DEPARTEMENT NATUURKUNDE
FACULTEIT WETENSCHAPPEN

UNIVERSITEIT ANTWERPEN

Feature Selection and Classifier Ensembles: A Study on Hyperspectral Remote Sensing Data

Kenmerk Selectie en Classificatie Ensembles: Een Studie op Hyperspectrale Afstandswaarnemingsdata

Proefschrift voorgelegd tot het behalen van de graad van

Doctor in de Wetenschappen

aan de Universitaire Instelling Antwerpen te verdedigen

door

ShiXin Yu

Promotoren:
Prof. dr. Paul Scheunders,
Prof. dr. Dirk Van Dyck.

Antwerpen, 2003

Feature Selection and Classifier Ensembles: A Study on Hyperspectral Remote Sensing Data

by

ShiXin Yu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Physics)
in The University of Antwerp
Antwerp, 2003

— *To my beloved parents*

ShouMei Cao and ShanHong Yu

ACKNOWLEDGEMENTS

Someone says that completing the requirements for a doctoral degree is just like running a marathon. Now at the time when I'm sprinting towards the final goal of this long race, I would like to acknowledge the following people who, in one way or another, influenced or supported me along the course.

First of all, I thank my dissertation advisor Prof. dr. Paul Scheunders for his enthusiastic supervision, continuous encouragement, great patience and effort for polishing my English writings. Thanks also go to him for giving me the complete time and freedom to develop and define my own research direction. While this has proven to be somewhat tough at first — it appeared that I just lost my way, wandering about where to go at the very beginning of my research, I have come to appreciate his wisdom and the way he trains the student to begin with the research. I have benefited very much from his guidance. In addition, I wish to extend my appreciation to Prof. dr. Dirk Van Dyck, director of the VisionLab and also my dissertation co-advisor, for providing me with a large and quiet office where I enjoyed my time to stay.

I am indebted to the members of the doctoral committee for carefully reading the manuscript of this dissertation.

I am very grateful to Hilde, the physics department secretary, for the kind help.

Thanks are also due to the members of the VisionLab. In particular, I thank Werner for translating the summary of this dissertation into dutch.

However, I could not finish without dedicating this dissertation to the two most important people in my life: my mother and my father. Nothing could ever eclipse the true love, the unconditional dedication, and the consistent support they have shown me during both the highs and lows over the last thirty or so years. I could never have achieved my dreams without you both. Thank you! Last but not least, I thank my sisters for their invaluable support and continual encouragement from thousands of miles away, especially for taking care of our parents during all these years while it should have been my duty.

Antwerp, Jan. 2003

ShiXin Yu

郁 士欣

SUMMARY

Feature Selection and Classifier Ensembles:

A Study on Hyperspectral Remote Sensing Data

by

ShiXin Yu

Pattern recognition is the research area that studies the operation and design of systems that recognize patterns in data [38].

In this dissertation, two important aspects of pattern recognition: the classifier ensemble problem, and the feature selection problem, are studied on hyperspectral remote sensing data. Rapid advances in sensor technology have made it recently possible to collect hyperspectral remote sensing data which spans typically from 200 to 400 spectral bands. Such high dimensionality, on one hand, provides us with more potential discrimination power for classification tasks. On the other hand, the classification performance improves up to a limited point as additional features are added, and then deteriorates due to the limited number of training samples. This shows the importance of feature reduction as a critical pre-processing step. Feature reduction includes feature selection and feature extraction. In part due to

the difficulty in interpreting the transformed features through feature extraction, feature selection is emphasized instead.

Another problem with classification of high dimensionality is that, the discrimination between classes becomes much more difficult, due to the fact that, the number of training samples is unlikely to catch up with the increase of dimensionality. Therefore another important topic for data analysis in hyperspectral remote sensing is the improvement of the classification performance. In this dissertation, this is achieved by studying the idea of fusing existing classification schemes to further improve classification performance. Because of the so many attractive advantages of the nearest neighbor classifier, its corresponding ensembles are focused on in this dissertation.

The roadmap of this dissertation is the following. First we study classifier ensemble methods. We then initiate the idea of using ensembled learning as a means to evaluate the merit of feature subsets during the selection stage for feature selection. Following this roadmap, two general overviews are first given on classifier ensembles (Chapter II) and feature selection (Chapter V) respectively and a categorization scheme is given for each.

In the frame of classifier ensembles, a comprehensive empirical study of nearest neighbor classsifier ensembles is carried out while taking into account the scheme of bias plus variance decomposition of the error rate (Chapter IV and Chapter VII). With the goal of both improving classification performance and decreasing storage requirements simultaneously, a method called CNN-ECOC, which utilizes the Condensed Nearest Neighbors (CNN) algorithm in conjunction with the technique of Error Correcting Output Codes (ECOC) is presented (Section 4.3.5). Another variant called kNN-ECOC-RS, which takes advantage of randomly selected subspaces in conjunction with the ECOC method to further improve the performance of nearest neighbors is then suggested as a by-product (Section 4.3.4).

In the frame of feature selection, we undertake the first study that hybridizes genetic algorithms with the ensembled learning scheme. The performance gains of using ensembled learning is then demonstrated through experiments (Chapter VII). In the mean time, a categorization scheme on the existing genetic feature selection methods is conducted (Chapter VI).

In conclusion, the contributions of this dissertation are summarized as follows.

- A categorization scheme on classifier ensemble methods.
- A categorization scheme on feature selection methods.
- A taxonomy of classification methods using the k Nearest Neighbor learning algorithm.
- A categorization scheme on genetic feature selectors.
- The CNN-ECOC methods, which takes advantage of the Condensed Nearest Neighbor (CNN) algorithm in conjunction with the technique of Error Correcting Output Codes (ECOC). This can be seen as the first major contribution of this dissertation.
- The kNN-ECOC-RS method, which takes advantage of randomly selected subspaces from the whole feature space, in conjunction with the ECOC technique. This method is suggested as a by-product.
- We initiate and advocate the idea of using ensembled learning in conjunction with genetic algorithms to perform feature selection. This can be seen as the second major contribution of this dissertation.
- A performance comparison between various k nearest neigbor ensembles, which takes into account the paradigm of bias plus variance decomposition of the error

rate. The classification algorithm, which has a small error rate and ideally at the same time also has a small bias and a small variance, is usually favored as a better choice.

- A performance comparison between the genetic search and the sequential floating forward search, which takes into account the number of subset evaluations, provides a pragmatic view of time complexity of these two search methods on hyperspectral data.

TABLE OF CONTENTS

DEDICATION . . . . **ii**

ACKNOWLEDGEMENTS . . . . **iii**

SUMMARY . . . . **v**

NOTATION . . . . **xi**

CHAPTER

I. Introduction 1

1.1 Background of Hyperspectral Remote Sensing 1
1.2 Statement of the Problem 3
1.3 Organization of the Dissertation 8

II. Classifier Ensembles: An Overview 11

2.1 Introduction 11
2.2 The Hypothesis: Why Classifier Ensembles Work 12
2.3 Categorization Scheme of Classifier Ensembles 14
2.3.1 Voting Classifier Ensembles 15
2.3.2 Classifier Ensembles Manipulating Training Samples 17
2.3.3 Classifier Ensembles with Different Input Feature Subsets 20
2.3.4 Heterogeneous Classifier Ensembles 21
2.3.5 Homogeneous Classifier Ensembles 23
2.3.6 Recursive Partition Classifier Ensembles 23
2.4 Summary 24

III. Nearest Neighbor Learning Algorithm Revisited 25

3.1 Introduction 25
3.2 Nearest Neighbor Classifier 26
3.3 Fuzzy Nearest Neighbor Classifier 27

IV. Ensemble Methods for Nearest Neighbor Learning Algorithm 30

4.1 Introduction 30
4.2 Error Correcting Output Codes (ECOC) 31
4.3 Ensemble Methods for Nearest Neighbor Classifiers 32
4.3.1 Nearest Neighbor Classification from Multiple Feature Subsets (MFS) 34
4.3.2 Voting over Multiple Condensed Nearest Neighbors (CNN) 34
4.3.3 Nearest Neighbor Classifiers with Small Prototype Sets: Skalak's NN Architecture 35

4.3.4 Nearest Neighbor Classifier with Error Correcting Output Codes and Feature Selection: kNN-ECOC-FS 39
4.3.5 Condensed Nearest Neighbors (CNN) with Error Correcting Output Codes: CNN-ECOC 40
4.4 Bias plus Variance Decomposition of the Error Rate 41

V. Feature Selection Methods: An Overview 44

5.1 Introduction 44
5.2 Relevance to the Concept: Weak and Strong Relevance 45
5.3 General Characteristics of Feature Selection Methods 46
5.4 Categorization Scheme of Feature Selection Methods 48
5.4.1 Filter Selection Model 50
5.4.2 Wrapper Selection Model 51
5.4.3 Embedded Selection Model 56

VI. Genetic Feature Selectors 57

6.1 Introduction 57
6.2 Genetic Algorithms 58
6.3 Categorization Scheme of Genetic Feature Selectors 60
6.3.1 Aggregation Selection with Parameter Variation 61
6.3.2 Pareto-based Selection 63

VII. Experiments and Discussion 65

7.1 Introduction 65
7.2 Experiments of Nearest Neighbor Classifier Ensembles 66
7.2.1 Experiment One 66
7.2.2 Experiment Two 72
7.2.3 Experiment Three 78
7.3 Summary and Remarks on Nearest Neighbor Classifier Ensembles 83
7.4 Experiments of Genetic Feature Selector 84
7.4.1 Experiment One 84
7.4.2 Experiment Two 85
7.5 Summary and Remarks on Genetic Feature Selector 89

VIII. Conclusion and Further Work 92

8.1 Conclusion 92
8.2 Further work 94

NEDERLANDSE SAMENVATTING **95**

LIST OF PUBLICATIONS **97**

BIBLIOGRAPHY **98**

NOTATION

The notation used in the dissertation follows the general conventions of mathematics and statistics.

e	Natural constant (like π), $e \approx 2.718$
$[x_1, x_2, \cdots, x_n]^T$	Transposed n-dimensional vector
$\underset{i}{\operatorname{argmax}} f_i$	Returns the largest value of f_i among all index i
$d(\mathbf{z}, \mathbf{x})$	Distance between two multi-dimensional vectors $\mathbf{z}$ and $\mathbf{x}$

The notation for classification schemes considered in the dissertation is as follows:

A training dataset T with m labeled examples $\mathbf{x}_i$ in Space $\mathbf{X}$ coming from y_i of predefined S classes in Space $\mathbf{Y}$, i.e.:

$T = \{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, m\}$

Samples $\mathbf{x}_i \in \mathbf{X}$, Labels $y_i \in \mathbf{Y} = \{1, 2, \ldots, S\}$

A classifier is a hypothesis H that predicts the corresponding y values, given new $\mathbf{x}$ values. All the classifiers are denoted as $h_1, h_2, \cdots$. In contrast, a target function F is the *true* corresponding relationship between $\mathbf{X}$ and $\mathbf{Y}$: $\mathbf{x}_i \longrightarrow y_i$.

CHAPTER I

Introduction

1.1 Background of Hyperspectral Remote Sensing

Earth observation by remote sensing has provided human being a global view of the Earth. Remote sensing is a versatile tool for exploring Earth and it involves the use of instruments or sensors to "capture" the spectral and spatial relations of objects and materials observable at a distance - typically from above them. An aerial photograph is a common example of a remotely sensed (by camera and film) product. There are numerous real world applications - these are typical: 1)monitoring forest tree species; 2) determining the status of a growing crop; 3) defining urban patterns; 4) delineating the extent of flooding; 5) recognizing rock types; 6) pinpointing areas of deforestation [37] [194].

Rapid advances in sensor technology have made it recently possible to collect remote sensing data from multispectral data which typically ranges around 20 $\sim$ 30 bands, to today's hyperspectral data, the spectral bands of which can span from 220 bands with 20-meter spatial resolution and 10 bits of dynamic range, e.g. the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) [10], to 300 bands in a range between 400 and 2500 nm and at a spatial ground resolution of 2 to 5 meter, e.g. the Airborne Prism EXperiment (APEX) [236]. The TRWIS III imaging spectrometer of the TRW Inc. [36] can produce as many as 384-spectral-band imagery with spatial resolutions spanning from less than 1 meter to more than 11 meters, and with spectral coverage from 380 to 2450 nm. Its spectral resolution is 5.25 nm in the visible/near infrared (380-1000 nm) and 6.25 nm in the short wave infrared (1000-2450 nm). The number of spectral bands tends to continue to increase with the rapid development of sensor technology. The sensor currently being developed can span to more than 400 spectral bands [183]. A hyperspectral image can be viewed as an image cube with as the third dimension the spectral domain represented by hundreds of narrow, contiguous spectral bands corresponding to the spectral reflectance. The plot of contiguous spectral reflectances can be compared with laboratory produced ones to identify, for example, some ores in the target area. The rapid development of hyperspectral sensor technology greatly extends the scope of traditional remote sensing, it not only provides the scientists in environmental and geoscience research communities much more power to explore the earth than ever but also provides many challenges for data analysis tasks.[1] Furthermore, the volume of hyperspectral data produced is staggering. The fast growing size of the hyperspectral database also requests the compression[2] research on it [196] [230] [258] [263] [158] [198] [233] [40] [220] [104] [283] [287]. In particular the compression algorithm should take advantage of the spectral nature of hyperspectral data.

Figure 1.1 shows an AVIRIS hyperspectral image, which contains 220 bands of 145×145 pixels, that is downloadable from [3], along with a groundtruth image, containing 16 classes as shown on Figure 1.2. This dataset is used to do experimental comparison in this dissertation. Each pixel $\mathbf{x}$ on this image is represented by a 220

[1] One obstacle for the research on hyperspectral remote sensing is due to the lack of data [184], despite the rapid advances in sensor technology and the importunate demands for the hyperspectral data, the general hyperspectral imagery markets are not yet ready to commercially support the industry for at least the next several years, due to a number of factors, e.g. distinct requirements from different market segments [187].

[2] Neural network is among the techniques used for compression [106].

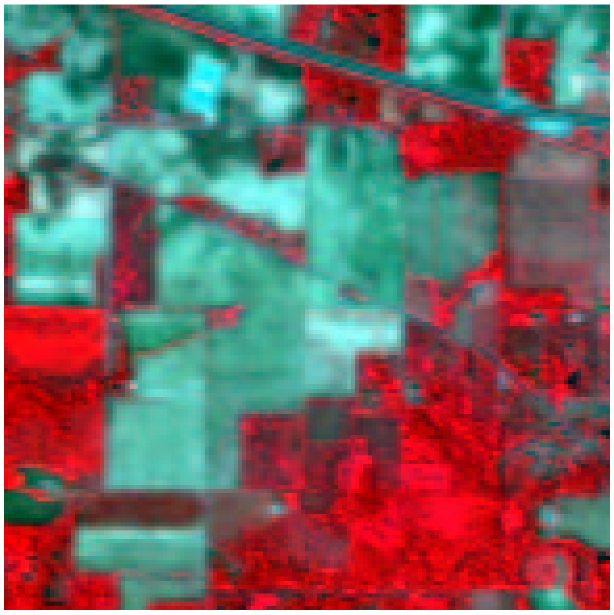


Figure 1.1: An example of a 220-band hyperspectral image (in simulated color infrared form).

dimensional vector, i.e.

$$\mathbf{x} = [x_1, x_2, ..., x_{220}]^T \tag{1.1}$$

The classification task here becomes to recognize new unknown objects given the predefined classes (ground truth or training classes). There are three views of the hyperspectral data: the image space, the spectral space and the feature space. Figure 1.1 is actually an image space. Figure 1.3 is an illustration of spectral space of three classes while Figure 1.4 shows its corresponding 2-dimensional feature space.

1.2 Statement of the Problem

Pattern recognition [259][3] [97] [238] is a vibrant research area. The term *"pattern recognition"* is meant in a broad sense, and generally speaking, it encompasses three

[3]Many researchers from the pattern recognition community speak highly of this book, and agree that "thus researchers from all walks of pattern recognition should get something out of this book", in part due to that it provides a better balance between theoretical and practical treatment of pattern recognition. For a long list of reseach monographs which could be useful for the understanding of pattern recognition, the reader is referred to the TUDelft pattern recognition group's page at [30]. Or for up-to-date books, simply input the key words of "pattern recognition" into the book search of large online book stores, such as *amazon.com*, *bn.com*, *buy.com*, etc, to find all the available up-to-date reference books on the topics relating to pattern recognition.

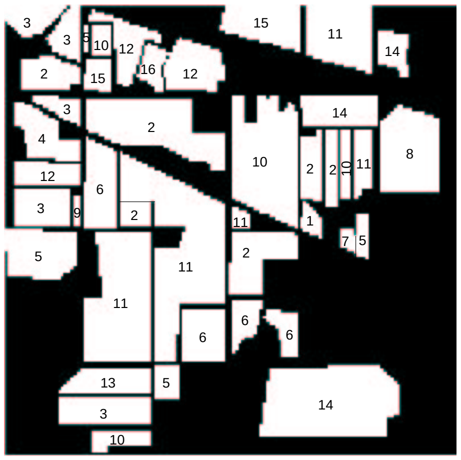


Figure 1.2: Groundtruth image containing 16 predefined classes; 1: Alfalfa; 2: Corn-notill; 3: Corn-min; 4: Corn; 5: Grass/pasture; 6: Grass/trees; 7: Grass/pasture-mowed; 8: Hay-windrowed; 9: Oats; 10: Soy-notill; 11: Soy-min till; 12: Soy-clean; 13: Wheat; 14: Woods; 15: Bldg-grass-trees- drives; 16: Stone-steel towers.

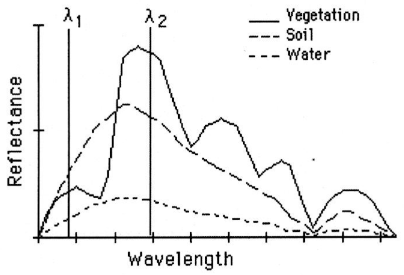


Figure 1.3: Spectral Space

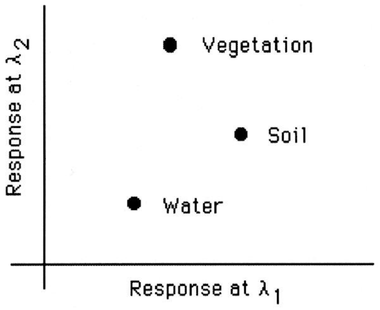


Figure 1.4: Feature Space

major categories: supervised learning, unsupervised learning, and semi-supervised learning (or partially supervised [154]). Supervised learning is a kind of machine learning where the learning algorithm is provided with a set of inputs for the algorithm along with the corresponding correct outputs, and learning involves the algorithm comparing its current actual output with the correct or target outputs, so that it knows what its error is, and modify things accordingly. In contrast to supervised learning, unsupervised learning signifies a mode of machine learning where the system is not told the ”right answer” — for example, it is not trained on pairs consisting of an input and the desired output. Instead the system is given the input patterns and is left to find interesting patterns, regularities, or clusterings among them [14] [27]. Semi-supervised learning can be thought to be a kind of supervised system where the unlabeled samples are incorporated somehow.

Figure 1.5 shows a typical supervised classification process for hyperspectral data analysis.

Typically, the classification performance improves up to a limited point as additional features are added, and then deteriorates. This is referred as the Hughes phenomenon[4] (or the peaking phenomenon)[139] as shown in Figure 1.6. The vertical axis is the mean recognition accuracy averaged from all possible classifiers. It is plotted as a function of measurement complexity on the horizontal axis. The more bands the hyperspectral data has, the greater the measurement complexity. The parameter m is the number of training samples. The Hughes phenomenon can be explained as follows. Let's consider a finite and fixed number of training samples. The accuracy of statistics estimation decreases as the dimensionality increases, leading to

[4]In my opinion, the term *curse of dimensionality* [56] [116] [114] is used to describe the difficult problems *in general* in the high dimensional multivariate analysis, e.g. Jacoby [146] uses the curse of dimensionality to describe the difficulty in visualization problem for high dimensional data analysis, therefore, I follow other researchers [4], and use *hughes phenomenon* to describe the difficulty in the high-dimensional data classification problem in this dissertation.

the deterioration of classification accuracy (Figure 1.7(b)). Although increasing the number of the spectral bands can potentially provides more class separability, this positive effect is diluted by poor statistical parameter estimation (Figure 1.7(a)). Consequently, the performance of the classifier with a fixed sample size may degrade with an increase in the number of features as illustrated in Figure 1.7(c). Although it seems logical to infer that a large number of features would give much more discriminating power, a high-dimensional space is, in fact, mostly empty [132] with modest number of samples, hence, the importance of feature reduction.

