident failure diversity are fairly consistently correlated whilst the double-fault measure was not strongly correlated with any of the other measures.

Considering the relationship between the diversity measures and the combination methods we found that there was very little consistent correlation between the two groups. The largest observed correlations were as shown below:

Diversity		Combination
disagreement		maximum
kw variance	→	minimum
entropy		
		majority vote
difficulty	→	average
		decision templates
double-fault	→	majority vote

It may be this unclear relationship between diversity measures and combination methods which makes the explicit use of diversity in multiple classifier systems such a thorny subject. It is often easier to calculate the diversity of an ensemble of classifiers rather than using a validation set to calculate the accuracy. The current consensus is that directly calculating the accuracy for the chosen combination methods is much more accurate than first calculating diversity and trying to predict the accuracy. Thus using the diversity to identify an ensemble that is likely to be accurate is not necessarily a viable approach. The ambiguous relationship between diversity and accuracy discourages optimising the diversity. It is better to try to enforce diversity in the ensemble or to use diversity to select classifiers for an ensemble when following an ‘overproduce and choose strategy’.

We then proceeded to investigate the diversity of classifier ensembles built using the AdaBoost algorithm. We carried out experiments with two datasets using ten-fold cross validation. We built 100 classifiers each time using linear classifiers, quadratic classifiers or neural networks. We studied how diversity varied as the classifier ensemble grew and how the different types of classifier compared. We confirmed that linear classifiers and quadratic classifiers are not particularly suited for use with AdaBoost. We also found that neural networks are better suited for use with AdaBoost, however their ability to reduce testing error may depend on the data. For neural networks, it seems that AdaBoost is increasing the diversity with each new classifier and that this is why the performance is improving. We have found that it may be possible to determine a good time to stop the AdaBoost algorithm by considering the minimum value of Q on the training data.

Next we considered ways of improving AdaBoost’s performance. We conducted an investigation into how modifying the size of the training sets and the complexity of the individual classifiers alters the ensemble’s performance. We carried out experiments using three datasets. We found that the best values to use can be very data specific. Some preliminary experiments to find these best values may be beneficial.

Lastly we considered using Pareto optimality to determine which classifiers built by AdaBoost to add to the ensemble. We carried out experiments with ten datasets. We

compared standard AdaBoost to AdaBoost with two versions of the Pareto optimality method called Pareto 5 and Pareto 10, to see whether we could reduce the ensemble size without harming the ensemble accuracy. We found that: AdaBoost was most accurate on five datasets, Pareto 5 was most accurate on four datasets and Pareto 10 was most accurate on just one dataset. We also found that the Pareto 5 approach can reduce the number of classifiers in the ensemble without necessarily increasing the error. In some cases it can reduce the error as well. Pareto 5 may therefore be better than standard AdaBoost but it depends upon the data set.

Overall, we have found that there are no hard and fast rules when it comes to which combination method, diversity measure or ensemble construction algorithm to use since the best choice tends to be data-dependent. Thus there is really no other option but to carry out preliminary experiments to guide further choices.

6.2 Limitations of the Thesis

The main limitations of this study were the small number of datasets and the limited number of experimental runs. In chapter 2 and 3, in the investigation into the relationships between the combination methods and the diversity measures, we used two datasets with random halves for one dataset and 10-fold cross-validation for the other dataset. In chapter 4, in the investigation into how AdaBoost affects classifier diversity, we again used two datasets with 10-fold cross-validation. In chapter 5, in the investigation into modifying the training set size and the neural network size with AdaBoost, we used three datasets and 10-fold cross validation. Also in chapter 5, in the investigation into using AdaBoost with Pareto optimality, we used ten datasets and used 100 runs to obtain our averages.

If we had had the time and computation capacity it would have been better to use a set-up similar to this last experiment with many datasets and lots of runs to get results we could be more confident in.

6.3 Summary of My Contributions

Chapter 2 - Combination Methods

- Comparing the accuracies of some of the more commonly used classifier combination methods: majority vote, naive Bayes, behaviour-knowledge space, Wernecke's method, maximum, minimum, average, product and decision templates, to each other and to the single best classifier using an ensemble of three classifiers. This gave the result that the best combination method to use will depend on the dataset being used.

- Examining the Pearson's product moment correlation between the outputs from the classifier combination methods and running a clustering program on the combination methods. This gave the relationships: average is closely related to product, max and min are equivalent for the two class case, behavior-knowledge space is correlated to Wernecke's method. majority vote and naive Bayes are closely related and decision templates are not closely related to any of the other combination methods.

Chapter 3 - Diversity Methods

- The derivation of upper and lower limits for the ten diversity measures: Q-statistic, the correlation coefficient, the disagreement measure, the double-fault measure, the Kohavi-Wolpert variance, the measure of interrater agreement, the entropy measure, the measure of difficulty the generalised diversity and the coincident failure diversity, for the three-classifier case.
- Examining the Pearson's product moment correlation between the diversity values obtained from these ten diversity measures and running a clustering program on the diversity measures. This gave the result that double-fault is not strongly correlated with any of the other diversity measures.
- The result that for three classifiers the entropy measure and the Kohavi-Wolpert variance (and for 0/1 outputs the disagreement method) differ only by a coefficient.
- Examining the Pearson's product moment correlation between the outputs from the classifier combination methods and the values from the diversity measures. This gave the result that there is no strong correlation between the combination methods and the diversity measures.