Feature reduction includes feature extraction and feature selection. Feature extraction refers to the process of finding a mapping that reduces the dimensionality of the patterns while feature selection refers to picking up a number of features to make up an at least suboptimal feature subset. In part due to the difficulty in interpreting the mapped features by feature extraction, the feature selection problem is emphasized instead as a critical preprocessing step.

Another problem arises for the classification task for high dimensionality, that is, the discrimination between classes becomes much more difficult. As mentioned above, this is due to the fact that, the assumption that enough training samples are available to accurately estimate the class statistics, is likely to fail for hyperspectral data because gathering enough training samples in practice is either difficult or expensive. Therefore how to improve classification performance remains one important task for data analysis in hyperspectral remote sensing. This is achieved in this dissertation by studying the idea of merely fusing the existing classification schemes rather than to develop new and sophisticated classification techniques.

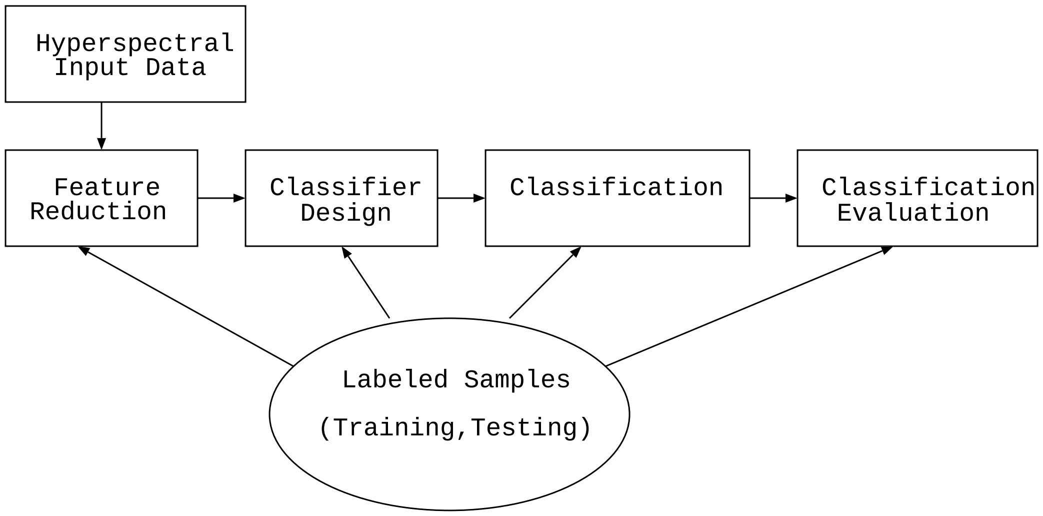


Figure 1.5: A typical supervised classification procedure for hyperspectral data analysis.

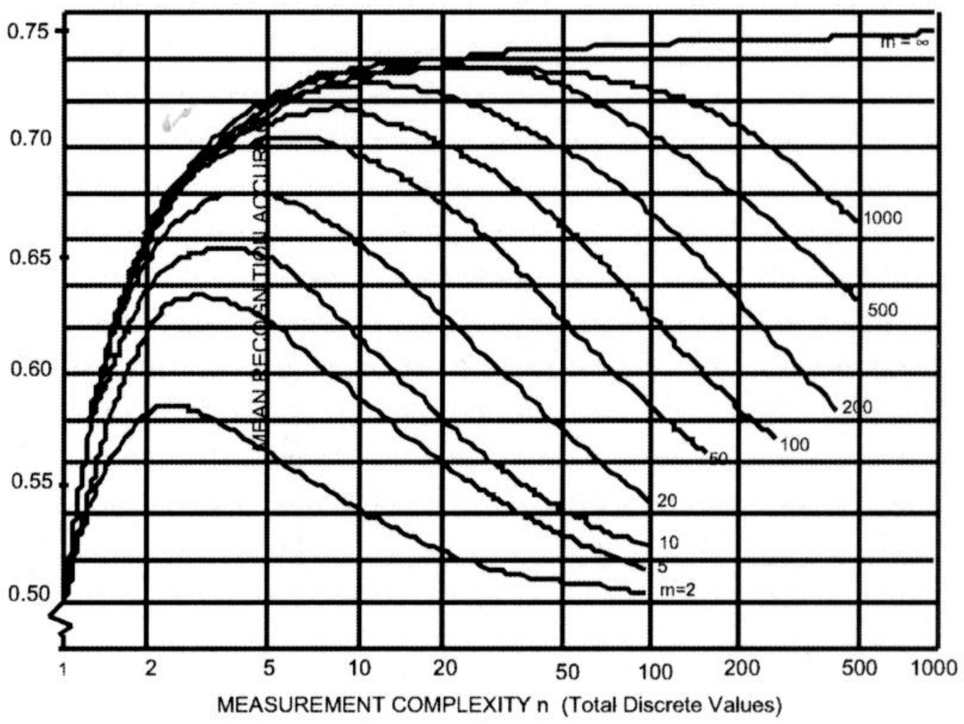
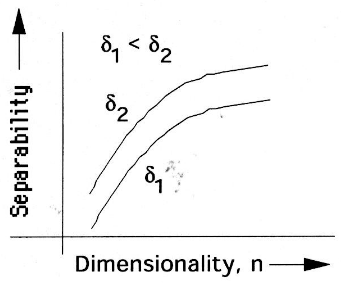
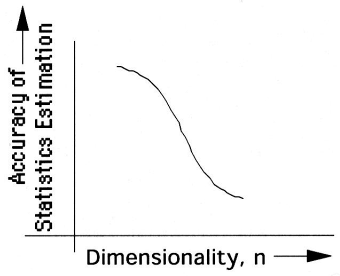


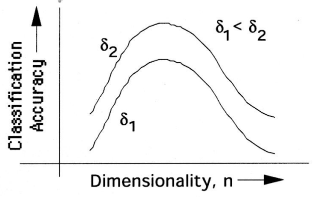
Figure 1.6: The Hughes phenomenon [139].



(a) High dimensionality potentially provides better class separability.



(b) The accuracy of statistics estimation decreases as dimensionality increases.



(c) The peaking phenomenon results from the combination of the two opposite effects as shown in (a) and (b).

Figure 1.7: The conceptual explanation of the Hughes phenomenon as shown in Figure 1.6.

1.3 Organization of the Dissertation

In the previous two sections, the background and motivation of the research conducted in this dissertation are explained. We present the remaining of the dissertation following the diagram of "past, present, and future". In "past", we generalize the primary frameworks of the research on both classifier ensembles and feature selection, which have been laid down by numerous researchers in this field; In "present", we present our studies on classifier ensembles (the nearest neighbor classifier ensembles) and feature selection (the genetic feature selectors); In "future", we propose some possible immediate further work.

The past, present and future diagram of the dissertation is illustrated in Figure 1.8.

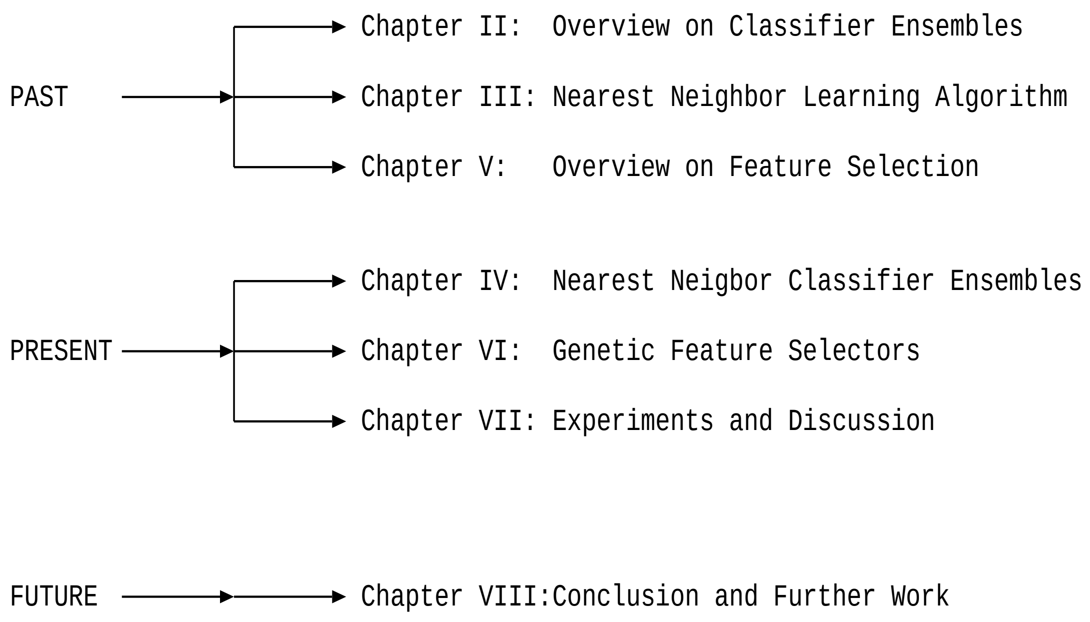

PAST
Chapter II: Overview on Classifier Ensembles
Chapter III: Nearest Neighbor Learning Algorithm
Chapter V: Overview on Feature Selection
PRESENT
Chapter IV: Nearest Neigbor Classifier Ensembles
Chapter VI: Genetic Feature Selectors
Chapter VII: Experiments and Discussion
FUTURE
Chapter VIII:Conclusion and Further Work

Figure 1.8: The Past, Present and Future diagram of the dissertation.

In Chapter II, an overview on classifier ensembles is given based on a categorization scheme. The nearest neighbor learning algorithm is revisited in Chapter III. Various methods for combining nearest neighbor learning algorithms are described in Chapter IV. The scheme of bias plus variance decomposition of error rate is then explained. With the goal of both further improving the classification performance and decreasing the storage requirements at the same time, a hybrid method which utilizes the Error Correcting Output Codes (ECOC) and the Condensed Nearest Neighbors (CNN) is then presented. Another variant which takes advantages of the randomly selected features in conjunction with ECOC is also suggested. In Chapter V, methods for feature selection are reviewed and described in a categorization scheme. A genetic feature selector which hybridizes genetic algorithms with the ensembled learning scheme is initiated in Chapter VI. In the mean time, a categorization scheme is done on the existing genetic feature selection methods. Experimental results of both nearest neighbor ensembles and genetic feature selector on a typical hyperspectral remote sensing data set and the corresponding discussion are given in Chapter VII. Finally, conclusion of this dissertation and possible future work are given in Chapter VIII.

CHAPTER II

Classifier Ensembles: An Overview

2.1 Introduction

It is the seminal work of Hansen and Salamon [126] at the very beginning of the 90's which recognized that the unstable nature of certain neural networks was helpful for ensembles that opened the door for the theoretical study on classifier ensembles. Since then, classifier ensembles have been under extensive theoretical as well as empirical studies. Classifier ensembles[1] [2] are one of the frontiers of pattern recognition [149], and they fall into, by and large, the paradigm of supervised learning.[3] In the literature, the design methodology for supervised learning was addressed [201], various supervised learning algorithms were reviewed in a categorization scheme [82] and their performance was systematically investigated [188].

On the other hand, the research on classical pattern recognition models, including feature selection and classifier ensembles, was challenged by the recent development of a novel approach called Support Vector Machines (SVMs) [270] [25]. The argument was that with the advanced design methodology of the SVMs, questions such as if

[1]Online classifier ensembles bibliography can be found in [7].

[2]In practice, it may sometimes be not feasible to collect all the data into one single flat file, due to reasons such as storage cost, bandwidth, or security, privacy, proprietary nature of the data [122]. Hence the ability to deal with these data in a distributed manner draws increasing attention. Indeed, the research on *distributed classification* [160] (as well as *distributed clustering* [159]) and the corresponding *distributed* ensemble methods pertains to another emerging, yet active direction. However, the distributed scheme is not discussed in this dissertation.

[3]There are semi-supervised learning ensembles that construct classification ensembles based on both labeled and unlabeled data, e.g. ASSEMEBL [58], an adaptive semi-supervised ensemble.

fusing multiple classifiers would become redundant or out of date were raised among researchers in the field. Kittler's lecture [164] assured us the rationale for continued need and interests in the research of classifier ensembles. Indeed, the recent years have seen a rapid progress on this research topic [231] [182] [279] [167] [210] [54] [126] [174] [91] [211]: not only do the SVMs not make the traditional pattern recognition model out of date, but also are the SVMs benefiting from the research on classifier ensembles [102].

2.2 The Hypothesis: Why Classifier Ensembles Work

The main discovery in the topic of combining classifiers is that the ensemble architecture can gain better accuracy than the individual component alone. The use of ensembles in machine learning is rather new, but the idea that aggregating the opinions of a committee of experts can obtain better accuracy is not new. Let's cite what the Condorcet Jury Theorem states:

"If each voter has a probability p of being correct and the probability of a majority of voters being correct is M, then $p > 0.5$ implies $M > p$. In the limit, M approaches 1, for all $p > 0.5$, as the number of voters approaches infinity."

Marquis Condorcet in 1784 proposed this theorem [88]. A more recent reference is due to Nitzan and Paroush [208].

The hypothesis, that the classifier ensemble is much accurate than any of its individual component classifiers if and only if the component classifiers are accurate and diverse, was first introduced by Hansen and Salamon [126]. They proved that if each of the individual classifiers is independent, and their error rates are all less than 50%, the error rate of the ensemble classifier will decrease with the number of individual classifiers. Two key factors are crucial: accurate classifiers and diversity

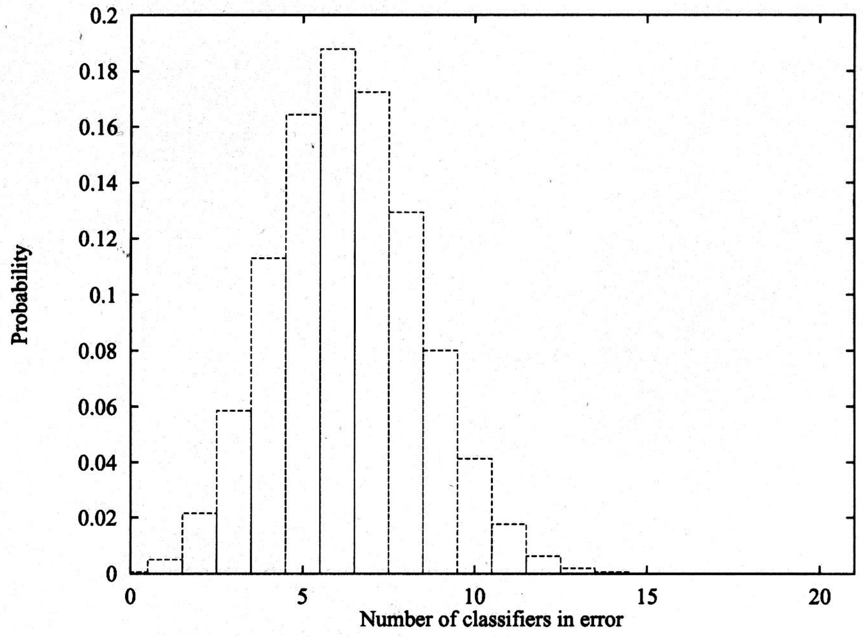


Figure 2.1: A simulation by Dietterich [98]

of the classifiers. An accurate classifier is defined as having a classification accuracy better than a random guess. Diverse classifiers make different predictions on new data points, i.e. they have different error rates.

A simulation made by Dietterich [98] is shown in Figure 2.1. Dietterich stated that if the error rates of all classifiers are equal and their values are less than 50%, furthermore, the error rates are assumed to be not correlated, then the probability that the majority vote is wrong is calculated as the area under the binomial distribution where more than half of the classifiers are wrong. In Figure 2.1, Dietterich simulated a classifier ensemble with 21 individual classifiers, each having an error rate of 30%. The area under the curve for 11 or more is 0.026. This is much less than the error rate given by the individual classifier (30%).

The relationship between the error rate of the classifier ensemble and the error rates of the individual classifiers was shown by Tumer and Ghosh [264] [265] [266] as follows:

$$E(ensemble) = \frac{1 + \rho(N-1)}{N} E(individual) + E(Bayes) \tag{2.1}$$

where N is the number of classifiers, ρ is the correlation among the classifier errors, E(Bayes) is the error rate obtained using the Bayes rule assuming that all the conditional probabilities are known. E(ensemble) and E(individual) represent the error rate of a classifier ensemble and the error rate of an individual component classifier respectively. $\rho = 0$ means the error of the whole ensemble decreases proportionally to the number of the component classifiers while $\rho = 1$ means the error of the ensemble architecture equals to the error of a single component classifier.

2.3 Categorization Scheme of Classifier Ensembles

Although the history of research on classifier ensembles is only about one decade, this area is becoming very active recently. In the literature, there exists very little effort to categorize the classifier ensembles. Generally speaking, the classifier ensembles can be divided into parallel ensembles and sequential ensembles [22]. The categorization scheme can also be done according to the combining strategies, e.g. the diversity of the classifier ensemble. Dietterich [99] describes the ensemble methods from the point of view of machine learning. Jain, Duin and Mao [149] list a number of popular ensemble methods in their review paper on statistical pattern recognition. Sharkey [244] points out that a limiting factor in reseach on combining classifiers is due to a lack of awareness of the full range of available modular structures. One reason for this is that there is as yet little agreement on a means of describing and classifying types of multiple classifier systems in the literature. He then presents a categorization scheme for types of multiple neural network systems based on four subdivisions:

(i) Involving competitive or cooperative combination mechanisms;

(ii) Combining either ensemble, modular, or hybrid components;

(iii) Relying on either bottom-up or top-down combination methods;

(iiii) When bottom up using either static or fixed combination methods.

In this chapter, we aspire to a general and comprehensive categorization scheme. We describe the classifier ensembles within the following 6 categories:

Section 2.3.1, Voting classifier ensembles;

Section 2.3.2, Classifier ensembles manipulating training samples;

Section 2.3.3, Classifier ensembles with different input feature subsets;

Section 2.3.4, Heterogeneous classifier ensembles;

Section 2.3.5, Homogeneous classifier ensembles;

Section 2.3.6, Recursive partition classifier ensembles.

2.3.1 Voting Classifier Ensembles

There are three primary categories in the voting ensemble scheme.

- Simple Voting:

  Simple voting, also called majority voting and select all majority (SAM) [53], considers each component classifier as an equally weighted vote. The classifier that has the largest amount of votes is chosen as the final classification scheme [53].

- Weighted Voting:

In weighted voting schemes, each vote receives a weight, which is usually proportional to the estimated generalization performance of the corresponding component classifier. Weighted voting schemes usually give better performance than simple voting [53].

- Weighted Majority:

  The weighted Majority algorithm [191] is actually a generalization of the HALVING [189] algorithm. It is similar to weighted voting, the main difference is how the weights are generated. It makes predictions by taking a weighted vote among a pool of classification algorithms and learns by altering the weight associated with each prediction algorithm. This algorithm starts with assigning a weight of 1 to each classification algorithm, then considers the training samples. If a classification algorithm misclassifies a new training sample, its weight is then decreased by multiplying it by some number β, where $0 \leq \beta < 1$.

  The number of mistakes made by the weighted majority algorithm can be bounded in terms of the number of mistakes made by the best classification algorithm in the voting pool. Suppose G is any set of n component classification algorithms and let r be the minimum number of mistakes made by any algorithm in G for the training set T. Littlestone and Warmuth [190] have generalized and showed that for any $0 \leq \beta < 1$, the bound for the number of mistakes is given by:

$$\frac{r log_2 \frac{1}{\beta} + log_2 n}{log_2 \frac{2}{1+\beta}} \tag{2.2}$$

If $\beta = \frac{1}{2}$, we obtain the following bound:

$$2.4(r + log_2 n) \tag{2.3}$$

2.3.2 Classifier Ensembles Manipulating Training Samples

The methods that fall into this category have the following characteristic: the learning algorithm is run several times, each time with a different partition of the training samples. It works well for unstable learning algorithms. By "unstable" we mean the learning algorithms whose output predictions have changes in response to a small change in the training samples.