Chapter 4 - Ensemble Construction Methods

- Examining how the AdaBoost ensemble construction method affects the diversity of the ensemble of classifiers it builds and whether this diversity is related to the generalisation error of the ensemble on combination. This showed us that as AdaBoost added neural network classifiers to the ensemble the value of Q decreased indicating that the diversity of the ensemble was indeed increasing.

Chapter 5 - Modifying AdaBoost

- Examining how modifying the sample size of training data, the number of neurons used, or both, affects the generalisation error of AdaBoost. The results show that this is data specific and preliminary experiments to identify the best architecture to use is the preferable approach.

- Investigating whether or not using Pareto optimal sets can produce considerably smaller ensembles of classifiers without significantly increasing the generalisation error when used with AdaBoost. The results showed that it is indeed possible to reduce the size of the ensemble without significantly altering the testing error.

Throughout - All examples were created by myself.

6.4 Possible Future Considerations

It is possible that future work may consider using the minimum Q to determine a termination point for AdaBoost in order to increase generalisation performance.

As we found that a reasonably high accuracy can be obtained after very few iterations of AdaBoost, it may also be of use to try to enhance the performance of a small ensemble. This could be done by applying different combination methods on a reasonably accurate and diverse ensemble.

It may also be interesting to investigate modifying the re-initialisation criterion for AdaBoost from $\text{error} > \frac{1}{2}$ to $\text{error} > \frac{1}{c}$ for c classes. This would be equivalent to only discarding a classifier which is worse than random guessing.

6.5 My References

The following references are papers relevant to this thesis which I have worked on during the course of my studies. I have also carried out a brief citation search to identify roughly how many times they have been cited in other works.

- L.I. Kuncheva, C.J. Whitaker, C.A. Shipp and R.P.W. Duin.
Is independence good for combining classifiers?,
Proc. 15th International Conference on Pattern Recognition, volume 2, pages 169-171, Barcelona, Spain, 2000. [63]
14 citations found.
- L.I. Kuncheva, C.J. Whitaker, C.A. Shipp, R.P.W. Duin.
Limits on the majority vote accuracy in classifier fusion,
Pattern Analysis and Applications, Vol. 6, pages 22-31 2003. [64]
9 citations found.
- C.A. Shipp and L.I. Kuncheva,
An Investigation into how ADABOOST Affects Classifier Diversity,
In International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pages 203-208, Annecy, France, IPMU2002. [98]

- C.A. Shipp and L.I. Kuncheva,
Relationships between combination methods and measures of diversity in combining classifiers,
Information Fusion, Vol. 3, pages 135-148, 2002. [99]
8 citations found.

I have also been involved in two studies into complexity of data during the course of my research. These, whilst interesting, were not relevant to this thesis:

- C.A. Shipp and L.I. Kuncheva,
Four Measures of Data Complexity for Bootstrapping, Splitting and Feature Sampling,
CIMA2001, Bangor, Wales, UK, 2001, pp429-435. [97]
- L.I. Kuncheva, F. Roli, G.L. Marcialis and C.A. Shipp,
Complexity of Data Subsets Generated by the Random Subspace Method: An Experimental Investigation,
In F. Roli and J. Kittler, editors, 2nd International Conference on Multiple Classifier Systems, Lecture Notes in Computer Science, pages 349-358 Cambridge, UK, 2001, MCS2001, Springer-Verlag. [58]

Appendix A

Proof of Equivalence Relationships

A.1 Proof that Max is equivalent to Min for two classes

Proposition 2 *Let $\mathcal{D} = \{D_1, \dots, D_L\}$, $\Omega = \{\omega_1, \omega_2\}$. Let a_1, \dots, a_L be the outputs of the classifiers for class ω_1 , and $1 - a_1, \dots, 1 - a_L$ be the outputs for class ω_2 , $a_i \in [0, 1]$. Then the class label assigned to \mathbf{x} by the MAX and MIN combination rules will be the same.*

Proof

Without loss of generality assume that $a_1 = \min_i a_i$, and $a_L = \max_i a_i$. Then the minimum combination rule will pick a_1 and $1 - a_L$ as the support for ω_1 and ω_2 respectively, and the maximum rule will pick a_L and $1 - a_1$. Consider the three possible relationships between a_1 and $1 - a_L$.

If $a_1 > 1 - a_L$ then $a_L > 1 - a_1$, and we would select class ω_1 with both methods,

If $a_1 < 1 - a_L$ then $a_L < 1 - a_1$, and we would select class ω_2 with both methods.

If $a_1 = 1 - a_L$ then $a_L = 1 - a_1$, and we will pick a class at random with both methods.

■

A.2 Proof that KW, Ent and D are equivalent for 3 classifiers

Proposition 3 *Let $L = 3$ so that $\mathcal{D} = \{D_1, D_2, D_3\}$. Then Ent and kw, calculated from a data set $Z = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$, $\mathbf{z}_j \in \mathbb{R}^n$, are equivalent up to a coefficient, i.e., $kw = \frac{2}{9}Ent$:*

Proof

$$\text{For 3 classifiers: } kw = \frac{1}{9N} \sum_{j=1}^N l(\mathbf{z}_j) (3 - l(\mathbf{z}_j)), \quad Ent = \frac{1}{N} \sum_{j=1}^N \min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\}$$

where $l(\mathbf{z}_j)$ is the number of classifiers that correctly classify object \mathbf{z}_j , therefore we need to show that:

$$\frac{1}{9N} \sum_{j=1}^N l(\mathbf{z}_j) (3 - l(\mathbf{z}_j)) = \frac{2}{9} \left(\frac{1}{N} \sum_{j=1}^N \min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\} \right).$$

Consider the possible values of $l(\mathbf{z}_j)$ with 3 classes, and the respective values for Ent and kw in Table A.1.