Among all these methods, Boosting and Bagging are the two most successful and representative methods developed to date for classifier ensembles. Many researchers have compared boosting and bagging with other methods and demonstrated their superiority [53] [223] [100].

- Bagging:

  Bagging is due to Breiman [65]. This method is run several times on training samples, i.e. on each run, it produces a replication of the original training samples by sampling with replacement with the same size as the original training size. Some training samples appear in the produced samples while others may not. Such a training set is called a *bootstrap replicate* of the original training set, and this technique is called *B̲ootstrap Agg̲regating̲*, from which the name of *Bagging* stems. For a training set with m examples, the probability of an example being reproduced is given by:

$$1 - (1 - \frac{1}{m})^m \tag{2.4}$$

```
Input:
    L: a learning algorithm
    N: an integer
        For $i = 1$ to $N$
            $\bar{T}$ = bootstrap sample from training set $T$
            $h_i = L(\bar{T})$
        End For
Output:
        $h_f(\mathbf{x}) = \underset{y \in \mathbf{Y}}{\operatorname{argmax}} \sum_{i}^{N} h_i(\mathbf{x}) = y$
```

Table 2.1: The Bagging Algorithm.

where for a large enough m, i.e. mathematically speaking, when $m \longrightarrow \infty$, this approximates to $1 - \frac{1}{e}$. This value is about 63.2%. So each bootstrap reproduces, on the average, 63.2% of the original training samples. All the individual classifiers are then used to classify each example in the test set, usually a vote scheme is taken.

The pseudocode for the bagging algorithm is shown in Table 2.1.

But how many bootstrap replicates are sufficient? Breiman states that: "more replicates are required with an increasing number of classes [66]." He also notes that "bagging is almost a dream procedure for parallel computing." In a simulation experiment varying the number of bootstraps, it was verified that 10 is usually sufficient.

- Boosting:

Boosting was first proposed by Schapire [237]. It was proven that any weak learning algorithm may be *boosted* to a strong one based on a theoretical model known as the weak learning model (PAC) [269]. This PAC learning model (Probably Approximately Correct) assumes that there exist weak learning algorithms which can do slightly better than random guessing regardless of the underlying probability distribution used when generating the examples.

The most well known boosting idea, AdaBoost, was introduced by Freund and Schapire [115]. The term AdaBoost stems from *Adaptive Boosting*. It solved many of the practical difficulties of the earlier boosting algorithms. Just like bagging, AdaBoost also manipulates the training examples to generate diverse hypotheses. The pseudocode for AdaBoost is illustrated in Table 2.2. It maintains a probability distribution $p_n(\mathbf{x})$ over the training samples. In each iteration n, it draws a training set of size m by sampling with replacement according to the probability distribution $p_n(\mathbf{x})$. The learning algorithm is then applied to produce a classifier h_n. The error rate e_n of this classifier on the training samples is computed and used to adjust the probability distribution on the training samples. AdaBoost was developed originally for two-class problems, but many methods have been developed for handling multi-class problems.

Kuncheva and Whitaker [181] described three variants of AdaBoost, i.e. Aggressive boosting, Conservative boosting and Inverse boosting. Since the first paper on boosting by Schapire, boosting alone has been an active research direction, for the up-to-date research trends on boosting, consult the website given in [16].

- Cross-Validated Committees:

  Cross-Validated Committees construct the training sets by leaving out disjoint subsets of the training data [98] [130]. For example, the training set can be randomly divided into 10 disjoint subsets. Then 10 overlapping sets can be constructed by dropping out a different one of these 10 subsets. The same procedure is employed to construct training sets for 10-fold cross validation. That is why this method is called *cross validated committees* [212].

```
Input:
        $L$: a learning algorithm
        $N$: an integer
[1] Initialize for all $i : w_1(i) = \frac{1}{m}$                     Initial weights
[2] *For* $n = 1$ to $N$ do
[3]     For all $i : p_n(i) = w_n(i)/(\sum_i w_n(i))$                 Normalize the weights
[4]     $h_n = L(p_n)$
[5]     $e_n = \sum_i p_n(i)\delta[h_n(\mathbf{x}_i) \neq y_i]$       Calculate the error for $h_n$
[6]     if $e_n > 1/2$ then
[7]         $N = n - 1$
[8]         goto 12
[9]     $\beta_n = e_n/(1 - e_n)$
[10]    For all $i : w_{n+1}(i) = w_n(i)\beta_n^{1-\delta[h_n(x_i)\neq y_i]}$    Calculate new weights
[11] *End for*
[12] Output:
    $h_f(\mathbf{x}) = \underset{y\in\mathbf{Y}}{\text{argmax}} \sum_{n=1}^{N} (log\frac{1}{\beta_n})\delta[h_n(\mathbf{x}) = y]$
```

Table 2.2: The AdaBoost Algorithm. The value of $\delta[Q]$ equals to 1, if Q is true, and 0 otherwise.

2.3.3 Classifier Ensembles with Different Input Feature Subsets

This is a general method where different feature subsets are taken and passed on to different classifiers. For example, Cherkauer [81] trained a neural network classifier ensemble which consists of 32 component neural networks to identify volcanoes on Venus. These neural networks are trained on 8 different subsets of the 119 available input features and 4 different network sizes. The input feature subsets were selected manually to group features that were based on different image processing operations. By doing this, they were able to match the performance of human experts in identifying volcanoes.

Combining classifiers with different features was also studied by Chen, Wang and Chi [79] with emphasis on text-independent speaker identification. They did a systematic investigation and classified into three frameworks, i.e. linear opinion pools, winner-take-all, and evidential reasoning. In the framework of linear opinion pools, the combination schemes make the final decision through the use of a linear combination of the predictions of the multiple classifiers. In the framework of winner-take-all, it chooses the best classifier which can be viewed as a winner. In the framework of evidential reasoning, for an input pattern, the output of each individual classifier is regarded as an evidence or an event and the combination scheme makes the final decision based on a method of evidential reasoning. Evidential reasoning is a methodology based on the Dempster-Shafer calculus of evidence [242].

2.3.4 Heterogeneous Classifier Ensembles

Stacked generalization and meta learning are two representative methods which fall into the category of heterogeneous classifier ensembles where different types of classifiers make up an ensemble architecture. A comparison study on combining heterogeneous sets of classifiers was done by Bahler and Navarro [51]

- Stacked Generalization:

  The general framework of stacked generalization was addressed by Wolpert [278]. It is a scheme for minimizing the generalization error rate of one or two generalizers. Stacked generalization is a layered architecture, where the classifiers at the lever-0 (bottom) layer receive the original data as input and each of the classifiers outputs a prediction. Successive layers receive the predictions of its immediate preceding layer as the input. The output is passed to the next layer.

  Most works deal with two-layer architectures [251] [262] [67]. For a two-layer (level-0 and level-1) architecture, it works as follows:

  (1) Train each of the level-0 classifiers using the following leave-one-out cross-validation. For each example in the training set, leave-one-out and train on the

remaining samples. After training, classify the left-out examples. Form a vector from the predictions of all level-0 classifiers and the actual class of that example.

(2) Train the level-1 classifier using the collection of vectors generated in the previous step as the training set. The number of examples in the level-1 data is equal to the number of examples in the training set.

(3) In step one, classifiers are generated using a leave-one-out method. To fully explore the training set, all level-0 classifiers are re-trained using the entire training set. The generated models are then used to classify the examples in the test set.

Of course, stacked generalization is not constrained to two-layers, it can be generalized to multi-layer architectures.

- Meta Learning:

  The survey paper on meta learning by Vilata and Drissi [274] shows the term meta learning has been ascribed different meanings by different research groups. Two methods based on meta learning were introduced by Chan and Stolfo [78], i.e. arbiter and combiner. Both schemes are to meta-learn a set of meta-classifiers whose training data are based on predictions of a set of base classifiers. An arbiter is learned by some learning algorithm to arbitrate among predictions generated by different base classifiers. The aim of the combiner strategy is to coalesce the predictions from the base classifiers by learning the relationship between these predictions and the correct prediction.

2.3.5 Homogeneous Classifier Ensembles

While research continues on the general classifier ensemble algorithms, efforts have been made by researchers to combine only one specific classifier in a classifier ensemble. Zheng [288] studied naive bayesian classifier ensembles. Various neural network ensembles were addressed and investigated systematically in [243]. We present a comprehensive study on nearest neighbor classifier ensembles in chapter IV where all the ensembles contain only one type of classifier: the nearest neighbor classifier.

2.3.6 Recursive Partition Classifier Ensembles

Recursive Partitioning algorithms for classifier ensembles use a divide-and-conquer strategy to partition a space into regions that contain instances of only one class. Utgoff and Brodley provide scheme examples for recursive partition ensemble algorithms [268] [70]. Utgoff's work on perception tree algorithms combines a univariate decision tree with linear threshold units. It first determines if a subspace is linearly separable by using a heuristic measure. If the subspaces are linearly separable, then a linear threshold unit is applied. If not, the space is divided using an information theoretic measure. Brodley's model class selection system creates a recursive, tree-structured hybrid classifier which combines decision trees, linear discriminant functions and instance-based classifiers.

In my opinion, the recursive partition method for classifier ensembles is getting less attention recently from the classifier ensemble community, but the idea of recursive partition itself has found a lot of real world applications (e.g. [84]).

For the online bibliography reference on research about recursive partition (but not limited to classifier ensembles), see [32].

2.4 Summary

In this chapter, we described the methods for classifier ensembles in a categorization scheme. Three basic criteria are usually used to evaluate a classifier ensemble [251], i.e. accuracy, efficiency and diversity, among which the accuracy is on the top priority of consideration. However, we argue that the tradeoff between the error rate and the bias-variance should be also taken into account in practical applications. That is, when considering a specific classifier, an often adopted strategy is to choose the classification algorithm which has a small error rate, and ideally, at the same time it also has both small bias and small variance. For example, if a classification algorithm has a low error rate, but it has a high bias and a high variance, then one should be cautious in this case, since high bias means this algorithm has a high systematic error, and high variance indicates this algorithm has poor generalization. For details about the scheme of bias plus variance decomposition of error rate, see chapter IV. In Chapter VII, we conduct the experiments on various nearest neighbor ensembles while taking into account this bias plus variance decomposition scheme.

CHAPTER III

Nearest Neighbor Learning Algorithm Revisited

3.1 Introduction

The nearest neighbor learning paradigm has been the central subject of many theoretical and experimental studies for over half a century. It is one of the oldest and simplest methods for performing non-parametric classification, where the class label of an input pattern is assigned based on the class labels represented by the k closest neighbors of the training set. Despite its simplicity, it has many advantages, e.g. it does not require any knowledge about statistical properties of the data beforehand [105], it may give competitive performance compared to many other methods.

The basic concept underlying the nearest neighbor classifier was first introduced by Fix and Hodges [110]. In 1967, Cover and Hart [90] formally defined the nearest neighbor rule and applied it to the pattern recognition problem. Since then, the nearest neighbor algorithm has been under extensive study [92] [94] [280] [250] [60].

An easy and effective way to calculate the classification error rate is by the "leave-one-out" method. Hereby, each time the complete training set, but one, is used, and the left out training sample is used for testing. By doing this for each training sample separately, the classification error rate can be evaluated.

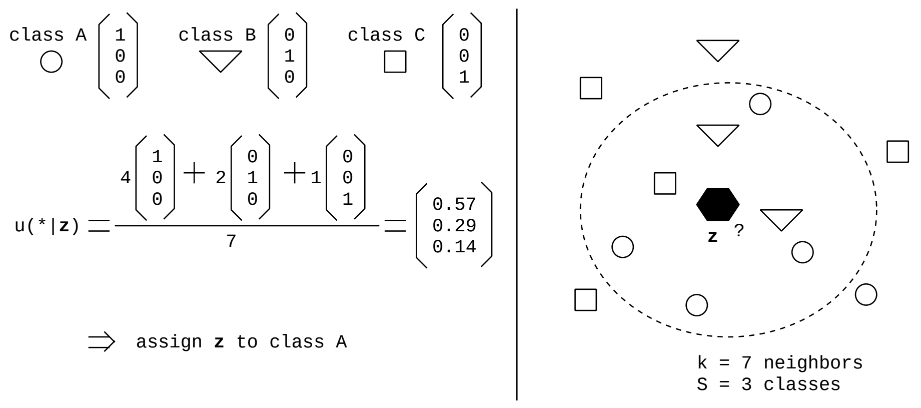


Figure 3.1: A geometric illustration of the crisp kNN classification rule. The label vector represents the absence (by a 0) or presence (by a 1) of a label.

3.2 Nearest Neighbor Classifier

Suppose $\mathbf{z}$ is an unlabeled vector, $\mathbf{x}_i$ is the ith labeled vector, k is the number of nearest neighbors to find in the neighborhood of $\mathbf{z}$, and there exists some distance measure between $\mathbf{z}$ and $\mathbf{x}_i$ in $\mathcal{R}^p$. The geometric illustration of the crisp kNN classification rule is give in Figure 3.1.

There are four algorithmic parameters associated with the kNN rule: (1) the value of k; (2) the choice of distance measure; (3) the distance weighting measure (weighted or nonweighted); (4) the method of counting votes.

Many distance metrics have been proposed in the literature, for example, the Chi-square metric [277], the Manhalanobis metric [108], the Cosine Similarity metric[1] [234], the Quadratic metric [117], the Modified Value Difference metric [89], of which

[1]The Cosine Similarity $CS_{\mathbf{xy}}$ is the cosine of the angle α between two L-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$, i.e.

$$CS_{\mathbf{xy}} = \cos\alpha = \frac{\sum\limits_{i=1}^{L} x_i y_i}{\sqrt{\sum\limits_{i=1}^{L} x_i^2 \sum\limits_{j=1}^{L} y_j^2}}$$

The more similar the two objects are, the closer the value of $\cos\alpha$ approaches to 1, hence the angle between them becomes close to 0. On the other hand, the Spectral Angle Mapper (SAM) [175] is a somewhat popular classifier using laboratory spectra to determine the similarity between two spectra by calculating the "spectral angle" between them. This algorithm, implemented also in the ENVI software package [33], actually shares the same idea as the Cosine Similarity metric, though it has the name of SAM in the remote sensing community: it's in fact similar to the nearest neighbor rule (1-NN) with the cosine similarity metric as the distance measure. So probably it's better to make a tunable parameter k in the SAM algorithm, so that one can make a choice at one's own needs.

the Euclidean distance is the most commonly used. How to select the distance metrics was studied by Barker [52]. Although the Euclidean distance assumes that the variables are uncorrelated [108], this might be not justified for hyperspectral data, still we use it in the dissertation, since it is the most commonly used. There are many variants of kNN, 51 milestone articles on the kNN rule and its variants were given in [92]. In the machine learning community, kNN is often called *instance-based learning* [44] or *memory-based learning* [254] where responses are computed by interpolating from a table of stored patterns.[2]

One of the drawbacks of nonparametric methods is that they require a large amount of computation time. In the case of k nearest neighbor classification, this is due to the fact that it must compute every time the distance of an input pattern with all training sample patterns in order to find the k nearest neighbors.[3] Many methods have been proposed to reduce the computation time in the literature [119] [224] [77].

3.3 Fuzzy Nearest Neighbor Classifier

One of the difficulties for kNN is that each of the labeled samples is given equal importance in deciding the class memberships of the patterns to be classified, regardless of their "typicalness". On the other hand, a problem arises for classification in high dimensionality, that is, the discrimination between classes becomes much

[2]The terms of instance-based and memory-based learning are just the synonymous names of lazy learning [42]. Lazy learning subsumes a family of algorithms that store the complete set of given examples and delay all further calcalations, until requests for classifying yet unseen instances are received. Other synonymous names of lazy learning include exemplar-based, cased -based and experience-based. As opposed to such "lazy" learners, the "eager" learners, such as artificial neural networks, are algorithms where training examples are complied into a model at training time but not available at runtime.

[3]In the field of computational geometry, this is referred as *nearest neighbor search* [143] (or similarity search [235]), i.e. given a database of points in a multidimensional space, construct a data structure which, given any query point, finds the database point(s) closest to it. In practice, when dimensionality becomes much higher and higher, computing *exact* nearest neighbor is a very difficult task. Few algorithms seem to be significantly better than a brute-force computation of all distances. In this case, the *approximate* nearest neighbor is sought instead of the *exact* nearest neighbor, but with a trade-off between accuracy and time complexity. The nearest neighbor search problem is a key issue in computational geometry, and has been of great importance in many areas of computer science, including pattern recognition, databases ([133]), vector compression, computational statistics and data mining [142].

more difficult. This is due to the fact that, the number of training samples needed to catch up with the increasing dimensionality grows overwhelmingly. In pattern recognition, crisp classification is often replaced by fuzzy classification [59]. In these techniques, the decision to classify a data point is delayed as long as possible by the use of memberships. Membership values are assigned to the point as a function of the point's distance from its k nearest neighbors and those neighbors' memberships in the possible classes. These techniques have proven to outperform the crisp classification techniques, especially when clusters tend to overlap. The theory of fuzzy sets has been introduced into the k nearest neighbor classification. The fuzzy kNN rule was given in [157]. A fuzzy kNN classifier was designed by Keller *et al.* [161], where class memberships are assigned to the sample, as a function of the sample's distance from its k nearest neighboring training samples. The fuzzy kNN procedure is described below:

1. Store training data T with their S partitions.
2. Choose k = number of neighbors to find.
3. Choose d: $\mathcal{R}^p \times \mathcal{R}^p \to \mathcal{R}^+$ any distance metric on $\mathcal{R}^p$.
4. For any vector $\mathbf{z} \notin T$, using $T = \{\mathbf{x}_i\}$, compute and rank-order the distances $d(\mathbf{z}, \mathbf{x}_i)$ as $\{d_1 \leq d_2 \leq \cdots \leq d_k \leq d_{k+1} \leq \dots\}$.

   Calculate

$$u_i(\mathbf{z}) = \frac{\sum_{j=1}^{k} u_{ij} \left(\frac{1}{d(\mathbf{z},\mathbf{x}_j)^{\frac{2}{w-1}}} \right)}{\sum_{j=1}^{k} \left(\frac{1}{d(\mathbf{z},\mathbf{x}_j)^{\frac{2}{w-1}}} \right)} \tag{3.1}$$

where $u_i(\mathbf{z})$ is the assigned membership of the vector $\mathbf{z}$, u_{ij} is the membership in the ith class of the jth vector of the labeled sample set, and w is a scaling parameter between 1 and 2. If Euclidean distance is used, then $d(\mathbf{z}, \mathbf{x}_j) = ||\mathbf{z} - \mathbf{x}_j||$ The memberships of the training samples u_{ij} can be defined in several ways. The 'crispest' way is to give them complete membership in their own class and nonmembership in all other classes. A more 'fuzzy' alternative is to assign the training samples' memberships based on the distance from their class mean. After calculating the memberships for the test sample, it is assigned to the class with highest membership. In our experiments we have found that the second approach leads to the best results.

CHAPTER IV

Ensemble Methods for Nearest Neighbor Learning Algorithm

4.1 Introduction[1]

Integration of the predictions of a number of classifiers has been shown to be an effective way to achieve more accurate classification than any of the component classifiers,[2] and promising results have been given in many real world applications, e.g. handwritten character recognition [281] [135] [120], protein structure prediction [286], calculation of fat content of ground meat [260]. There are many general algorithms for combining classifiers such as Bagging [65] and Boosting [16]. This area is one of the four most important directions in machine learning research [98], and has many different names [180], e.g. classifier fusion, classifier ensembles, censensus aggregation.

In contrast to the huge amount of research in this active area [231] [182] [279] [167] [210] [54] [126] [174] [211], little work has been done on combining the specific classifier: the k nearest neighbor classifier (kNN) [105] [90]. Since its introduction, the kNN rule has been well studied and improved [118], but the ensemble methods for kNN classifiers are limited in the literature [251] [47] [55] [228].

[1]This chapter, together with part of Chapter VII, is the generalized work of [285].

[2]The problem of combining preference arises in many applications, such as combining the results of different search engines [145].

The purpose of this chapter is two-fold: (i) a comprehensive description of the nearest neighbor classifier ensembles is performed (with the experimental study on hyperspectral remote sensing data in Chapter VII). (ii) a method of CNN-ECOC which combines Condensed Nearest Neighbors (CNN) in conjunction with Error Correcting Output Codes (ECOC) is proposed in Section 4.3.5. Another variant method, kNN-ECOC-RS, which utilizes the Randomly Selected features with Error Correcting Output Codes is suggested as a by-product in Section 4.3.4.

This chapter is organized as follows: In the next section, the technique of Error Correcting Output Codes (ECOC) is briefly explained. In section 4.3, different kinds of nearest neighbor classifier ensemble methods are recapitulated and the extended methods are then presented. A taxonomy of classification methods using the k-NN algorithm is also given. In section 4.4, the scheme of bias plus variance analysis of the error rate is explained.

4.2 Error Correcting Output Codes (ECOC)

Error Correcting Output Codes (ECOC) are one kind of *distributed* output representations, and were first proposed for multi-class classification tasks by Dietterich and Bakiri in their seminal work of [101], where classifiers are combined for multi-class problems by decomposing into multiple two-class distribution classifiers. Each class is assigned a binary code word and each component classifier is assigned the task of learning one-bit position of that code word. The resulting predictions of the component classifiers then form a vector, representing the separation of the classes into two disjoint subsets. A Hamming distance measure is used to compute the closest codebook vector to the vector of predictions.

Table 4.1 shows an example of typically pre-defined ECOC codes for a 5-class

Class	*Code Word*														
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Corn/min	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
Grass/pasture	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
Grass/trees	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*
Soy-clean	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*
Wheat	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*

Table 4.1: An example of typical 15-bit ECOC codes for a 5-class problem.

problem. Suppose we have a 5-class classification problem, and the description of the 5 classes is given in Table 7.1 of Chapter VII. To classify a new object, the 15-bits are evaluated to obtain a 15-bit binary string, say, 111010011101001. Then the Hamming distance (which counts the number of bits that differ with each other) of this string to each of the pre-defined 5 codewords is calculated. The nearest codeword is 111111111111111, and this corresponds to class Corn/min. Hence, the new object is assigned to the Corn/min class.

4.3 Ensemble Methods for Nearest Neighbor Classifiers

In this section, we describe several existing methods [251] [47] [55] [43] for combining nearest neighbor classifiers, and present several extended methods. All the ensemble methods belong to the category of homogeneous classifier ensembles as described in Section 2.3.5, which contain only the nearest neighbor classifiers as component classifiers.

A taxonomy of classification methods presented in this chapter using the k-NN algorithm is illustrated in Figure 4.1. We first divide the methods using k-NN algorithm into ensembles where k-NN is used as component classifier and non-ensembles where only a single k-NN (or its variant) is used. We then divided the k-NN ensembles into three different categories: Voting ensembles; Ensembles with different input feature subsets; Ensembles manipulating training samples. Voting over multiple condensed NN (section 4.3.2) belongs to the voting ensembles. The ensembles

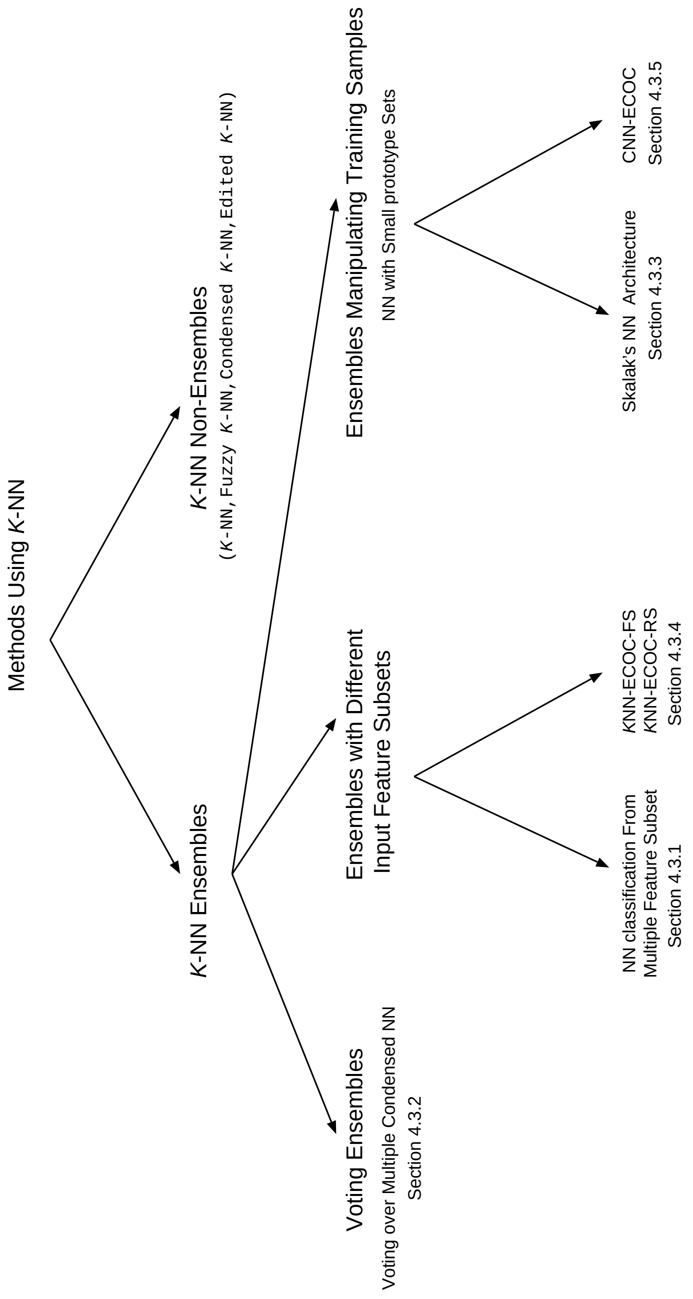

Methods Using K-NN
K-NN Non-Ensembles
(K-NN, Fuzzy K-NN, Condensed K-NN, Edited K-NN)
K-NN Ensembles
Voting Ensembles
Voting over Multiple Condensed NN
Section 4.3.2
Ensembles with Different Input Feature Subsets
NN classification From Multiple Feature Subset
Section 4.3.1
KNN-ECOC-FS
KNN-ECOC-RS
Section 4.3.4
Ensembles Manipulating Training Samples
NN with Small prototype Sets
Skalak's NN Architecture
Section 4.3.3
CNN-ECOC
Section 4.3.5

Figure 4.1: A taxonomy of classification methods using k-NN algorithm.

with different input feature subsets consist of three methods: the NN classification from multiple feature subsets (section 4.3.1), kNN-ECOC-Feature Selection method: kNN-ECOC-FS, and kNN-ECOC-Randomly Selected feature method: kNN-ECOC-RS (section 4.3.4). Skalak's NN Architecture (section 4.3.3) and Condensed Nearest Neighbors (CNN) with Error Correcting Output Codes: CNN-ECOC (Section 4.3.5) are two methods which belong to the ensembles manipulating training samples.

4.3.1 Nearest Neighbor Classification from Multiple Feature Subsets (MFS)

Bay [55] proposed an algorithm: Nearest neighbor classification from multiple feature subsets (MFS). It uses a simple voting scheme, and just takes the output which has the highest accuracy among the output of a number of component NN classifiers. Each of the component NN classifiers has the same number of features, and all the feature subsets are chosen by sampling from the original feature space. Two sampling methods, i.e. sampling with replacement and sampling without replacement, are used.

4.3.2 Voting over Multiple Condensed Nearest Neighbors (CNN)

When the number of the training patterns is too large, the need to store the whole patterns requires a large memory. Hart's [127] condensing algorithm solved this problem by storing only a subset of the full training set. This algorithm works as follows. First it starts with two subsets: the Grabbag subset and the Store subset. The Store subset is empty while the Grabbag subset contains the whole training set. Then the Store subset is initialized with a sample taken randomly from Grabbag. On each iteration, randomly take one sample from Grabbag, if it can not be correctly classified by the current samples in Store, then place it in subset Store, otherwise throw it back in subset Grabbage. The procedure is often referred to as *condensing.*

This method is actually a local search[3] in that the resulting subset Store depends on the order of the samples stored. If one shuffles the training samples, one may get different results. One may want to stop this procedure after several iterations in order to speed up the process, for example, when the stored patterns in the last iteration are less than, say 10%, of the training set. The final patterns stored in the Store subset are the condensed patterns, which can be used as training patterns. The classification rule used in this procedure is 1-NN, if the k-NN rule is used (i.e. $k > 2$), then the computational cost is becoming higher.

Alpaydin [47] proposed to train multiple condensed nearest neighbor subsets [127] and take a vote over them. Two voting schemes are used: simple voting where voters have equal weights and weighted voting where weights depend on the classifiers' confidence in their predictions. A simple method [47] is used to calculate the weights: taking the two most nearest neighbors of $\mathbf{x}$, say, $\mathbf{z}_1$ is the closest pattern, and $\mathbf{z}_2$ is the second closest one, we have weight α:

$$\alpha = \begin{cases} 1 & : \text{ if class label } h(\mathbf{z}_1) = h(\mathbf{z}_2) \\ \frac{d(\mathbf{x},\mathbf{z}_2) - d(\mathbf{x},\mathbf{z}_1)}{d(\mathbf{x},\mathbf{z}_2)} & : \text{ otherwise} \end{cases}$$

where d is the distance measure.

Besides, another way is the union method, which combines the multiple CNN's by applying NN to the union of subsets obtained by multiple CNN's.

4.3.3 Nearest Neighbor Classifiers with Small Prototype Sets: Skalak's NN Architecture

We follow Skalak's [251] discussion on composite nearest neighbor classifiers. Of the many architectures for classifier combinations till now, there are 3 primary architectures for combining classification algorithms: 1. *Stacked Generalization* (Figure

[3]This local search strategy is also used by both IB2 [44] and Grow and Learn (GAL) learning scheme [46].

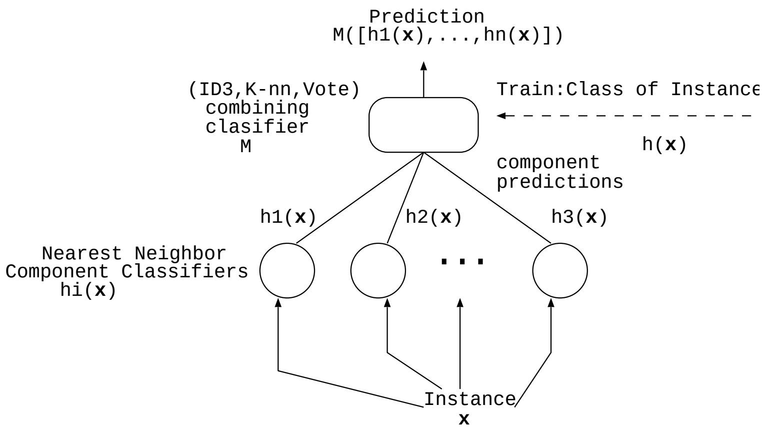


Figure 4.2: Stacked nearest neighbor classifier architecture.

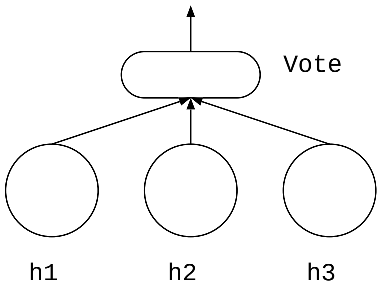


Figure 4.3: Boosting architecture. Classifier h1 is taken as given, while h2 and h3 are additional component classifiers.

4.2); 2. *Boosting* (Figure 4.3); 3. *Recursive Partitioning* (Figure 4.4).

Wolpert [278] was probably the first to discuss the idea of Stacked Generalization in its full generality. Stacked Generalization assumes that a set of n level-0 (component) learning algorithms, a level-1 learning (combining) algorithm, and a training set of classified instances have been given. It is a recursive layered structure for combining classifiers, where at each layer the classifiers are used to combine the output of the classifiers just under that layer. Boosting is due to Schapire [237]. The goal of boosting is to increase the accuracy of a given algorithm on a given distribution of training instances. It successively creates complementary component classifiers by filtering the training set. Recursive Partitioning algorithms use a divide-and-conquer strategy to partition a space into regions that contain instances of only one class.

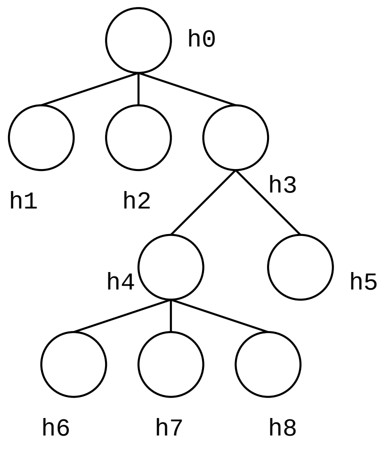


Figure 4.4: An example of a recursive partitioning architecture. Each of the component classifiers, from h0 to h8, applies only to a particular region of the instance space.

Figure 4.5 is one of the composite architectures which were studied by Skalak [251]. It's a two-layer architecture, consisting of level-0 and level-1 classifiers. The level-0 classifiers consist of two classifiers: a base classifier (say h_0), i.e. a full nearest neighbor classifier, which uses all instances as prototypes, and a complementary classifier (say h_1), which is a minimal nearest neighbor classifier, storing only one prototype per class. The complementary nearest neighbor classifier h_1 is obtained through the following procedure:

(1) Randomly sample n sets of S instances (with replacement) from the training set T, where S is the number of classes exposed in T, one instance is drawn from each class.

(2) Use each set as a prototype set to construct a nearest neighbor classifier.

(3) Classify all instances in T using each of these n classifiers.

(4) Choose the classifier with highest classification accuracy on T as the complementary classifier h_1.

For the level-1 combining algorithm, the decision tree algorithm ID3 [221] is used. For each original training instance $\mathbf{x} \in T$ with class S_i, a level-1 training instance is

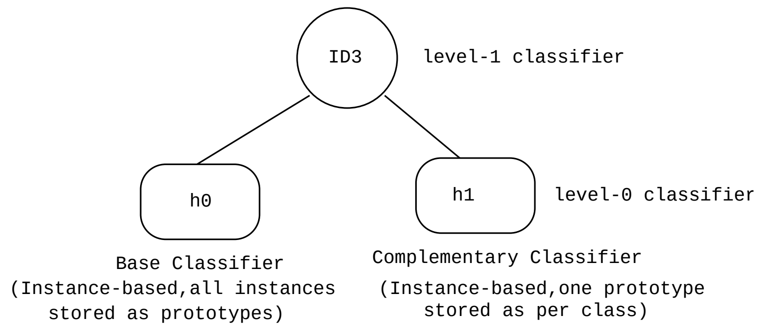

ID3
level-1 classifier
h0
h1
level-0 classifier
Base Classifier
(Instance-based,all instances stored as prototypes)
Complementary Classifier
(Instance-based,one prototype stored as per class)

Figure 4.5: A composite architecture.

created: $(h_0(\mathbf{x}), h_1(\mathbf{x}), S_i)$. So, for example, let $\mathbf{x}$ be a level-0 instance, if h_0 predicts class A when applied to instance $\mathbf{x}$, while h_1 predicts class B when applied to instance $\mathbf{x}$, then the level-1 feature representation for $\mathbf{x}$ becomes (A, B). The *entire* level-1 representation for $\mathbf{x}$ also includes the class of $\mathbf{x}$ (say A), i.e. the level-1 representation is actually (A, B, A). The set of these three-tuples of class samples is the training set used to train the level-1 learning algorithm ID3.

ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This procedure continues until the tree perfectly classifies the training examples, or until all attributes have been used. Although ID3 is no longer considered as a state-of-the-art decision tree, we use it as the level-1 combining algorithm. All the level-1 features are symbolic, the implementation of ID3 uses the same feature selection metric as described by decision tree C4.5 [222], a descendant of ID3.

In Dietterich and Bakiri's work [101], they showed that ECOC can improve decision trees and neural networks. We are motivated by this, and hypothesize that using ECOC in combination with the structure shown in Figure 4.5 can probably further improve the classification gains. The experimental results on this will be explained in Section 7.2.2 of Chapter VII.

4.3.4 Nearest Neighbor Classifier with Error Correcting Output Codes and Feature Selection: kNN-ECOC-FS

While neither Bagging nor (*directly*) combining the Error Correcting Output Codes (ECOC) with kNN improves the classification performance, Aha and Bankert [43] proposed to further combine ECOC with feature selection for each output bit and their empirical study on the cloud data showed performance gains. The procedures are as follows:

(1) Create the codewords for the specific problem. In this step, first the confusion matrix is obtained using the IB1 classifier [44] on the training set. Next, create the output bits by building a set of partitions, one per output bit, repeat until the set of partitions distinguishes each pair of classes according to at least the requested Hamming distance.

(2) Then kNN-ECOC computes the set of features to use when predicting bit values for each output bit (i.e. featue selection for each bit).

(3) For each test sample, kNN-ECOC predicts a value for each output bit and compares the predicted output string with each codeword, yielding the most similar codeword's class as its prediction for the test sample's class. The classification accuracy is then calculated.

Another probably interesting technique is the following. The kNN-ECOC idea combined with a feature selection method improves the classification gains, where feature selection was used for each bit. This is a considerable computational cost. But by combining with a randomly chosen feature subset for each bit instead of feature selection could act in the same way as feature selection: distances are computed differently for each bit, but this procedure will save considerable computational time. We call this variant as kNN-ECOC-RS in this dissertation. While with a random sampling of the original feature space, one can naturally only by chance get the classification performance improved. By using ECOC in conjunction with a random sampling, errors occurring by chance in each bit of the ECOC codes can be smoothly alleviated, as a consequence, the entire performance can be improved.

4.3.5 Condensed Nearest Neighbors (CNN) with Error Correcting Output Codes: CNN-ECOC

The ”nonlocal” learning algorithms (i.e. those that induce compact classifiers), e.g. the decision tree C4.5 and the neural networks trained by backpropagation, benefit from the use of ECOC, but the local ones (i.e. those that generate predictions based on information near the query samples such as the nearest neighbor algorithm), do not [171]. The reason is that, when using only local information, the bias errors in different output bits will be correlated, which will prevent the ECOC from reducing bias errors. Aha and Bankert [43] solved this problem by combining ECOC with a feature selection technique. On the other hand, the method of voting on Multiple Condensed Nearest Neighbors (CNN) generates different CNN's by shuffling the training samples on each run. Therefore this leads us to propose to combine the ECOC with Multiple CNN's: it relies on running CNN for each output bit, so that similarity can be computed on different resulting CNN's for each bit. This will, in fact, cause different stored samples to be retrieved for different output bits, and their predictions should be not correlated with each other. This follows the same strategy as the seminal work by Aha and Bankert [43]: while still depending on using local information during classification predictions, it uses different, yet local information for each output bit.

We have found that rather than trying to create a well-designed codeword in step (1) of Section 4.3.4, it would be much more convenient to use the large sets of codewords for ECOC already designed by Dietterich's group, which can be downloaded from [1]. In our experiments in Chapter VII, we used their pre-designed codes.

4.4 Bias plus Variance Decomposition of the Error Rate

It is unfair to evaluate the classification performance by the error rate alone, without considering the bias and variance effects of the classification algorithm. For example, if a classification algorithm has an acceptable classification performance, but also has a high variance, one should be cautious about it, since the high variance suggests that this algorithm has a poor generalization. The bias plus variance decomposition [121] is a powerful tool for explaining how changes to a potential algorithm can affect the resulting error rates. Researchers have proposed a number of decomposition methods in the literature [66] [150] [169] [171] [261]. The basic idea behind this theoretical framework is that a classification algorithm has two kinds of errors: (i) a systematic error, which is due to the representation languages used by the learning algorithm itself. (ii) an error resulting from random variation and noise in the training set and from any random behavior of the learning algorithm, i.e. this error depends on the generated model from the training set.

We follow Kohavi and Wolpert's definitions for bias and variance, because their decomposition method avoids potentially negative variance [169].