Table A.1: POSSIBLE VALUES FOR Ent AND kw FROM THE DIFFERENT VALUES OF $l(\mathbf{z}_j)$

$l(\mathbf{z}_j)$	$(3 - l(\mathbf{z}_j))$	Ent $\min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\}$	kw $l(\mathbf{z}_j)(3 - l(\mathbf{z}_j))$
0	3	0	0
1	2	1	2
2	1	1	2
3	0	0	0

We can see that the sum of entries from column 4 of Table A.1 will always be twice the sum of the corresponding entries from column 3 of Table A.1. Denote $\mathcal{B} = \sum_{j=1}^N b_j$ where $b_j = \min \{l(\mathbf{z}_j), 3 - l(\mathbf{z}_j)\}$.

$$\begin{aligned} \text{Then } Ent &= \frac{1}{N} \mathcal{B} \text{ and} \\ kw &= \frac{1}{9N} 2\mathcal{B} = \frac{2}{9} Ent \end{aligned}$$

■

Note that this only holds for the case when there are 3 classifiers. If there are four or more classifiers there is no linear relationship between the values for kw and Ent as in the table.

Appendix B

Data Sets Used in this Thesis

Many of the datasets used in this thesis come from the UCI repository of machine learning¹ and the others come from the ELENA database². This appendix gives a more detailed look at each of the data sets used.

A summary of the basic statistics and an outline of the problem involved with each dataset is shown in the table below.

Dataset	Features	Classes	Examples	Problem
Sonar	60	2	208	tell mines from rocks from sonar signals
Glass	9	7	214	identify source of glass fragment
Haberman	3	2	306	survival likelihood after breast cancer operation
Ecoli	7	8	336	localisation site of Ecoli bacteria
Liver	6	2	345	identify likely sufferers of liver disorders
Ionosphere	34	2	351	tell good from bad radar images
votes	16	2	435	tell democrats from republicans from voting records
Wisconsin breast cancer	30	2	569	tell benign from malignant breast masses
Pima Indian diabetes	8	2	768	tell female Pima Indians with and without diabetes
Vehicle	18	4	846	classify vehicle from its silhouette
German	23	2	1000	tell good from bad credit ratings in Germany
Phoneme	5	2	5404	distinguish nasal from oral vowels

¹available from <http://www.ics.uci.edu/~mllearn/MLRepository.html>

²available via anonymous ftp at <ftp.dice.ucl.ac.be>, directory `pub/neural-nets/ELENA/databases`

B.1 The Sonar Identification dataset

Source: Terry Sejnowski, now at the Salk Institute and the University of California at San Diego. The data set was developed in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center.

Maintained by: Scott E. Fahlman

The Problem: The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. The object is to develop a system capable of distinguishing between a mine and a rock. The mine patterns were obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The rock patterns were obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

Statistics: 60 Features, 2 Classes, 208 Examples.

The features: Each pattern is a set of 60 features which are all numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.

The classes: mines (111), rocks (97).

B.2 The Glass Identification dataset

Created by: B. German, Central Research Establishment Home Office Forensic Science Service Aldermaston, Reading, Berkshire RG7 4PN

Donated by: Vina Spiehler, Ph.D., DABFT Diagnostic Products Corporation (213) 776-0180 (ext 3014)

Date: September, 1987

The Problem: To identify the source of a fragment of glass. The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified!

Statistics: 9 Features, 7 Classes, 214 Examples.

The features: 1. RI: refractive index, 2. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10), 3. Mg: Magnesium, 4. Al: Aluminium, 5. Si: Silicon, 6. K: Potassium, 7. Ca: Calcium, 8. Ba: Barium, 9. Fe: Iron

The classes: 1. building windows float-processed (70), 2. building windows non-float-processed (76), 3. vehicle windows float-processed (17), 4. vehicle windows non-float-processed (none in this database) (0), 5. containers (13), 6. tableware (9), 7. headlamps (29).

B.3 The Haberman dataset

Donated by: Tjen-Sien Lim (limt@stat.wisc.edu)

Date: March 4, 1999

The Problem: The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

Statistics: 3 Features, 2 Classes, 306 Examples.

The features: 1. Age of patient at time of operation, 2. Patient's year of operation, 3. Number of positive axillary nodes detected.

The classes: Survival status (class attribute) 1 = the patient survived 5 years or longer (225) 2 = the patient died within 5 years (81)

B.4 The Ecoli dataset

Created and maintained by: Kenta Nakai Institute of Molecular and Cellular Biology Osaka, University 1-3 Yamada-oka, Suita 565 Japan nakai@imcb.osaka-u.ac.jp <http://www.imcb.osaka-u.ac.jp/nakai/psort.html>

Donated by: Paul Horton (paulh@cs.berkeley.edu)

Date: September, 1996

The Problem: Identifying the localization site of the ecoli proteins.

Statistics: 7 Features, 8 Classes, 336 Examples.

The features: 1. mcg: McGeoch's method for signal sequence recognition. 2. gvh: von Heijne's method for signal sequence recognition. 3. lip: von Heijne's Signal Peptidase II consensus sequence score. 4. chg: Presence of charge on N-terminus of predicted lipoproteins. 5. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins. 6. alm1: score of the ALOM membrane spanning region prediction program. 7. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence.

The classes: 1. cp (cytoplasm) (143) 2. im (inner membrane without signal sequence) (77) 3. pp (periplasm) (52) 4. imU (inner membrane, uncleavable signal sequence) (35) 5. om (outer membrane) (20) 6. omL (outer membrane lipoprotein) (5) 7. imL (inner membrane lipoprotein) (2) 8. imS (inner membrane, cleavable signal sequence) (2)

B.5 The Liver dataset

Created by: BUPA Medical Research Ltd.

Donated by: Richard S. Forsyth 8 Grosvenor Avenue Mapperley Park Nottingham NG3 5DX 0602-621676

Date: 5/15/1990

The Problem: Seems to be the identification of liver disorders in male patients. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes the record of a single male individual.

Statistics: 6 Features, 2 Classes, 345 Examples.

The features: 1. mcv mean corpuscular volume 2. alkphos alkaline phosphatase 3. sgpt alamine aminotransferase 4. sgot aspartate aminotransferase 5. gammagt gamma-glutamyl transpeptidase 6. drinks number of half-pint equivalents of alcoholic beverages drunk per day

The classes: It appears that the two classes are those with a liver disorder and those without a liver disorder, however it is not clear which group is which. Class 1 (145), class 2 (200).

B.6 The Johns Hopkins University Ionosphere dataset

Source: Space Physics Group Applied Physics Laboratory Johns Hopkins University
Johns Hopkins Road Laurel, MD 20723 **Donated by:** Vince Sigillito (vgs@aplcn.apl.jhu.edu)

Date: 1989

The Problem: To classify radar signals as either ‘Good’ or ‘Bad’. This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. ‘Good’ radar returns are those showing evidence of some type of structure in the ionosphere. ‘Bad’ returns are those that do not; their signals pass through the ionosphere.

Statistics: 34 Features, 2 Classes, 351 Examples.

The features: Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal. This results in 34 features, all of which are continuous numbers.

The classes: Good (225), Bad (126).

B.7 The 1984 United States Congressional Voting Records dataset

Source: Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985.

Donated by: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

Date: 27 April 1987

The Problem: To identify whether a person is a democrat or a republican based upon their voting record on various key issues. This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac (CQA). The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

Statistics: 16 Features, 2 Classes, 435 Examples.

The features: 1. handicapped-infants, 2. water-project-cost-sharing, 3. adoption-of-the-budget-resolution, 4. physician-fee-freeze, 5. el-salvador-aid, 6. religious-groups-in-schools, 7. anti-satellite-test-ban, 8. aid-to-nicaraguan-contras, 9. mx-missile, 10. immigration, 11. synfuels-corporation-cutback, 12. education-spending, 13. superfund-right-to-sue, 14. crime, 15. duty-free-exports, 16. export-administration-act-south-africa

The options for each vote were y or n or ? for an unknown position. We have converted this to 1-no, 2-yes, 3-unknown position. **The classes:** Democrat (267), Republican (168).

B.8 The Wisconsin Breast Cancer dataset

Created by: Dr. William H. Wolberg, General Surgery Dept., University of Wisconsin, Clinical Sciences Center, Madison, WI 53792 wolberg@eagle.surgery.wisc.edu,

W. Nick Street, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 street@cs.wisc.edu 608-262-6619,

Olvi L. Mangasarian, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 olvi@cs.wisc.edu

Donated by: Nick Street

Date: November 1995

The Problem: to determine whether a mass within a breast is malignant or benign. Features are computed from a digitised image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Statistics: 30 Features, 2 Classes, 569 Examples.

The features: Ten real-valued features are computed for each cell nucleus: a) radius (mean of distances from centre to points on the perimeter) b) texture (standard deviation of gray-scale values) c) perimeter d) area e) smoothness (local variation in radius lengths) f) compactness ($\frac{\text{perimeter}^2}{\text{area}-1.0}$) g) concavity (severity of concave portions of the contour) h) concave points (number of concave portions of the contour) i) symmetry j) fractal dimen-

sion ('coastline approximation' - 1)

The mean, standard error, and 'worst' or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, feature 1 is Mean Radius, feature 11 is Radius SE, feature 21 is Worst Radius.

The classes: Benign (357), Malignant (212).

B.9 The Pima Indian Diabetes dataset

Created by: National Institute of Diabetes and Digestive and Kidney Diseases

Donated by: Vincent Sigillito (vgs@aplcn.apl.jhu.edu) Research Center, RMI Group Leader Applied Physics Laboratory The Johns Hopkins University Johns Hopkins Road Laurel, MD 20707 (301) 953-6231

Date: 9 May 1990

The Problem: The problem is to identify whether a female of Pima Indian descent exhibits signs of diabetes. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organisation criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives near Phoenix, Arizona, USA.

Statistics: 8 Features, 2 Classes, 768 Examples.

The features: 1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test 3. Diastolic blood pressure (mm Hg) 4. Triceps skin fold thickness (mm) 5. 2-Hour serum insulin (μ U/ml) 6. Body mass index ($\frac{\text{weight in kg}}{(\text{height in m})^2}$) 7. Diabetes pedigree function 8. Age (years)

The classes: no diabetes (500), tested positive for diabetes (268).