Given values of hypothesis H and target function F, the associated zero-one loss function for a test sample is a mapping from $\ell : Y \times Y \longrightarrow 0, 1$. i.e. $\ell(y_F, y_H) = 0$, if $y_F = y_H$, and 1 , otherwise. The cost, C is a random variable defined as the loss over the random variable Y_F and Y_H. The expected cost, or error rate, is expressed as:

$$E(C) = 1 - \sum_{y \in Y} P(Y_H = Y_F = y) \tag{4.1}$$

The above equation can be rewritten as:

$$\begin{aligned} E(C) = \sum_{y \in Y} -P(Y_H = Y_F = y) + \sum_{y \in Y} P(Y_H = y)P(Y_F = y) + \\ \sum_{y \in Y} [-P(Y_H = y)P(Y_F = y) + \frac{1}{2}P(Y_F = y)^2 + \frac{1}{2}P(Y_H = y)^2] + \\ \frac{1}{2} - \frac{1}{2}\sum_{y \in Y} P(Y_H = y)^2 + \frac{1}{2} - \frac{1}{2}\sum_{y \in Y} P(Y_F = y)^2 \end{aligned} \tag{4.2}$$

By rearranging the terms, we have

$$\begin{aligned} E(C) = \sum_{y \in Y} [P(Y_H = y)P(Y_F = y)] - P(Y_F = Y_H = y) + \\ \frac{1}{2}\sum_{y \in Y} [P(Y_F = y) - P(Y_H = y)]^2 + \\ \frac{1}{2}[1 - \sum_{y \in Y} P(Y_H = y)^2] + \\ \frac{1}{2}[1 - \sum_{y \in Y} P(Y_F = y)^2] \end{aligned} \tag{4.3}$$

We assume that Y_F and Y_H are conditionally independent, then $P(Y_H = Y_F = y)$ is actually $P(Y_H = y)P(Y_F = y)$, so the first term in the above equation is zero.

When estimating the expected cost for a fixed target and averaging over the training set T with size m, it is often written as:

$$\sum_x P(x)E(C|f,m,x) \qquad for \quad x \in T$$

So

$$E(C) = \sum_x P(x)((\sigma_x)^2 + bias_x^2 + variance_x) \tag{4.4}$$

where

$$\sigma_x^2 = \frac{1}{2}(1 - \sum_{y \in Y} P(Y_F = y|x)^2) \tag{4.5}$$

$$bias_x^2 = \frac{1}{2}\sum_{y \in Y}(P(Y_F = y|x) - P(Y_H = y|x)^2 \tag{4.6}$$

$$variance_x = \frac{1}{2}(1 - \sum_{y \in Y} P(Y_H = y|x)^2) \tag{4.7}$$

The “$bias^2$” is the difference between the learning algorithm’s average prediction and the target. It refers to the systematic error of the learning algorithm. The “*variance*” instead tells us ”how much the learning algorithm’s prediction bounces around for different training sets of the given size” [169]. It results from random variation and noise in the training set and from any random behavior of the learning algorithm. The σ_x^2 is the intrinsic noise of the learning algorithm. In Chapter VII we take into account the bias plus variance decomposition of the error rate to study the performance of various nearest neighbor classifier ensembles.

CHAPTER V

Feature Selection Methods: An Overview

5.1 Introduction

Feature selection[1]or attribute selection[2] has been a traditional research topic dating back to at least as early as the 70's (e.g. [203]). It is a broad subject that spans to research disciplines such as statistics ([207] [63] [200]), pattern recognition ([148] [255] [165]), data mining[3] ([80]), machine learning ([125]), neural networks ([241]), fractals ([72]), rough sets theory ([95]), mathematical programming ([64] [141]) and many others.

The advantages of feature selection are that it reduces the dimensionality of the feature space and removes the redundant, irrelevant or noisy data. The immediate effects for data analysis tasks are speeding up the running time of the learning algorithms, improving the data quality, increasing the accuracy of the resulting model.

So what is feature selection? Suppose $\mathbf{X}$ is the original feature space with a cardinality of q, and $\bar{\mathbf{X}}$ is the selected feature space with a cardinality of $\bar{q}$, $\bar{\mathbf{X}} \subseteq \mathbf{X}$, $J(\bar{\mathbf{X}})$ is the selection criterion for selected feature space $\bar{\mathbf{X}}$. Without loss of generality, we assume that a higher value of J indicates a better feature space. The goal is to

[1]Online feature subset selection bibliography can be found in [6] [8].

[2]Throughout this dissertation, we use feature selection, attribute selection, and variable selection without any distinction.

[3]Refer to the book [193] for feature selection for knowledge discovery and data mining, consult its appendix A for easy reference of the machine learning (ML), data mining and knowledge discovery (KD) resources or see online [31].

maximize $J()$. Formally, the problem of feature selection is to find a sub-space $\bar{\mathbf{X}} \subseteq \mathbf{X}$ such that

$$J(\bar{\mathbf{X}}) = \max_{\mathbf{Z} \subseteq \mathbf{X}, |\mathbf{Z}| = \bar{q}} J(\mathbf{Z}) \tag{5.1}$$

If an exhaustive approach is performed, then we need to consider all $\binom{q}{\bar{q}}$ possible combinations. The number of combinations grows exponentially, making the exhaustive search unfeasible for larger values of q. Even for moderate values of q, performing the exhaustive search is impractical. Finding the best feature subset is usually intractable [168], and many problems related to feature selection have been shown to be NP-hard [62]. There are three kinds of feature selection strategies: (i) The number of features, say $\bar{q}$ is already given, and the task of the search algorithms is to decide which $\bar{q}$ features constitute a (sub)optimal feature subset. (ii) The second strategy is to search the smallest feature dimensionality for which the discrimination performance exceeds a specified value. (iii) The third search strategy selects a (sub)optimal feature subset which has a trade-off between the class discriminability (e.g. classification error rate) and the subset size (e.g. the number of selected features).

5.2 Relevance to the Concept: Weak and Strong Relevance

Determining which of the features are relevant to the learning task is a central issue in machine learning, as the inclusion of irrelevant or redundant features can reduce the performance of different learning algorithms. In order to determine which of the features are relevant or not, we need to first know the concepts of weak relevance and strong relevance. There are a number of different definitions in the machine learning literature for what it means for features to be "relevant". John, Kohavi and Pfleger

[156] [155] define two notations of relevance [124]:

Strong Relevance: An attribute x_i is strongly relevant if its removal yields a deterioration of the performance of the Bayes Optimum Classifier.

Weak Relevance: An attribute x_i is weakly relevant if not strongly relevant and there exists a subset of variables V such that the performance on $V \cup \{x_i\}$ is better than the performance on V.

Therefore features that are neither strongly relevant nor weakly relevant are irrelevant. Irrelevant features should be left out.

5.3 General Characteristics of Feature Selection Methods

Feature selection aims to search the relevant features in the feature space. Researchers have studied various aspects of feature selection. From the point of view of heuristic search, Blum and Langley [61] argue that the following four issues, which affect the nature of the search, can characterize any feature selection method.

1. The starting point in the feature space.

   Depending on which point to start with, the search direction will vary. Search from no features and successively add others is called forward selection. In contrast, search from all features and successively remove features is called backward selection. A third method could be to combine forward and backward search.

2. The organization of the search procedure.

   Obviously, if the number of features is too large, the exhaustive search of all the feature subspace is prohibitive, as there are 2^N possible combinations for N features. For example, heuristic search is more realistic than exhaustive search, but it doesn't guarantee finding the optimal solutions.

3. The evaluation strategy.

How feature subsets are evaluated is an important problem. As for classification, the ideal feature subset should have the best separation of the data. Data separation is usually computed by an inter-class distance measure [166]. Another most frequently used discriminating measure is the Wilk lambda [276], it is defined as follows:

$$\lambda = \frac{|W|}{|W + \bar{B}|} \tag{5.2}$$

where W is the intra-class matrix dispersion corresponding to the selected variable set, $\bar{B}$ is the corresponding inter-class matrix, $|W|$ is the determinant of matrix W. W and $\bar{B}$ are computed respectively:

$$W = \sum_{j=1}^{g} \sum_{l=1}^{N_j} (x^l - \mu^j)^t (x^l - \mu^j) \tag{5.3}$$

$$\bar{B} = \sum_{j=1}^{g} N_j (\mu - \mu_j)^t (\mu - \mu_j) \tag{5.4}$$

where g the number of classes, N_j the number of samples in class j, μ_j the mean of class j and μ the global mean. The smaller the value of λ, the better discriminating power it indicates.

In this dissertation, classification accuracy is used for the evaluation of the feature subset. Classification accuracy is defined as the percentage of test examples correctly classified by some algorithm. Many induction algorithms incorporate a criterion based on information theory, others directly measure accuracy on the training set.

4. The criterion for stopping the search.

   During the process of evaluation, we might want to stop the search, when observing that there is no improvements of the classification accuracy.

5.4 Categorization Scheme of Feature Selection Methods

There is plenty of effort to compare and evaluate different feature selection methods [203] [178], but there are very few attempts to categorize the feature selection methods in the literature. Siedlecki and Sklansky [245] discussed the evolution of feature selection methods and grouped the methods into past, present and future categories. Their main focus was the branch and bound method and its variants. Dash and Liu [93] divided 32 existing feature selection methods into different groups based on the major two characteristics of feature selection: generation procedure (complete, heuristic, random) and evaluation function (distance, information, consistency, classification error rate). A taxonomy of feature selection algorithms into broad categories was given by Jain and Zongker [147], where the methods were first divided into those based on statistical pattern recognition (SPR) classification techniques, and those using artificial neural networks. The SPR category was then further divided into sub-categories. The categorization can also be simply done according to the monotonicity of the selection evaluation criteria, that is, monotonic versus non-monotonic. Another categorization could be according to the time complexity of the feature selection algorithm, e.g. the time complexity of floating search methods [217] is $O(2^n)$,[4] while that of the sequential backward and sequential forward

[4]Let's take a somewhat pragmatic viewpoint of the time complexity for (both forward and backward) floating search methods through an example. Suppose the number of selected features $\bar{q}$ is much less than the original feature space with q features, say, $q = 10 * \bar{q}$. Then the time complexity for *forward* floating search is [240]:
$O_{forward} = 9.5 * \bar{q}^2 + 0.5 * \bar{q}$,
while that for *backward* floating search is:
$O_{backward} = 49.5 * \bar{q}^2 + 4.5 * \bar{q} + 1$.
One can see clearly in this case the time complexity for backward floating search is more than *five* times as that for forward floating search, this indicates that the forward version of floating search is preferred when both the forward and backward floating search methods are applicable.

selection methods is $\Theta(n^2)$, where Θ denotes a tight estimate of complexity, while O denotes an estimate of complexity for which only an upper bound is known. Moreover, the feature selection methods can be categorized into two general groups [177], that is, the classifier-*specific* selection methods where the goodness is evaluated by a given criterion (e.g. the error rate of a certain classifier, this is useful for cases where we know which classification will be performed after selection) and the classifier-*independent* selection methods where the goodness is evaluated by the methods' own criterion (e.g. measures based on the approximation of class-conditional probability density functions, this is useful for cases where we don't know which classification will be used). Other categorization schemes include simply dividing the feature selection methods into: optimal (e.g. exhaustive search) vs. non-optimal (suboptimal), from the point of view of the optimality of the resulting subset; backward elimination vs. forward selection, from the point of view of starting point in the feature search space; and many others.

On the other hand, feature selection can be generally regarded as an optimization problem. For a general optimization problem, one may use the NEOS optimization tree category [13], that divides optimization techniques into discrete optimization and continuous optimization, both of which are then further divided into other subcategories. For more references on optimization, the reader is referred to Optimization Online [29].

We describe three typical model approaches in the following,[5] i.e. the Filter Selection Model, the Wrapper Selection Model, and the Embedded Selection Model.

[5] It seems to me that there is also a need to categorize the availale selection methods into two major groups, i.e. those promising ones for *large dataset* (e.g. the genetic feature selectors which are discussed in Chapter VI and floating search methods [217] [253]) that are very effective for large scale databases, especially nowdays at a time when information grows at an amazing speed, and those less-promising ones that have weak power to deal with the high dimensional data due to reasons such as the costly computation complexity problem, therefore they are mainly targeted for the use with small or medium scale databases.

5.4.1 Filter Selection Model

The filter selection model is the earliest approach to feature selection. It utilizes an independent search criterion to find the appropriate feature subset before a machine learning algorithm is performed, thus it was termed as *filter* method by John, Kohavi and Pfleger [156]: it filters out irrelevant attributes before induction occurs, that is, the search is done independently of an induction algorithm. The procedure of the filter model is shown in Figure 5.1. The advantage of the filter model is that it does not need to re-run the algorithm for every induction algorithm when choosing to run on a reduced feature dataset, as a consequence, the filter approach is generally computational efficient, and it is practical for data sets with very high dimensionality.

There are a number of different representative filter algorithms in the literature. FOCUS, an algorithm designed by Almuallim and Dietterich [45] originally for the boolean domain, searches the feature space by looking at each feature in isolation, then turn to pairs of features, triples, and so on, and stops until it finds the minimal combination of features. The minimal feature subset divides the training data into pure classes, i.e. no instances have more than one class. The original training samples which are characterized by the resulting feature subset, are then passed to the decision tree induction algorithm ID3 [221].

Another representative work of the filter approach is the RELIEF algorithm due to Kira and Rendell [163]. The RELIEF algorithm follows the general and simple filter scheme, that is, it first evaluates the individual feature according to the evaluation criterion, and thereafter, the best n features are selected. However it uses a more complex evaluation function. The training samples, characterized by the selected features, are then passed to ID3. Two extensions were made to this algorithm by Kononenko [172], where more general data types can be treated. Although both

FOCUS and RELIEF use the decision tree induction algorithm after feature selection, they are naturally not confined to decision tree algorithms, i.e. other induction algorithms can be used instead.

Table 5.1 shows a list of filter approaches to feature selection in the literature.

Since the filter approach does not take into account the learning bias introduced by the final induction algorithm, it may not be able to select the most suitable subset for the final induction algorithm. For this reason, the wrapper model was proposed.

5.4.2 Wrapper Selection Model

The strategy of the wrapper model is to use an induction algorithm to estimate the merit of the searched feature subset on the training data and using the estimated accuracy of the resulting classifier as its metric . The wrapper approaches often have better results than the filter approaches because they are tuned to the specific interaction between an induction algorithm and its training data. In Chapter VI, we discuss a typical wrapper selection approach called genetic feature selectors, which use genetic algorithms as the search engine (and one of them uses the ensembled nearest neighbor classifiers as the induction algorithm). In this way, feature selection takes into account the biases from the final learning algorithm. The use of wrapper approaches was supported by the study of Aha and Banket [41], Doak [103] and John *et al.* [156].

The wrapper selection procedure is illustrated in Figure 5.2.

The disadvantage of the wrapper model is that it is less tractable because of the prohibitive cost of running the classification algorithm many times when the dimensionality is considerably high.

Table 5.2 shows different wrapper approaches to feature selection in the literature.

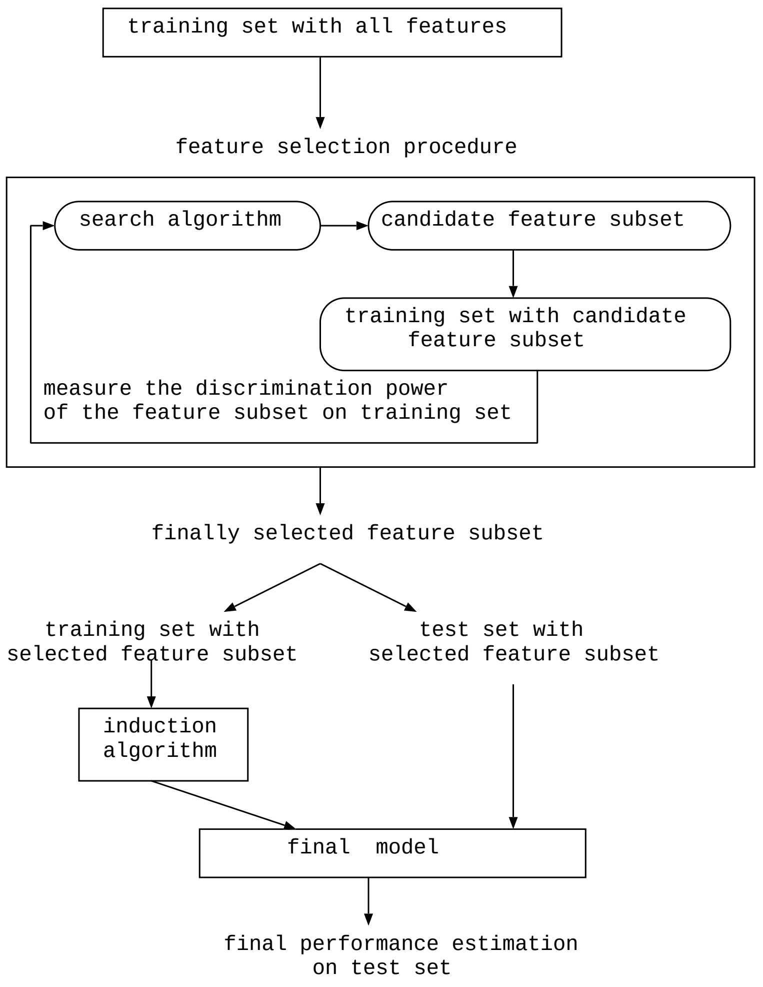

training set with all features
feature selection procedure
search algorithm
candidate feature subset
training set with candidate feature subset
measure the discrimination power of the feature subset on training set
finally selected feature subset
training set with selected feature subset
test set with selected feature subset
induction algorithm
final model
final performance estimation on test set

Figure 5.1: Filter selection procedure.

Author(System)	Starting Point	Search Control	Evaluation Criterion	Learning Algorithm
Aha and Bankert [41]	None	Beam variants of forward and backward selection	Calinski-Harabsz separability index[a]	IB1
Almuallim and Dietterich:FOCUS [45]	None	Breadth-First	Consistency[b]	ID3
Cardie [75]	None	Greedy	Consistency	Nearest Neighbor
Kira and Rendell:RELIEF [163]	—	Ordering	Threshold	ID3
Singh and Provan [249]	None	Greedy	No Information Gain	Bayesian Network
Koller and Sahami [170]	All	Greedy	Threshold	Tree/Bayes
Liu and Setiono:LVF [192]	Random	Las Vegas[c]	Consistency	ID3
Kubat et al [176]	None	Greedy	Consistency	Naive Bayes
Schlimmer [239]	None	Systematic[d]	Consistency	None

Table 5.1: Summary of different filter approaches to feature selection.

[a] See reference [73].

[b] As described in Section 5.4.1, the FOCUS algorithm searches each feature in isolation, then turn to pairs of features, triples, and so on, until it finds the minimal subset which divides the training data into pure classes, in this context, we say that, the FOCUS algorithm finds a minimal subset which is *consistent* with the training data.

[c] Las Vegas algorithm uses *randomness* to guide search, i.e. a *random* subset is generated during each round of search.

[d] Systematic search is used to avoid revisiting search states.

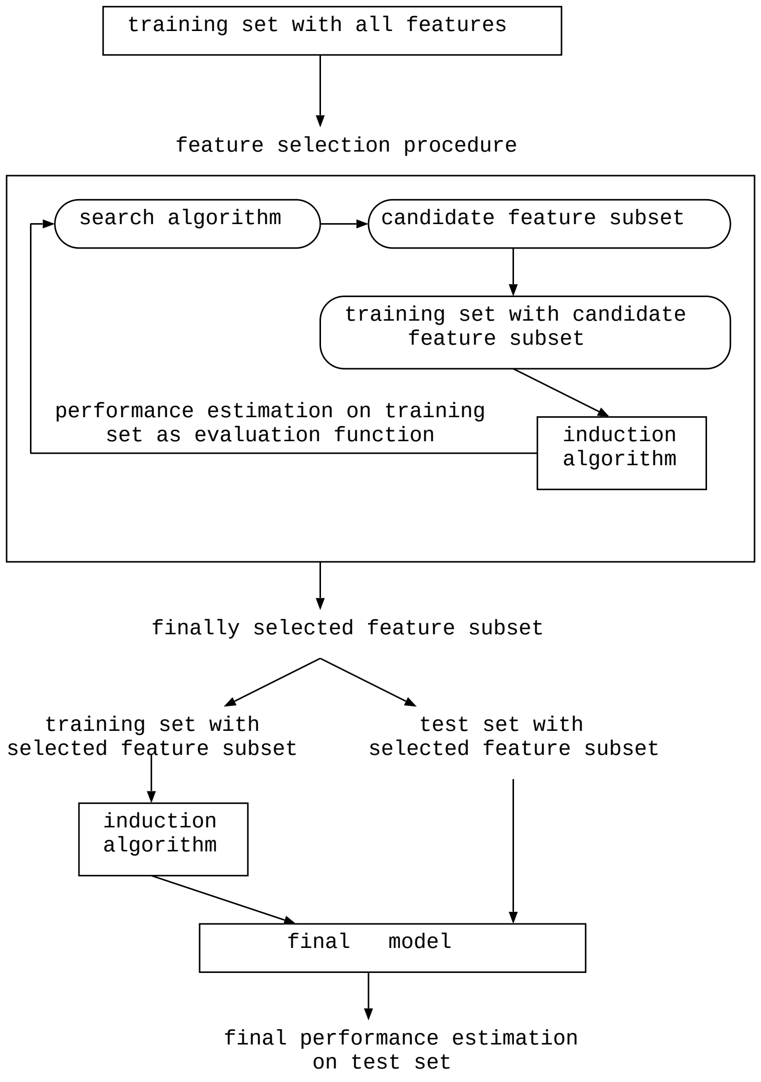

training set with all features
feature selection procedure
search algorithm
candidate feature subset
training set with candidate feature subset
performance estimation on training set as evaluation function
induction algorithm
finally selected feature subset
training set with selected feature subset
test set with selected feature subset
induction algorithm
final model
final performance estimation on test set

Figure 5.2: Wrapper selection procedure.