B.10 The Vehicle Silhouette Identification dataset

Created by: Turing Institute, Glasgow, Scotland.

Donated by: Drs.Pete Mowforth and Barry Shepherd Turing Institute George House 36 North Hanover St. Glasgow G1 2AD

Contact: Alistair Sutherland Statistics Dept. Strathclyde University Livingstone Tower 26 Richmond St. GLASGOW G1 1XH Great Britain Tel: 041 552 4400 x3033 Fax: 041 552 4711 e-mail: alistair@uk.ac.strathclyde.stams

Date: 1986-1987

The Problem: To distinguish between four models of vehicle from their silhouettes. This data was originally gathered at the TI in 1986-87 by JP Siebert. It was partially financed by Barr and Stroud Ltd. The original purpose was to find a method of distinguishing 3D objects within a 2D image by application of an ensemble of shape feature extractors to the 2D silhouettes of the objects. Four 'Corgi' model vehicles were used for the experiment:

a double decker bus, Chevrolet van, Saab 9000 and an Opel Manta 400. This particular combination of vehicles was chosen with the expectation that the bus, van and either one of the cars would be readily distinguishable, but it would be more difficult to distinguish between the cars. The images were acquired by a camera looking downwards at the model vehicle from a fixed angle of elevation (34.2 degrees to the horizontal). The vehicles were placed on a diffuse back-lit surface (light-box). The vehicles were painted matt black to minimise highlights. The vehicles were rotated and their angle of orientation was measured using a radial graticule beneath the vehicle. 0 and 180 degrees corresponded to ‘head on’ and ‘rear’ views respectively while 90 and 270 corresponded to profiles in opposite directions. Two sets of 60 images, each set covering a full 360 degree rotation, were captured for each vehicle. The vehicle was rotated by a fixed angle between images. A further two sets of images were captured with the camera at elevations of 37.5 degrees and 30.8 degrees.

Statistics: 18 Features, 4 Classes, 846 Examples.

The features:

1. Compactness $\frac{(\text{average perim})^2}{\text{area}}$,
2. Circularity $\frac{(\text{average radius})^2}{\text{area}}$,
3. Distance Circularity $\frac{\text{area}}{\text{av distance from border}}$,
4. Radius ratio $\frac{\text{max.rad}-\text{min.rad}}{\text{av.radius}}$,
5. Pr.Axis aspect ratio $\frac{\text{minor axis}}{\text{major axis}}$,
6. Max.Length aspect ratio $\frac{\text{length perp. max length}}{\text{max length}}$,
7. Scatter Ratio $\frac{\text{inertia about minor axis}}{\text{inertia about major axis}}$,
8. Elongatedness $\frac{\text{area}}{(\text{shrink width})^2}$,
9. Pr.Axis Rectangularity $\frac{\text{area}}{\text{pr.axis length} \times \text{pr.axis width}}$,
10. Scaled variance along major axis $\frac{2\text{nd order moment about minor axis}}{\text{area}}$,
11. Scaled variance along minor axis $\frac{2\text{nd order moment about major axis}}{\text{area}}$,
12. Scaled radius of gyration $\frac{mavar+mivar}{\text{area}}$,
13. Skewness about major axis $\frac{3\text{rd order moment about major axis}}{\sigma_{min}^3}$,
14. Skewness about minor axis $\frac{3\text{rd order moment about minor axis}}{\sigma_{maj}^3}$,
15. Kurtosis about minor axis $\frac{4\text{th order moment about major axis}}{\sigma_{min}^4}$,
16. Kurtosis about major axis $\frac{4\text{th order moment about minor axis}}{\sigma_{maj}^4}$,
17. Hollows ratio $\frac{\text{area of hollows}}{\text{area of bounding polygon}}$,

Where σ_{maj}^2 is the variance along the major axis and σ_{min}^2 is the variance along the minor axis, and area of hollows= area of bounding poly-area of object The area of the bounding polygon is found as a side result of the computation to find the maximum length.

The classes: Opel (212), Saab (217), bus (218), van (199).

B.11 The German Credit dataset

Source: Professor Dr. Hans Hofmann Institut für Statistik und Ökonometrie Universität Hamburg FB Wirtschaftswissenschaften Von-Melle-Park 5 2000 Hamburg 13

The Problem: To identify whether a prospective customer is a ‘good’ or ‘bad’ credit risk for the purposes of obtaining a loan. The original dataset, in the form provided by Prof. Hofmann, contained categorical/symbolic attributes. For algorithms that need numerical attributes, Strathclyde University produced a second version of the dataset. This data has been edited and several indicator variables added to make it suitable for algorithms which cannot cope with categorical variables. Several attributes that are ordered categorically (such as attribute 17) have been coded as integer. This is the version which we have used.

Statistics: 23 Features, 2 Classes, 1000 Examples.

The original features are: 1. Status of existing checking account, 2. Duration in months, 3. Credit history, 4. Purpose, 5. Credit amount, 6. Savings account / bonds, 7. Present employment since?, 8. Installment rate in percentage of disposable income, 9. Personal status and sex, 10. Other debtors / guarantors, 11. Present residence since, 12. Property?, 13. Age in years, 14. Other installment plans, 15. Housing, 16. Number of existing credits at this bank, 17. Job, 18. Number of people liable to provide maintenance for, 19. Telephone?, 20. Foreign worker?

The 23 features in the version of the dataset we have used have been derived from these by converting to numerical features.

The classes: Good risk (700), Bad risk (300).

B.12 The Phoneme dataset

Source: Dominique VAN CAPPEL (33) 92 96 45 44 THOMSON-SINTRA, 525 route des Dolines, BP157, F-06903 Sophia Antipolis Cedex, France

The Problem: To identify whether spoken vowels are Nasal or Oral in nature. This database contains vowels coming from 1809 isolated syllables (for example: pa, ta, pan,...).

Statistics: 5 Features, 2 Classes, 5404 Examples. **The features:** Five different attributes were chosen to characterise each vowel: they are the amplitudes of the five first harmonics AH_i , normalised by the total energy Ene (integrated on all the frequencies): AH_i/Ene . Each harmonic is signed: positive when it corresponds to a local maximum of the spectrum and negative otherwise. Three observation moments have been kept for each vowel to obtain 5427 different instances: - the observation corresponding to the maximum total energy Ene . - the observations taken 8 msec before and 8 msec after - the observation corresponding to this maximum total energy.

From these 5427 initial values, 23 instances for which the amplitude of the 5 first harmonics was zero were removed, leading to the 5404 instances of the present database.

The classes: Nasal vowels, Oral vowels.

Bibliography

- [1] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach. *Journal of Machine Learning Research*, 1:113–141, 2000.
- [2] S.G. Alsing, K.W. Bauer, and Jr. J.O. Miller. A multinomial selection procedure for evaluating pattern recognition algorithms. *Pattern Recognition*, 35:2397–2412, 2002.
- [3] H. Altincay and M. Demirekler. An information theoretic framework for weight estimation in the combination of probabilistic classifiers for speaker identification. *Speech Communication*, 30:255–272, 2000.
- [4] R. Battiti and A.M. Colla. Democracy in neural nets: voting schemes for classification. *Neural Networks*, 7(4):691–707, 1994.
- [5] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.
- [6] G. Blanchard, G. Lugosi, and N. Vayatis. On the rate of convergence of regularised boosting classifiers. *Journal of Machine Learning Research (special issue on learning theory)*, 4:861–894, October 2003.
- [7] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [8] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.
- [9] L. Breiman. Combining predictors. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, chapter 2, pages 31–50. Springer-Verlag, 1999.
- [10] L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40:229–242, 2000.
- [11] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
- [12] L. Breiman. Using iterated bagging to debias regressions. *Machine Learning*, 45:261–277, 2001.

- [13] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- [14] N.V. Chawla, L.O. Hall, K.W. Bowyer, Jr. T.E. Moore, and W.P. Kegelmeyer. Distributed pasting of small votes. In F. Roli and J. Kittler, editors, *Third International Workshop, Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 52–61, Cagliari, Italy, 2002. MCS2002, Springer.
- [15] C.C. Chibelushi, F. Deravi, and J.S.D. Moore. Adaptive classifier integration for robust pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 29(6):902–907, December 1999.
- [16] C. Cortes and V. Vapnik. Support-vector networks. *machine learning*, 20(3):273–297, September 1995.
- [17] P. Cunningham. Overfitting and diversity in classification ensembles based on feature selection. Technical Report TCD-CS-2000-07, Department of Computer Science, Trinity College Dublin, 2000.
- [18] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. Technical Report TCD-CS-2000-02, Department of Computer Science, Trinity College Dublin, 2000.
- [19] D. T. Denison. Boosting with bayesian stumps. *Statistics and Computing*, 11:171–178, 2001.
- [20] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):1–19, 1999.
- [21] T.G. Dietterich. Ensemble methods in machine learning. In F. Roli and J. Kittler, editors, *First International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, Cagliari, Italy, 2000. MCS2000, Springer.
- [22] H. Drucker. Boosting neural networks. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, chapter 3, pages 51–78. Springer-Verlag, 1999.
- [23] H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6:1289–1301, 1994.
- [24] R.O. Duda, P.E.Hart, and D.G. Stork. *Pattern Classification*, chapter 9, pages 453–516. John Wiley & sons, New York, 2nd edition, 2001.

- [25] N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. *Theoretical Computer Science*, 284:67–108, 2002.
- [26] R.P.W. Duin. Prtools version 2: a matlab toolbox for pattern recognition. Pattern Recognition Group, Delft University of Technology, June 1997. available at <http://www.ph.tn.tudelft.nl/prtools>.
- [27] G. Eibl and K. P. Pfeiffer. How to make adaboost.m1 work for weak base classifiers by changing only one line of the code. *Lecture Notes in Artificial Intelligence*, 2430:72–83, 2002.
- [28] J.L. Fleiss. *Statistical methods for Rates and Proportions*. John Wiley & Sons, 1981.
- [29] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
- [30] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Third International Conference*. Machine Learning, 1996.
- [31] Y. Freund and R.E. Schapire. A decision-theoretic generalisation of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [32] Y. Freund and R.E. Schapire. Discussion of the paper “arcing classifiers” by leo breiman. *The Annals of Statistics*, 26(3):824–832, 1998.
- [33] Y. Freund and R.E. Schapire. Discussion of the paper “additive logistic regression: A statistical view of boosting by jerome friedman, trevor hastie and robert tibshirani. *The Annals of Statistics*, 38(2):391–393, April 2000.
- [34] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.
- [35] G. Fumera and F. Roli. Performance analysis and comparison of linear combiners for classifier fusion. In *SSPR*, pages 110–120, Windsor, Canada, 2002. SSPR.
- [36] S. Ghahramani. *Fundamentals of Probability*. Prentice Hall, N.J., second edition, 2000.
- [37] G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22:25–33, 2001.
- [38] G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification processes. *Image and Vision Computing Journal*, 19:697–705, 2001.