Author(System)	Starting Point	Search Control	Evaluation Criterion	Learning Algorithm
Aha and Bankert[41]	Random	Beam variants of forward and backward selection	Leave-one-out cross validation	IB1
Moore and Lee [204]	Comparison	Greedy	No better	1-NN
Skalak [251]	Random	Mutation[a]	k-fold Cross Validation	1-NN
Langley and Sage [186]	All	Greedy	k-fold Cross Validation	1-NN
Langley and Sage [185]	None	Greedy	Accuracy on training set	Naive Bayes
Singh and Provan [248]	None	Greedy	3 Information theoretic measures	Bayesian Network

Table 5.2: Summary of different wrapper approaches to feature selection.

[a] A binary vector is used to represent the feature subset, on each search step, mutate one element, or bit, chosen at a random point of the binary vector, and then evaluate the new vector.

5.4.3 Embedded Selection Model

In contrast to the wrapper approach, which treats feature selection as a *wrapper around* the induction process, the embedded approach *embeds* the selection *within* the basic induction algorithm. Examples of this model are the decision tree algorithms ID3 and C4.5[6] [7] by Quinlan [221] [222] and CART [8] by Breiman [68]. These decision tree algorithms use recursive partitioning methods for induction, and carry out a greedy search through the space of decision trees. At each stage they use an evaluation function to select the attribute that has the best ability to discriminate among the classes. They partition the training data based on this attribute and repeat the process on each subset, extending the tree downwards until no further discrimination is possible.

Besides these three approaches, another model called weighted model was also introduced [214] [213], where feature weighting is considered.

[6]The fuzzy set theory has been introduced into decision trees by many researchers [151] [140] [267] [153].

[7]Quinlan has developed advanced version called C5.0/see5 [34].

[8]An algorithm called RECPAM [83], is a generalization of the well-know CART algorithm. The difference between ID3 and CART is that while ID3 aims at knowledge comprehensibility and is based on symbolic domains, CART is naturally designed to deal with continuous domains but lacks the same level of comprehensibility.

CHAPTER VI

Genetic Feature Selectors

6.1 Introduction[1]

In this section we will introduce some background of evolutionary algorithms and meanwhile situate the genetic algorithms within the frame of evolutionary algorithms. Evolutionary algorithms (EAs) [48] [11] [5] are a broad class of different randomized search heuristics, which currently include Evolution Strategies (ESs) [49], Evolutionary Programming (EP) [111], Genetic Programming (GP) [21] [173] and Genetic Algorithms (GAs) [123]. They all stem from modeling the natural evolution processes. Although based on the same evolutionary principles, each of them employs its own particular chromosomal representation, set of genetic operators and selection and replacement scheme. Evolutionary computation[2] [3] [4] [50] has found countless real world applications, e.g. it has been also applied to the field of hyperspectral image analysis and remote sensing [225] [128] [71]. On the other hand, evolutionary algorithms were already proposed for optimization in the 60's [69] and the research on this continues [24].

Genetic algorithms are a successful soft-computing technique for solving optimiza-

[1]This chapter, together with part of Chapter VII, is the generalized work of [284].

[2]We don't make any distinction between evolutionary algorithms and evolutionary computation here.

[3]For more reference on evolutionary computation, refer to the page of the International Society for Genetic and Evolutonary Computation (ISGEC), and check the menu of *Books by ISGEC members* [23].

[4]Cooperative coevolutionary computation [216] [215] is yet another type of evolutionary computation, which promises many advantages over traditional evolutionary algorithms in terms of the corresponding adaptability and potential open-endedness.

tion problems, and were already applied to the problem of feature selection [246] [179] [144] [282]. GAs were found to be very efficient to do so [147]. In this chapter we will discuss the problem of using the genetic algorithms as the search engine to perform feature selection for high dimensionality with limited training data.

The outline of this chapter is as follows: in the following two sections, we first describe the basic concepts of the standard genetic algorithms, we then address the categorization scheme on the existing feature selection methods using genetic algorithms, one of which is a feature selector using genetic algorithms in conjunction with an ensembled learning scheme.

6.2 Genetic Algorithms

Genetic Algorithms (GAs)[5], invented by Holland [136] in the 70's, are general purpose search algorithms that utilize the principles inspired by natural population genetics to evolve solutions to problems.

Although different variants of genetic algorithms vary in many aspects, they share a prototypical procedure as shown in Figure 6.1. GAs are an iterative optimization process where a set of operators, such as crossover and mutation, are applied. First a solution is represented by a finite sequence of 0's and 1's, called a chromosome. The chromosomes are allowed to 'crossover', e.g. for two parental chromosomes, the simple way is to choose randomly some point (called crossover point) and everything before this point copies from the first parent and then everything after this crossover point copies from the second parent. In this way, two parental chromosomes exchange their parts at the crossover point to create two new child chromosomes. Chromosomes are also allowed to 'mutate', i.e. a small change (e.g. flipping of a bit) can be made

[5]See [15] for online genetic algorithm archive. Jarmot T. Alander of Finland has compiled a series of indexed bibliography of Genetic Algorithms and their applications in many areas. The series of bibliography can be downloaded from [2].

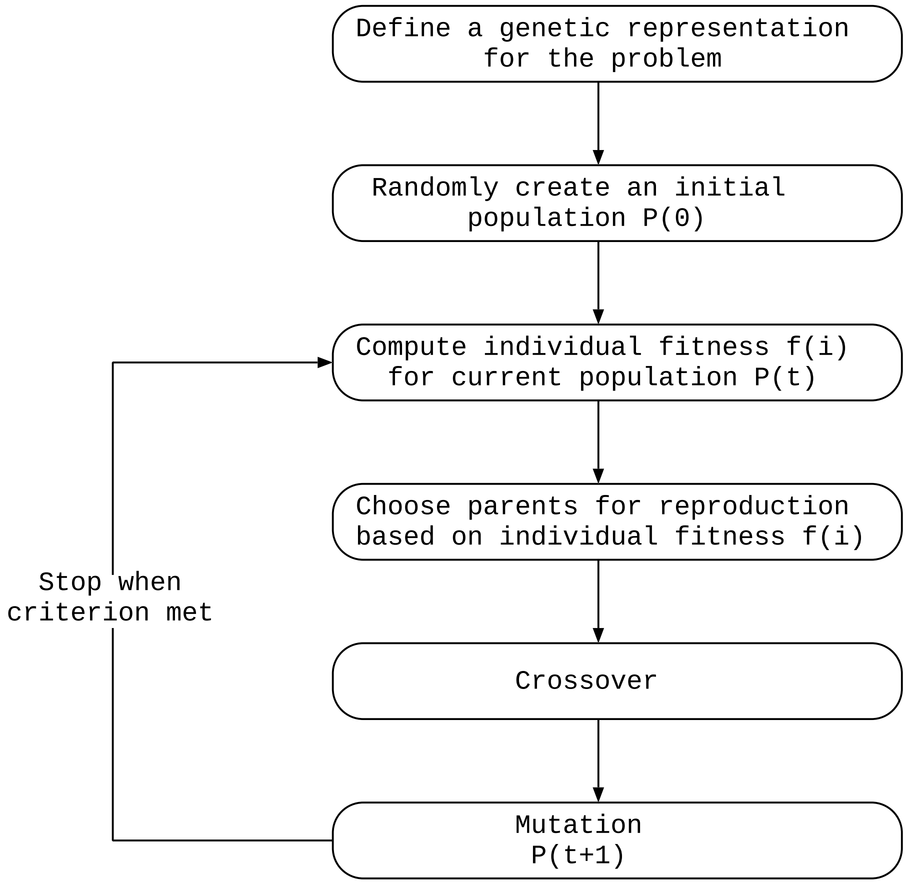

Define a genetic representation for the problem
Randomly create an initial population P(0)
Compute individual fitness f(i) for current population P(t)
Choose parents for reproduction based on individual fitness f(i)
Crossover
Mutation P(t+1)
Stop when criterion met

Figure 6.1: Outline of the standard GAs.

to a chromosome. The optimization process is carried out in 'generations', where each time a population of new chromosomes is generated. Since the population size is finite, only the 'best' chromosomes are allowed to survive. A 'fitness' function is defined that allows to calculate a fitness score for each of the chromosomes.

Due to the inherent parallel nature of genetic aglrithms, many parallel versions of genetic algorithms have been proposed in the literature [74]. The theory of fuzzy set has also been introduced into genetic algorithms [275] [131]. For a throughout study of various aspects of genetic algorithms, the reader is referred to the following standard introductory material [123] [137] [199] [202].

6.3 Categorization Scheme of Genetic Feature Selectors

Genetic feature selectors are a series of feature selection methods which use genetic algorithms to guide the selection procedure. Basically speaking, the genetic feature selectors fall into the category of the wrapper model as described in Section 5.4.2 of Chapter V. GAs may, in general, hybridize with any classification scheme, therefore they can also be categorized into two groups, that is, those that combine with a *single* learning scheme (i.e. non-ensembled learning scheme) and those that combine with an *ensembled* learning scheme, e.g. in [284], where an ensemble of nearest neighbor learning algorithms is proposed to evaluate the merit of the features selected during the selection process. In Chapter VII, we will do some experiments where the ensembled learning scheme is used, and demonstrate its superiority over the single learning scheme for genetic feature selection.

Genetic search is a type of search which has the following properties: it is (i) a stochastic search, (ii) a multi-point search, (iii) a direct search, (iiii) a parallel search. Due to these reasons, genetic algorithms have been applied to the problem of feature selection. The genetic feature seletion was originally inspired by the seminal work by Siedlecki and Sklansky [246]. They designed a genetic feature selection algorithm that was found to be very efficient for high (> 20) dimensionalities [147]. Later work by Kelly and Davis [162], and by Punch *et al.* [218] expanded this approach to use GAs for feature extraction. Raymer *et al.* [227] [226] then further extended the genetic feature selection through the simultaneous optimization of feature weights and selection of key features by including a masking vector on the GA chromosome. For our work, we followed the procedure of Siedlecki and Sklansky's seminal work [246] and mainly initiated the idea of using the ensembled learning scheme as a means to evaluate the intermediate subsets during the selection stage [284].

Unlike classical optimization procedures the genetic feature selector does not optimize a single solution, but, instead, it modifies a population of solutions at the same time. This guarantees at least a suboptimal optimization.

For the problem of feature selection, a chromosome has length d, the total number of features. A '1' stands for a selected feature, whereas a '0' stands for a rejected feature. There are two ways to optimize such a binary string. One way is to minimize the classification error rate. This however will not necessarily minimize the number of selected features. Better is to optimize both the classification performance and the number of selected features simultaneously.

We will categorize the genetic feature selection methods in the following based on how the fitness function is chosen. Optimization problems by evolutionary algorithms can be broadly divided into single-objective optimization and multi-objective optimization. In real world, the direct use of single-objective optimization, where the objective funtion and fitness function are identical, is very rare. Rather optimizing several objectives simultaneously is very common. Based on the way the fitness assignment and selection are performed, the existing feature selection methods by genetic algorithms can be classified into the following two categories [289]: Aggregation Selection with Parameter Variation (Section 6.3.1) and Pareto-based Selection (Section 6.3.2).

6.3.1 Aggregation Selection with Parameter Variation

In this category, different objectives are combined, or aggregated into one scalar fitness function. The weighted sum approach is very popular in this category, which adds different objective functions using different weighting coefficients into one composite fitness function [86]. In practice, defining a suitable trade-off between different objectives needs the knowledge of the domain concerned, and is, as a consequence, in general a non-trivial task. One of the advantages of this approach is that it is first of all the simplest and most efficient, because no further interaction with the decision maker is needed [86]. Besides, the optimization is done in multiple directions, in that all members of the population are evaluated by a different objective function [289]. Siedlecki and Sklansky's idea [246] is to define a threshold error rate t, and to find the binary string with the lowest number of selected features that leads to an error rate e , lower than t. A fitness function is defined as follows:

$$f(a_i) = J(a_j) - (< J(a_i) > -\eta \Delta J(a_i)) \tag{6.1}$$

where $< . >$ and Δ are the mean and standard deviation over the population, η is a small positive constant which assures that $minf(a_i) > 0$, i.e. even the least fit chromosome is given a chance to reproduce. The score $J(a)$ of a string a is given by:

$$J(a) = l(a) + p(e(a)) \tag{6.2}$$

with $l(a)$ is the 'length' (= number of '1's) of string a, and $p(e)$ is a penalty function[6] for the obtained error rate e. If e is below the threshold error rate t, $p(e)$ is negative, and if e grows larger than t, $p(e)$ grows rapidly:

$$p(e) = \frac{\exp\frac{(e-t)}{\xi} - 1}{\exp(1) - 1} \tag{6.3}$$

with ξ a small scaling parameter (about 1%).

[6]Penalty function has been the approach to constrained optimization problems in the literature for decades. The penalty methods for constrained genetic optimization were already addressed [247] [87]. Smith and Coit [252] categorize the penalty functions into three groups, i.e. *static* penalty functions, *dynamic* penalty functions and *adaptive* penalty functions. Runarsson and Yao [232] show that applying different penalty function methods in evolutionary optimization is equivalent to using different selection schemes.

Yang and Honavar's [282] genetic feature selector combines a neural network classifier with a standard genetic algorithm. They defined a fitness function which combines two different criteria — the classification accuracy by the neural network and the cost on the classification:

$$fitness(\Omega) = accuracy(\Omega) - \frac{cost(\Omega)}{accuracy(\Omega) + 1} + cost_{max} \tag{6.4}$$

where Ω is the feature subset, $fitness(\Omega)$ is the fitness on Ω. $accuracy(\Omega)$ is the classification accuracy by the neural network classifier on the subset Ω, which can be estimated by calculating the percentage of patterns in a test set. $cost(\Omega)$ is the cost of classification, which has a number of different measures, e.g. cost of measuring the value of a specific feature needed for classification, the risk involved, etc. $cost_{max}$ is the upper bound on the costs of candidate solutions.

Ishibuchi and Nakashima's work [144] is very similar to that of Kuncheva and Jain's [179], both of them actually optimize three competing objectives simultaneously: minimize the training set, minimize the classification error rate, and minimize the number of selected features.

6.3.2 Pareto-based Selection

While the method of aggregation selection with parameter variation is commonly used, it has several disadvantages. First of all, the difficulty with artificially designed composite fitness funtions is that, one should at least be aware of the behavior of every objective function beforehand. Secondly, The trade-off between model accuracy and complexity is difficult to explore. Thirdly, the use of penalities and weights has proven to be problematic. That is, the final GA solution is usually very sensitive to small changes in the penalty function coefficients and weighting factors [229]. For this, the Pareto-based selection approach was proposed.

A solution is said to be dominant if its performance is superior over another with respect to all criteria. A solution is said to be Pareto optimal [112] if it cannot be dominated by any other solution. Under this category, we are only aware of the work by Emmanouilidis *et al.* [107], a multi-objective genetic feature selector, which uses multiobjective genetic algorithms aiming at producing Pareto optimal feature subsets. Their main idea is to use a variant of the Niched Pareto Genetic Algorithm (NPGA) [138] to do feature selection.

In essence, genetic feature selection falls into the scope of multi-objective optimization as it must optimize several objectives (e.g. minize the error rate and minize the number of selected features) simultaneously. Multi-objective optimization is an active yet rich research area. Different methods for multi-objective optimization by evolutionary algorithms are studied and compared [290] [257] [219]. For a literature study, we refer the reader to the following remarkable surveys [256] [113] [112] [85] [86] [272] [273]. For an excellent self-contained introduction on the topic of multi-objective optimization, we recommend the dissertation by Van Veldhuizen [271]. For more comprehensive references on multi-objective optimization using evolutionary algorithm, see the unique online EMOO (Evolutionary Multi-Objective Optimization) repository of information in [26], which could also serve as a gateway for those who are interested in this area.

CHAPTER VII

Experiments and Discussion

7.1 Introduction

The dataset used for the empirical study is an AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) dataset which was shown in Figure 1.1 in Section 1.1 of Chapter I.

In this chapter, experiments are carried out with nearest neighbor classifier ensembles and genetic feature selection respectively. Due to our lack of accessing hyperspectral data, we conduct all the experiments on the freely available dataset as described above, but validate some of the experiments in the following two ways: We conduct the experiments on (i) the first N bands of the data; and (ii) the last N bands of the data; where N can take values of 20, 40, 60, ..., 220. Of course, another alternative to validate the data could be randomly shuffling the spectral bands and then take the first N bands, and then randomly shuffling the spectral bands again and take the first N bands again, and so on, but this leads to the question of how to design an optimal experiment given a very limited hyperspectral dataset, which is beyond the scope of this dissertation, though it might be interesting.

Class Name	Corn-min	Grass/pasture	Grass/trees	Soy-clean	Wheat
No. of Samples	144	198	184	140	126

Table 7.1: Description of the 5 classes in the AVIRIS data set.

7.2 Experiments of Nearest Neighbor Classifier Ensembles

All experiments were carried out on a five-class problem as described in Table 7.1, and were implemented in Java[1] with the use of the machine learning package — *Weka* [19] from the University of Waikato, and the open source library — *Colt* [12] from CERN. The random number generator used here is the implementation of Mersenne Twister algorithm after Matsumoto and Nishimura [197], one of the strongest uniform pseudo-random number generators known so far, and at the same time it is quick. For more reference on pseudo-random number generators, the reader is referred to [28].

We conducted three experiments, all experiments were done using a training set and a test set. We used Aha and Bankert's [43] setting scheme on training set and test set, that is, each time randomly splitting the original data set into 70% training set and 30% testing set. The calculation of bias (Equation 4.6) and variance (Equation 4.7) in all the three experiments is base on Section 4.4. The experiment was run ten times, the average result was then obtained.

7.2.1 Experiment One

In the first experiment, the methods which utilize the dynamic nature of the Condensed Nearest Neighbor (CNN) learning algorithm are compared. In particular, we study the performance of the proposed CNN-ECOC method. The methods are CNN, voting CNN (both simple and weighted), Union CNN, CNN-ECOC and NN. NN is

[1]For the Integrated Development Environment (IDE) for Java, I used the free package — the Community Edition of then *Forte For Java*, now called *Sun ONE Studio* from Sun MicroSystems [35]. Another better alternative is to use the free package of *Eclipse* [20].

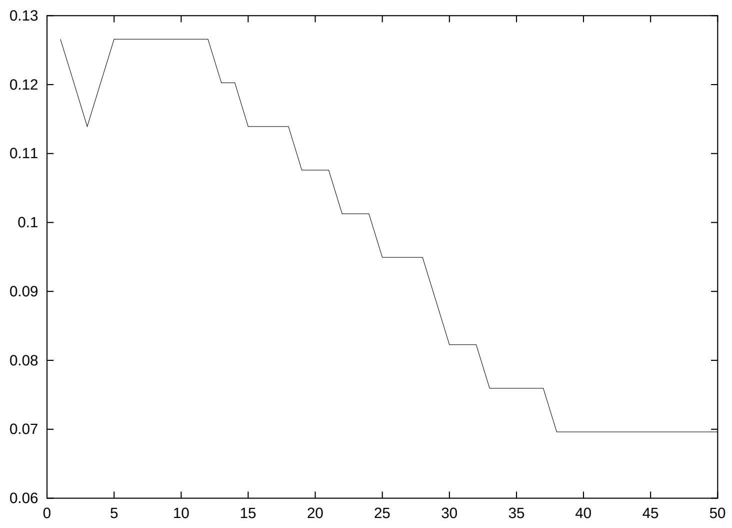


Figure 7.1: Error rate as a function of the number of voters for 220-spectral-band data.

used as the base for comparison. See Section 4.3.2 for details about Voting CNN, Union CNN, and Section 4.3.5 for the CNN-ECOC method. As already pointed out in Chapter IV, the CNN is very well suited for classification tasks that demand a reasonably small memory footprint, but meanwhile tolerate some acceptable performance deterioration.

The error rate, bias and variance are plotted respectively in function of spectral bands, starting from 20 bands to the full 220 bands: i.e. the first/last 20 bands, the first/last 40 bands, ..., the full 220 bands. Figures 7.3 — 7.8 show respectively the comparison of error rate, bias and variance between nearest neighbor classifier (NN), condensed nearest neighbor (CNN), simple voting of CNN, weighted voting of CNN, union voting, and CNN-ECOC. The voting method, which performs better with the increase of the number of voters (see Figure 7.1), doesn't show higher performance gains when the number of voters is too small. The number of stored patterns of this method increases with the number of voters, and it is almost a linear relation as shown in Figure 7.2. In our experiment, we set the number of voters to 7.

From the plots 7.3 — 7.8, one can obtain the following conclusions:

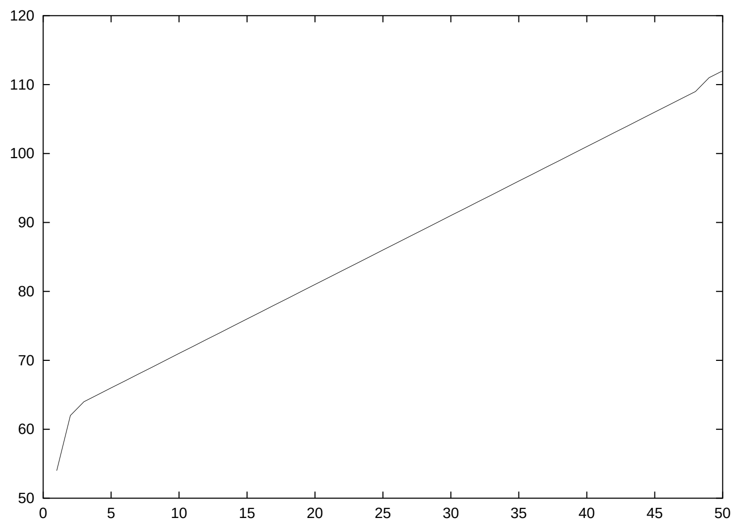


Figure 7.2: The number of stored patterns as a function of the number of voters for 220-spectral-band data.

- While weighted voting CNN is regarded to perform better than simple voting CNN, one can observe that the method of weighted voting CNN doesn't show advantages over the method of simple voting CNN: their performances coincide completely on our dataset. This maybe due largely to the setup of the weighting coefficients: only the first two nearest neighbors are taken into account.

- Although some author [47] reported that the performance of the union CNN, which also has less computational cost, is promising on some datasets, this conclusion did not appear on our dataset. One can also see that the performance of union CNN at some points is becoming worse than CNN alone, while the voting CNN outperforms CNN alone over all spectral bands. Since the subset of union CNN is obtained by applying NN to the union of subsets chosen by multiple CNN's, the resulting union CNN has a considerable number of training samples. Our results highlight that it is the voting process, not the number of training samples, that contributes to the performance gains.

- We can also see that the proposed method of CNN-ECOC achieves a system-

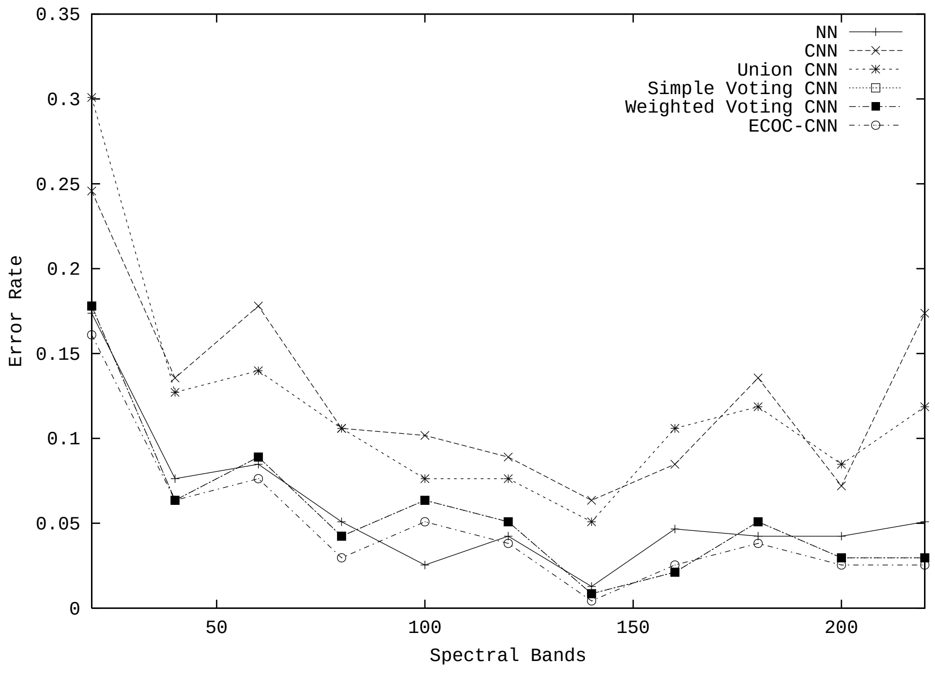


Figure 7.3: Comparison of error rate between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

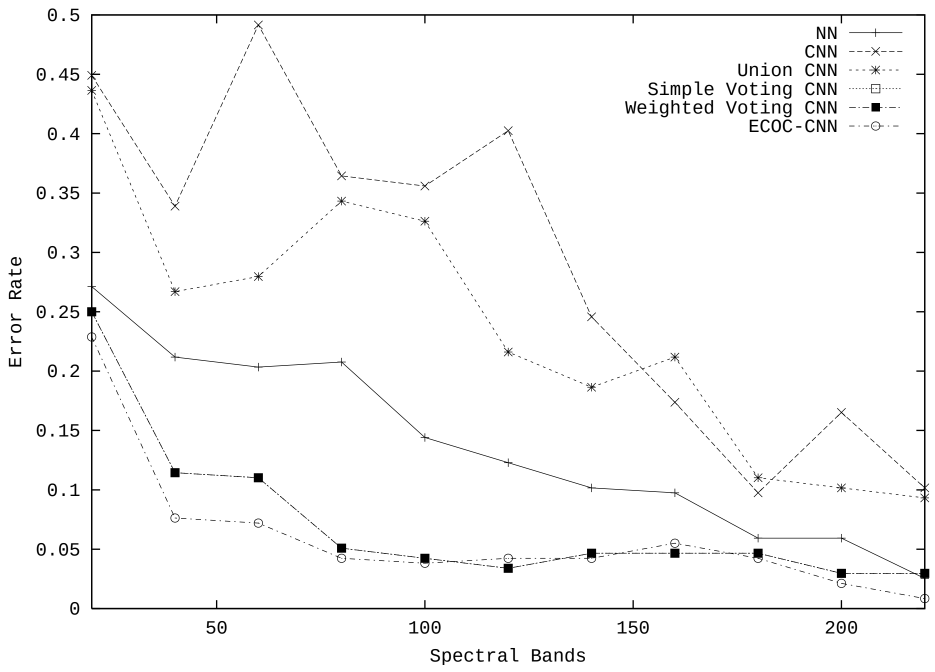


Figure 7.4: Comparison of error rate between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands.

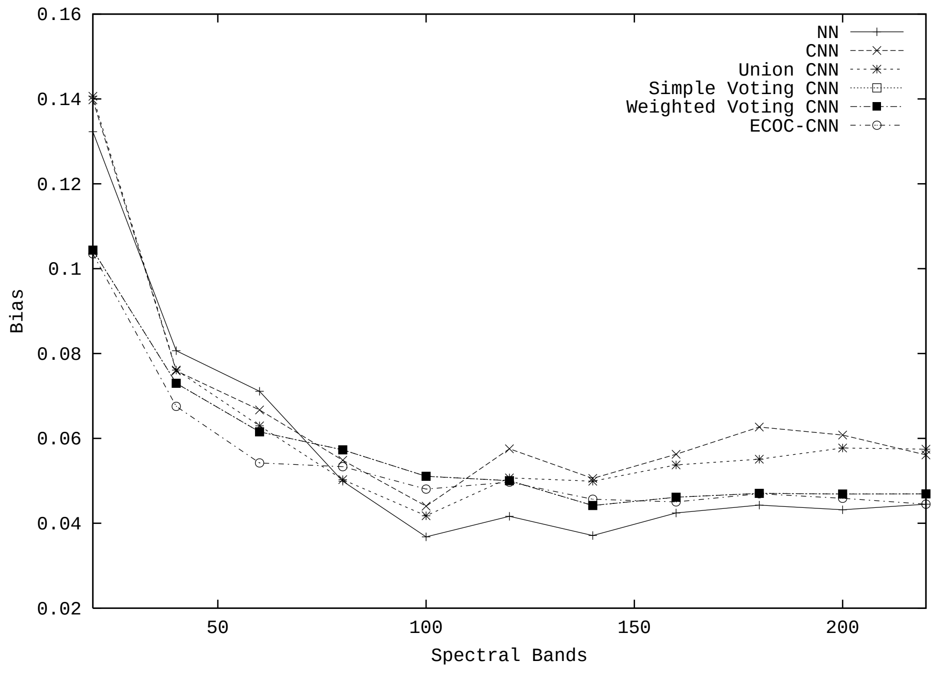


Figure 7.5: Comparison of bias between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

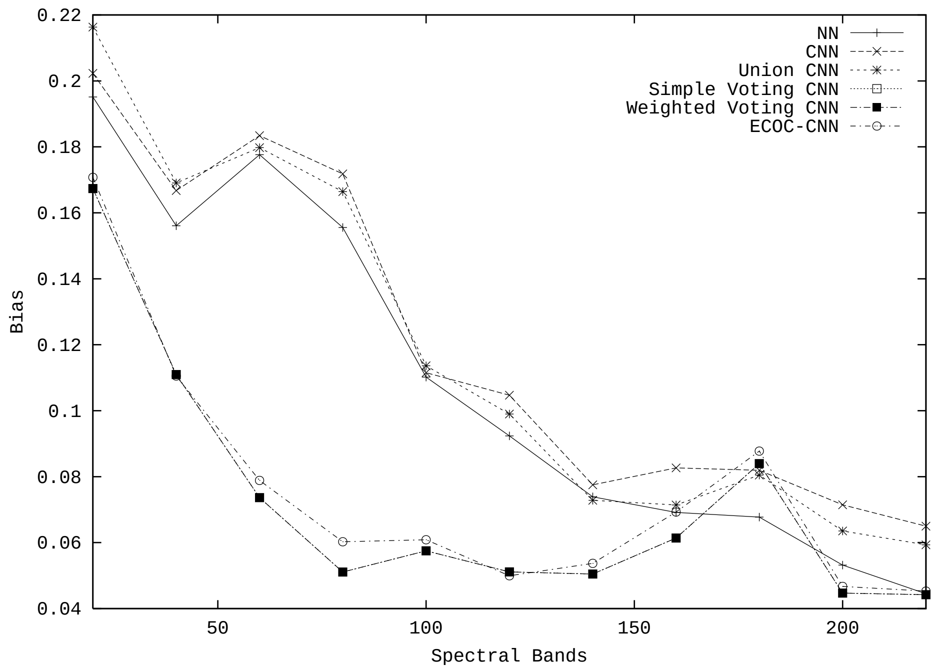


Figure 7.6: Comparison of bias between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands.

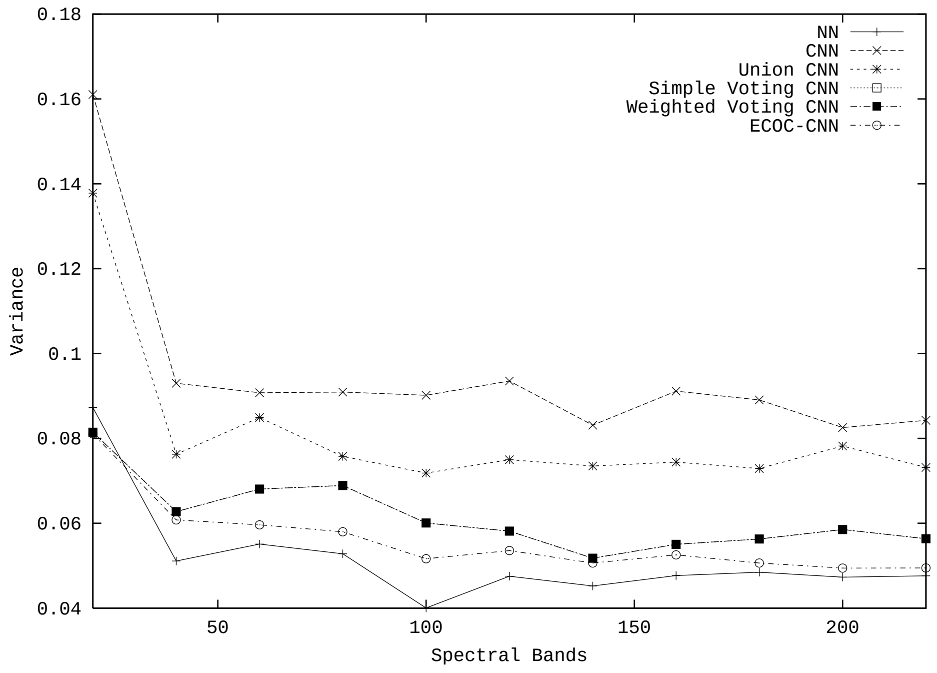


Figure 7.7: Comparison of variance between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

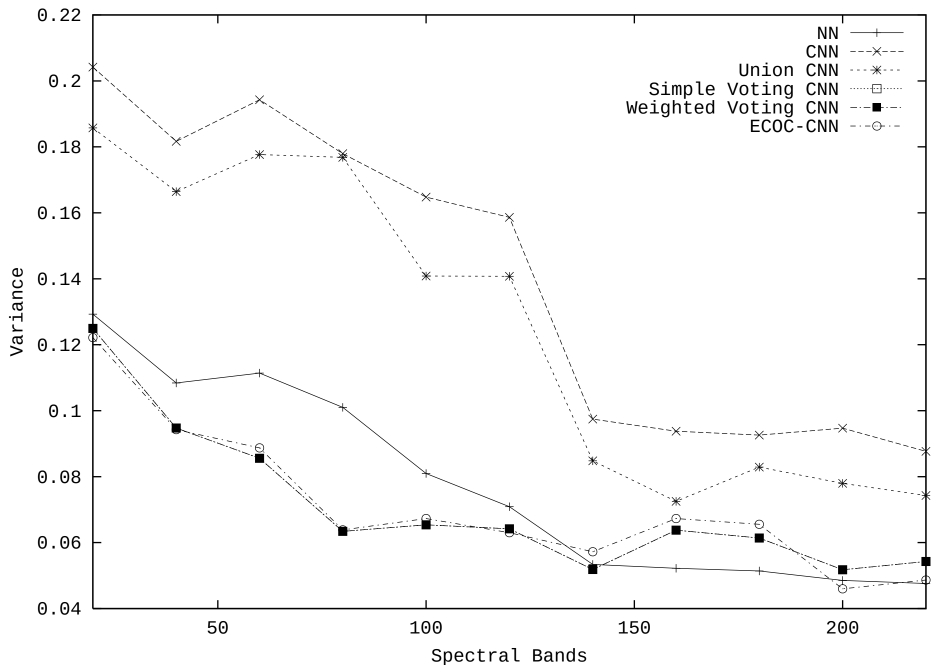


Figure 7.8: Comparison of variance between NN, CNN, Union CNN, Simple Voting CNN, Weighted Voting CNN, CNN-ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands.

atic classification performance improvement over CNN, and this improvement is comparable with that of NN (Figure 7.3) or even better than NN (Figure 7.4) while keeping a lower memory demand for the training set than NN. ECOC decreases both bias and variance (of the classifier it combines with) [171], this claim can be verified from Figure 7.5, Figure 7.6, Figure 7.7 and Figure 7.8. In particular, the CNN-ECOC has a smaller variance than CNN. This shows that while the classification performance of CNN depends largerly on the specific dataset involved (i.e. in part due to the nature of the local search that the CNN uses), the tendency that, the classification performance of CNN-ECOC varies from dataset to dataset, is mitigated by the use of ECOC technique. That is, by decomposing the multi-class problem into mutiple two-class problems, the ECOC technique is able to diminish the errors occurred in a series of distributed bits of the ECOC codes. This results in a better generalization of the proposed CNN-ECOC method.

7.2.2 Experiment Two

The purpose of the second experiment is to show the advantage of using selected sub-spaces from the whole feature space in conjunction with ECOC to improve classification performance. There are two ways to choose a sub-space from the original feature space: (1) the feature sub-space is selected by using a feature selection method: kNN-ECOC-FS (Section 4.3.4); (2) the feature sub-space is randomly taken: kNN-ECOC-RS (Section 4.3.4) and Multiple Feature Subsets (MFS, including both MFS-WITH-REPLACEMENT and MFS-NO-REPLACEMENT, Section 4.3.1). The nearest neighbor classifier (NN) is again used here as the base for comparison. The comparison of error rate, bias and variance is shown in Figures 7.9 — 7.14 as a function of the number of spectral bands. In the following, two main observations from the figures are explained.

- The effectiveness of Multiple Feature Subsets (MFS) is shown on these experimental results. There are two parameters to be set for the MFS algorithm: i.e. the size of the feature subset and the number of classifiers to combine. In our experiment, we used Bay's [55] setting value on the subset size: the subset size was set on the basis of cross-validation accuracy estimates, and then ten evenly spaced intervals over the size of the original feature set were evaluated. While Bay sets the number of classifiers to 100 we choose this parameter to 10 in order to save computation time. From the plot, one can observe that there is not much difference between the MFS with replacement and MFS without replacement in terms of error rate, bias and variance on our dataset.

- For the method we presented: the kNN-ECOC-RS (RS means Randomly Selected features), there are many ways to fix the number of features to be selected for every bit of ECOC code. In our experiments, we choose this parameter as half of all features. The performance of kNN-ECOC-FS (FS means Feature Selection) method by Aha and Bankert [43] depends largely on the feature selection method embedded. In this experiment we choose a wrapper feature selection method (Section 5.4.2). It is a simple Sequential Forward Selection method (SFS) [96], and we use the NN classifier as the induction algorithm with the SFS. The SFS algorithm selects successive features with respect to the current set of features.

  In Figure 7.9 the error rate of kNN-ECOC-FS becomes larger than kNN-ECOC-RS. In the mean time, the kNN-ECOC-FS has both a high bias (especially in Figure 7.11) and a high variance (especially in Figure 7.13). High bias indicates that kNN-ECOC-FS (mainly due to the adoption of SFS as the feature

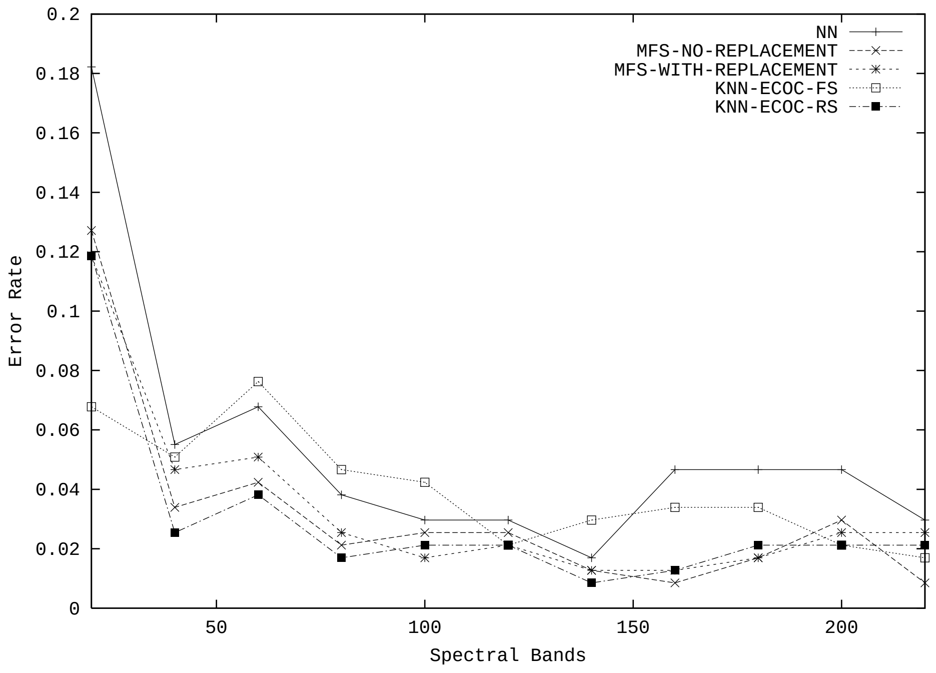


Figure 7.9: Comparison of error rate between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, *k*NN-ECOC-FS, *k*NN-ECOC-RS. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands. *k*=1.