- [39] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
- [40] S. Hashem. Treating harmful collinearity in neural network ensembles. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, chapter 5, pages 101–125. Springer-Verlag, 1999.
- [41] T.K Ho. The random space method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- [42] S. Hoche and S. Wrobel. Scaling boosting by margin-based inclusion of features and relations (extended abstract). In *Machine Learning Workshop FGML*, 2002.
- [43] A. Hough. *Physiotherapy in Respiratory Care: A problem-solving approach to respiratory and cardiac management*. Stanley Hornes (Publishers) Ltd, second edition, 1997.
- [44] Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:90–93, 1995.
- [45] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
- [46] T.M. Jorgensen and C. Linneberg. Feature weighted ensemble classifiers - a modified decision scheme. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 218–227, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [47] Y. Kim. Averaged boosting: A noise-robust ensemble method. *Lecture Notes in Artificial Intelligence*, 2637:388–393, 2003.
- [48] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–238, 1998.
- [49] E.M. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473–490, May 2000.
- [50] R. Kohavi and D.H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In L. Saitta, editor, *Machine Learning: Proc. 13th International Conference*, pages 275–283. Morgan Kaufman, 1996.

- [51] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 231–238. MIT Press, Cambridge MA, 1995.
- [52] L.I. Kuncheva. Two-level classification schemes in medical diagnostics. *International Journal of Biomedical Computing*, 32:197–210, 1993.
- [53] L.I. Kuncheva. *Fuzzy Classifier Design*. Number 49 in Studies in Fuzziness and Soft Computing. Springer Verlag, Berlin, 2000.
- [54] L.I. Kuncheva. Combining classifiers: Soft computing solutions. In S.K. Pal and A. Pal, editors, *Pattern Recognition: From Classical to Modern Approaches*, chapter 5, pages 427–452. World Scientific Publishing Co., Singapore, 2001.
- [55] L.I. Kuncheva. 'fuzzy' vs 'non-fuzzy' in combining classifiers designed by boosting. *IEEE Transactions on Fuzzy Systems*, 11(6):729–741, 2003.
- [56] L.I. Kuncheva. That elusive diversity in classifier ensembles. In *Proceedings of IbPRIA 2003*, Lecture Notes in Computer Science, pages 1126–1138, Mallorca, Spain, 2003. Springer-Verlag.
- [57] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34:299–314, 2001.
- [58] L.I. Kuncheva, F. Roli, G.L. Marcialis, and C.A. Shipp. Complexity of data subsets generated by the random subspace method: An experimental investigation. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 349–358, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [59] L.I. Kuncheva and C.J. Whitaker. Feature subsets for classifier combination: an enumerative experiment. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [60] L.I. Kuncheva and C.J. Whitaker. Ten measures of diversity in classifier ensembles: limits for two classifiers. In *IEEE Workshop on Intelligent Sensor Processing*, Birmingham, UK, 10 2001. ISP2001.
- [61] L.I. Kuncheva and C.J. Whitaker. Using diversity with three variants of boosting: Aggressive, conservative, and inverse. In F. Roli and J. Kittler, editors, *Third International Workshop, Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 81–90, Cagliari, Italy, 2002. MCS2002, Springer.

- [62] L.I. Kuncheva and C.J. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, 51:181–207, 2003.
- [63] L.I. Kuncheva, C.J. Whitaker, C.A. Shipp, and R.P.W. Duin. Is independence good for combining classifiers? In *Proc. 15th International Conference on Pattern Recognition*, volume 2, pages 169–171, Barcelona, Spain, 2000.
- [64] L.I. Kuncheva, C.J. Whitaker, C.A. Shipp, and R.P.W. Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, 6:22–31, 2003.
- [65] L. Lam. Classifier combinations: implementations and theoretical issues. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 78–86, Cagliari, Italy, 2000. MCS2000, Springer.
- [66] L. Lam and C.Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995.
- [67] L. Lam and C.Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 27(5):553–567, 1997.
- [68] Y. Liu and X. Yao. Negatively correlated neural networks for classification. In *3rd international Symposium on Artificial Life and Robotics*, pages 736–739, Japan, 1998. AROBIII’98.
- [69] Y. Liu and X. Yao. Simultaneous learning of negatively correlated neural networks. In *9th Australian Conference on Neural Networks*, pages 183–187, Brisbane, Australia, 1998. ACNN’98.
- [70] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.
- [71] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
- [72] S.W. Looney. A statistical technique for comparing the accuracies of several classifiers. *Pattern recognition Letters*, 8:5–9, July 1998.
- [73] F. Lozano and V. Koltchinskii. Direct optimization of simple cost functions of the margin. In *NF2002*, 2002.
- [74] D.D. Margineantu and T.G. Dietterich. Pruning adaptive boosting. In *14th International Conference on Machine Learning*, pages 358–366, 1997.

- [75] L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38:243–255, 2000.
- [76] F. Masulli and G. Valentini. Effectiveness of error correcting output codes in multi-class learning problems. In F. Roli and J. Kittler, editors, *First International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science 1857, pages 107–116, Cagliari, Italy, 2000. MCS2000, Springer.
- [77] F. Masulli and G. Valentini. Dependence among codeword bits errors in ecoc learning machines: An experimental analysis. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science 2096, pages 158–167, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [78] S. Merler, C. Furlanello, B. Larcher, and A. Sboner. Automatic model selection in cost-sensitive boosting. *Information Fusion*, 4:3–10, 2003.
- [79] J. Mingers. An empirical comparison of pruning methods for decision trees. *Machine Learning*, 4(2):227–243, 1989.
- [80] R. Nock and M. Sebban. A bayesian boosting theorem. *Pattern Recognition Letters*, 22:413–419, 2001.
- [81] N.C. Oza. Boosting with averaged weighting vectors. In *Fourth international workshop on multiple classifier systems*, pages 15–24, Guildford, UK, 2003. Springer-Verlag Heidelberg.
- [82] D. Partridge and W.J. Krzanowski. Distinct failure diversity in multiversion software. (personal communication to L.I. Kuncheva 1999).
- [83] D. Partridge and W.J. Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information & Software Technology*, 39:707–717, 1997.
- [84] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, California, 1993.
- [85] J.R. Quinlan. Bagging, boosting and c4.5. In *13th National Conference on Artificial Intelligence*, pages 725–730, Cambridge, MA, 1996.
- [86] G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.
- [87] G. Rätsch and M. K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 2003. (submitted) available online.

- [88] G. Ridgeway. Looking for lumps: boosting and bagging for density estimation. *Computational Statistics & Data Analysis*, 38:379–392, 2002.
- [89] F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 78–87, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [90] B.E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3/4):373–383, 1996.
- [91] D. Ruta and B. Gabrys. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 399–408, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [92] R. E. Schapire. The boosting approach to machine learning: An overview. MSRI Workshop on Nonlinear Estimation and Classification, 2002. available at <http://www.cs.princeton.edu/~schapire/whatsnew.html>.
- [93] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [94] R.E. Schapire. Theoretical views of boosting. In *Computational Learning Theory: Fourth European Conference*, pages 1–10. EuroCOLT'99, 1999.
- [95] R.E. Schapire, Y. Freund, P. Bartlett P, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 10 1998.
- [96] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
- [97] C.A. Shipp and L.I. Kuncheva. Four measures of data complexity for bootstrapping, splitting and feature sampling. In *Computational Intelligence: Methods and Applications*, pages 429–435, Bangor, Wales, UK, 2001. CIMA2001, ICSC Academic Press.
- [98] C.A. Shipp and L.I. Kuncheva. An investigation into how adaboost affects classifier diversity. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 203–208, Annecy, France, 2002. IPMU2002.

- [99] C.A. Shipp and L.I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3:135–148, 2002.
- [100] D.B. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *Proc. American Association for Artificial Intelligence, Integrating Multiple Learned Models Workshop*. AAAI, 1996.
- [101] M. Skurichina and R.P.W. Duin. Bagging and the random subspace method for redundant feature spaces. In F. Roli and J. Kittler, editors, *2nd International Conference on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1–10, Cambridge, UK, 2001. MCS2001, Springer-Verlag.
- [102] M. Skurichina and R.P.W. Duin. The role of combining rules in bagging and boosting. *Lecture Notes in Computer Science*, 1876:631–640, 2001.
- [103] M. Skurichina, L.I. Kuncheva, and R.P.W. Duin. Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy. In F. Roli and J. Kittler, editors, *Third International Workshop, Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 62–71, Cagliari, Italy, 2002. MCS2002, Springer.
- [104] E. Takimoto and A. Maruoka. Top-down decision tree learning as information based boosting. *Theoretical Computer Science*, 292:447–464, 2003.
- [105] C. Tamon and J. Xiang. On the boosting pruning problem. *Lecture Notes in Artificial Intelligence*, 1810:404–412, 2000.
- [106] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.
- [107] K. Tumer and J. Ghosh. Error correlation and error reduction. *ensemble classifiers, Connection Science*, 8(3/4):385–404, 1996.
- [108] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, chapter 6, pages 127–161. Springer-Verlag, 1999.
- [109] N. Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 22(2):207–215, February 2000.
- [110] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

- [111] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis. Soft combination of neural classifiers: a comparative study. *Pattern Recognition Letters*, 20:429–444, 1999.
- [112] K. D. Wernecke. A coupling procedure for discrimination of mixed data. *Biometrics*, 48:497–506, 1992.
- [113] J. Wickramaratna, S. Holden, and B. Buxton. Performance degradation in boosting. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, volume 2096 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2001.
- [114] T. Windeatt and G. Ardeshir. Boosted tree ensembles for solving multiclass problems. In F. Roli and J. Kittler, editors, *Third International Workshop, Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 42–51, Cagliari, Italy, 2002. MCS2002, Springer.
- [115] K. Woods, Jr. W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.
- [116] L. Xu, A. Kryzak, and C.Y. Suen. Methods for combining multiple classifiers and their application to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:418–435, 1992.
- [117] X. Yao and Y. Liu. Neural networks for breast cancer diagnosis. In *1999 Congress on Evolutionary Computation*, volume 3, pages 1760–1767, Piscataway, NJ, USA, 1999. IEEE Press.
- [118] G.U. Yule. On the association of attributes in statistics. *Phil. Trans. A*, 194:257–319, 1900.
- [119] Z. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137:239–263, 2002.