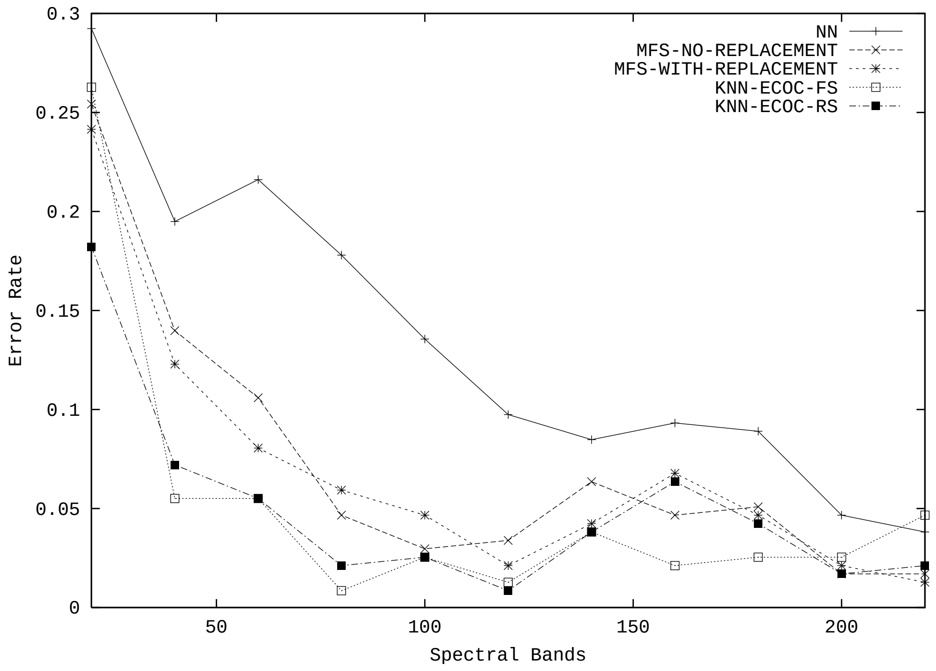


Figure 7.10: Comparison of error rate between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, *k*NN-ECOC-FS, *k*NN-ECOC-RS. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands. *k*=1.

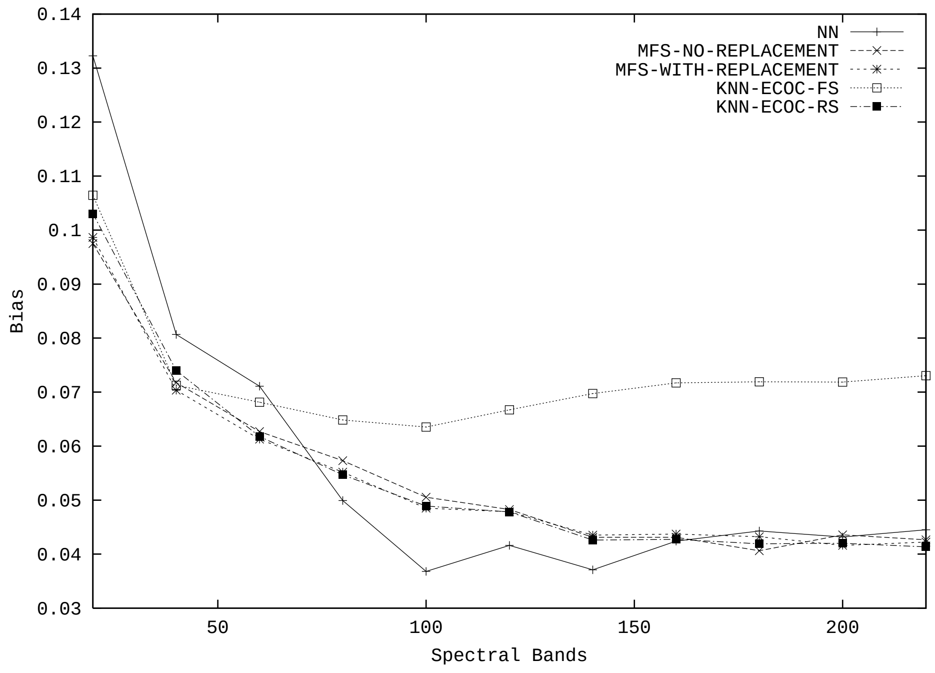


Figure 7.11: Comparison of bias between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, kNN-ECOC-FS, kNN-ECOC-RS. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands. k=1.

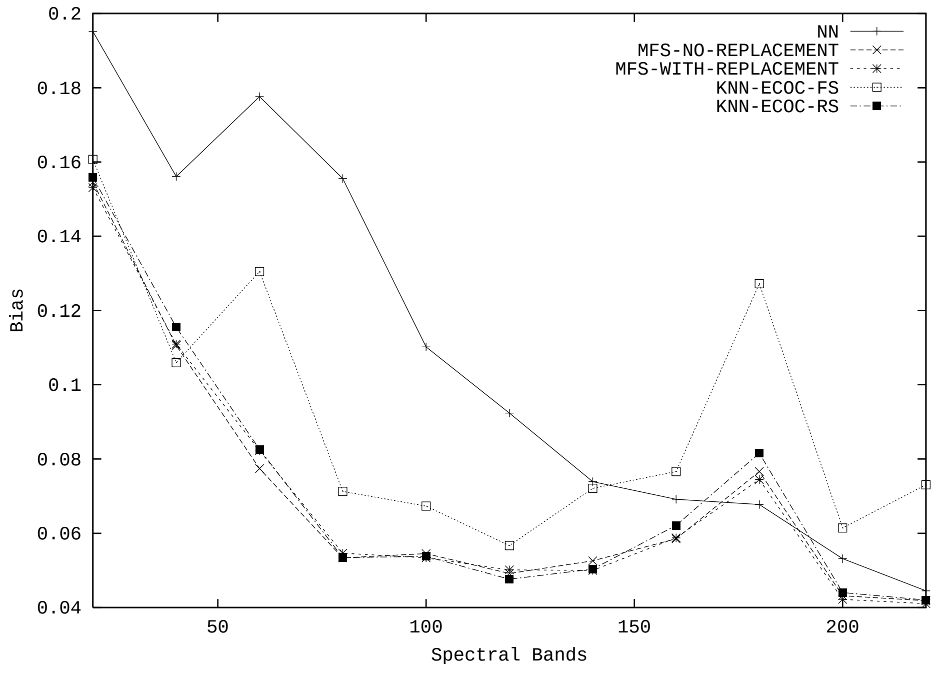


Figure 7.12: Comparison of bias between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, kNN-ECOC-FS, kNN-ECOC-RS. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands. k=1.

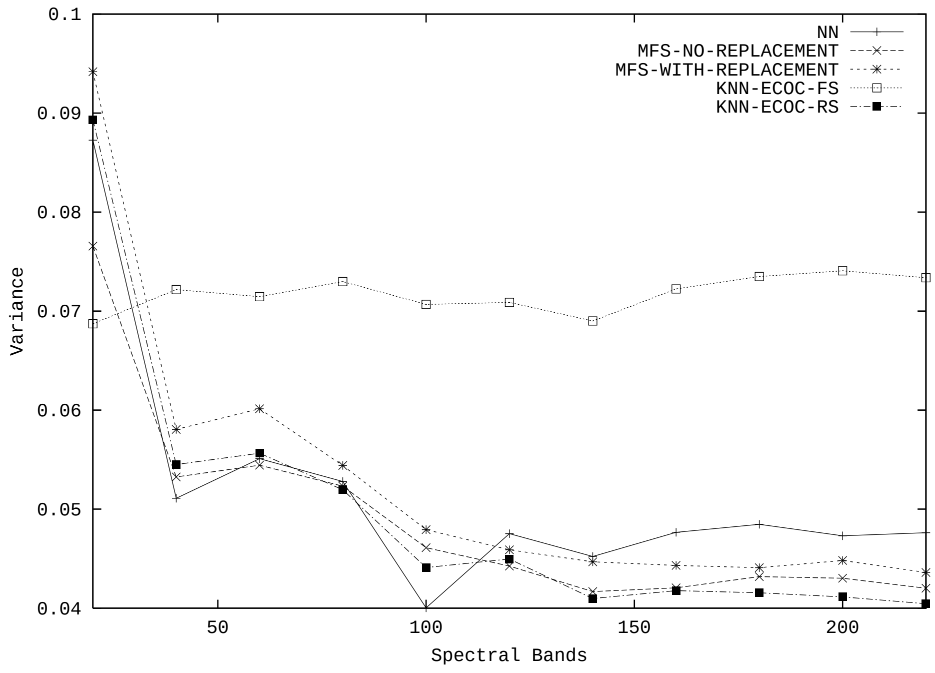


Figure 7.13: Comparison of variance between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, *k*NN-ECOC-FS, *k*NN-ECOC-RS. Results are shown respectively for the first 20 bands, the first 40 bands, ..., 220 bands. *k*=1.

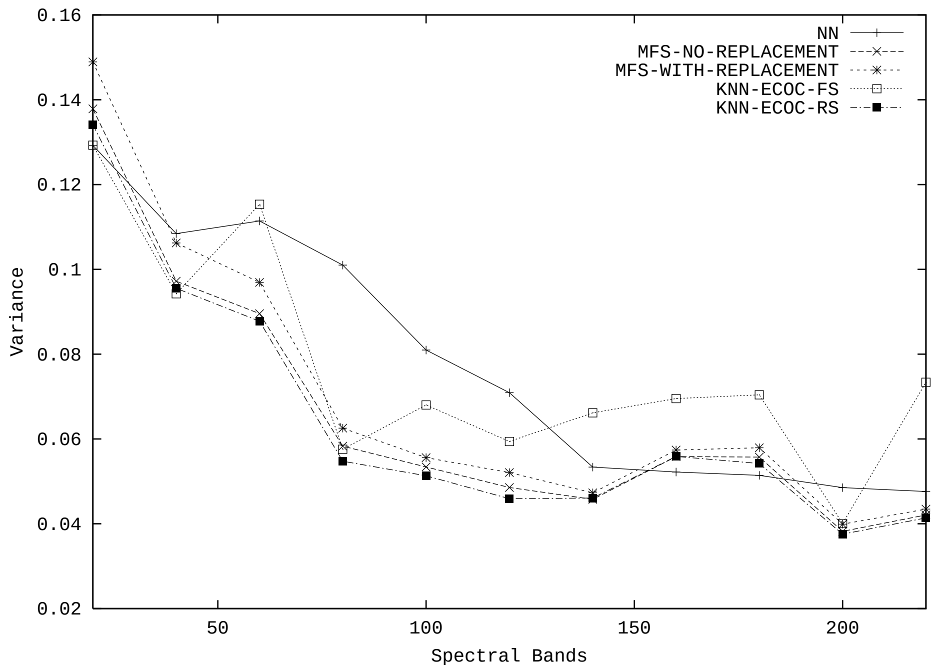


Figure 7.14: Comparison of variance between NN, MFS-NO-REPLACEMENT, MFS-WITH-REPLACEMENT, *k*NN-ECOC-FS, *k*NN-ECOC-RS. Results are shown respectively for the last 20 bands, the last 40 bands, ..., 220 bands. *k*=1.

selection method to be combined with) has a high systematic error, while high variance suggests it has a poor generalization. This is because once a feature is included by SFS in the feature set, it has no mechanism for delecting it from the feature set, even at a later stage when more features have added and this feature becomes superfluous. In other words, the feature sets are nested. For example, two best features chosen by SFS are not necessarily the best two. One remedy for the SFS is to adopt a mechanism that can delete the feature when later finding it to be irrelevant or redundant. For example, by adopting the corresponding *floating* version of the Sequential Forward Search — SFFS [217], the kNN-ECOC-FS method will certainly perform very well, but a corresponding computational cost is also expected (the computational cost of SFFS is pretty high, for this, see Section 7.4.2). The poor generalization of kNN-ECOC-FS (with SFS) causes its performance to be uncertain, i.e. the actual performance of kNN-ECOC-FS (with SFS) depends on the individual dataset concerned. This can be seen from Figure 7.9 — it can even be challenged by simply adopting random sampling from the original feature space. Of course, the most important contribution to this phenomenon is due to the power of ECOC technique. While by a simple random sampling of the orignal feature space, one can only by chance get the classification performance improved. By using ECOC in conjunction with a random sampling, errors occurring by chance in a series of distributed bits of the ECOC codes (for details about ECOC, see Section 4.2) are smoothly alleviated, and consequently, the entire performance is improved.

The computational cost also increases with the the number of nearest neighbors, for this we chose k=1 in this experiment for the sake of computational efficiency.

7.2.3 Experiment Three

Other methods, mainly Skalak's composite NN architecture (Section 4.3.3), are evaluated in the third experiment. The experiment compares 1) the k nearest neighbor classifier (kNN), 2) the fuzzy k nearest neighbor classifier (fuzzy kNN), 3) Skalak's composite NN architecture, 4) Skalak's composite NN architecture with ECOC, 5) a decision tree C4.5, and 6) Naive-Bayes (NB) [76]. The error rate, bias and variance are plotted in Figures 7.15 — 7.20 in function of the number of spectral bands, starting from 20 bands to the full 220 spectral bands: i.e. the first/last 20 bands, the first/last 40 bands, the first/last 60 bands, ..., the full 220 bands.

The k nearest neighbor classifier is used here as the base for comparison, and the parameter k was chosen to 5 using cross-validation. The justification for also comparing with C4.5 is based on the fact that the level-1 combination algorithm in Figure 4.5 of Chapter IV is ID3, and the C4.5 is a further extension of ID3.

Major conclusions concerning this experiment are summarized as follows:

- The Naive-Bayes classifier performs poorly on our dataset. The rationale for choosing to include Naive-Bayes as well is that some researchers [168] noted that the accuracy of the very simple NB classifier is superior over that of C4.5 in some real world datasets. We would like to take advantage of this experiment to verify if this observation also holds for our dataset. Unfortunately one can conclude from the plots that their observation didn't show in our dataset.

- One can see that the classification performance of fuzzy kNN very slightly outperforms its counter-part — the crisp kNN, in the very low spectral bands, but their classification performances coincide when the dimensionality increases. Both bias and variance of fuzzy kNN are a little smaller than those of the crisp

kNN.

- From the figures, it is also concluded that Skalak's composite NN architecture gives the best prediction rate. In particular, its low bias indicates that this architecture has a small systematic error and its low variance suggests it has a better generalization, henceforth, the tendency that its classification performance depends largely on the individual dataset concerned is averted. While we have hypothesized that further using ECOC technique probably improves the classification performance of Skalak's composite NN architecture (See last paragraph of Section 4.3.3), this does not show in the plots. In fact, Skalak's composite NN architecture with and without ECOC completely coincide. This confirms Dietterich's [39] arguments that if the representation for the feature input to the decision tree(C4.5/ID3) is ambiguous, then (C4.5/ID3) will have difficulty in finding a good decision tree, and ECOC will not be able to overcome this problem.

  The purpose of Skalak's composite architecture is to improve the performance of the base classifier, in this example, a kNN rule was used in this architecture as the base classifier. Therefore one may hypothesize that if we instead use the fuzzy version of the kNN rule as the base classifier, then its performance could be improved, indeed this hypothesis has already been confirmed [284]. One may further question that if and how this architecture would improve an ensembled learning (instead of a non-ensembled single learning like kNN). This could be an interesting open topic.

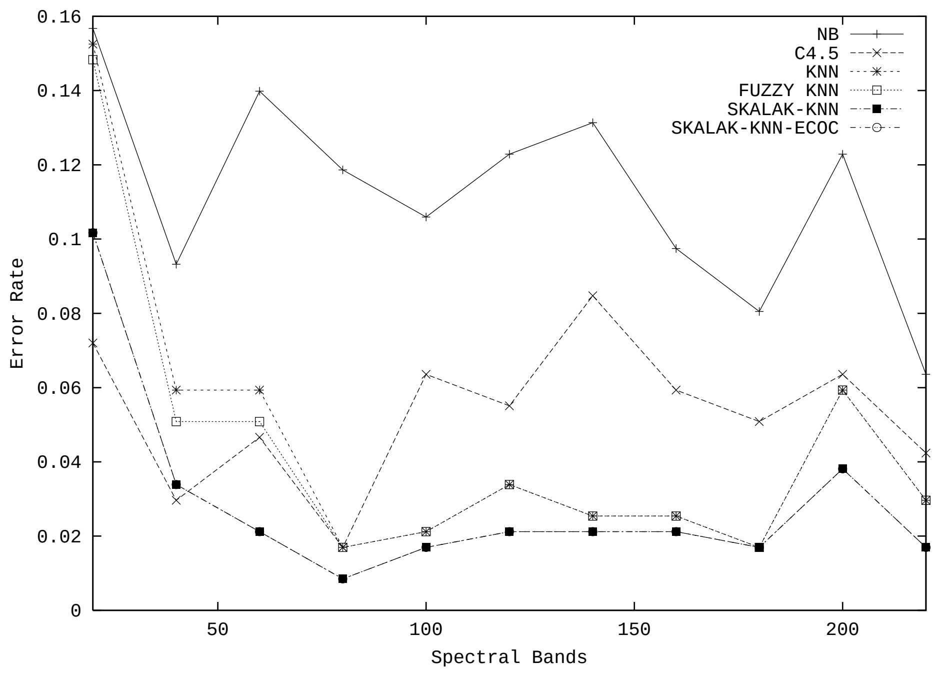


Figure 7.15: Comparison of error rate between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands. k=5.

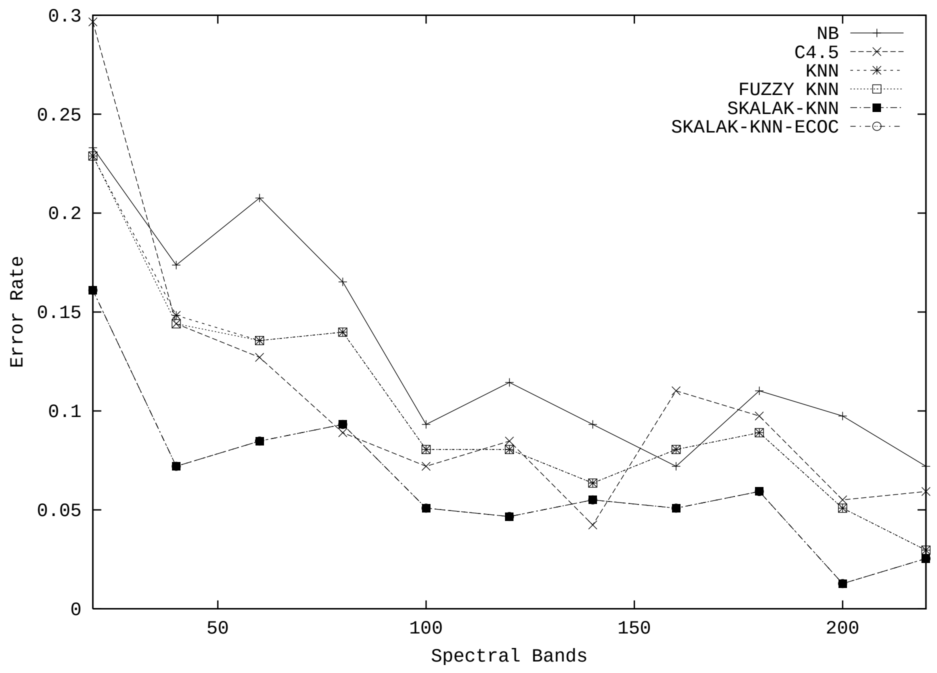


Figure 7.16: Comparison of error rate between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands. k=5.

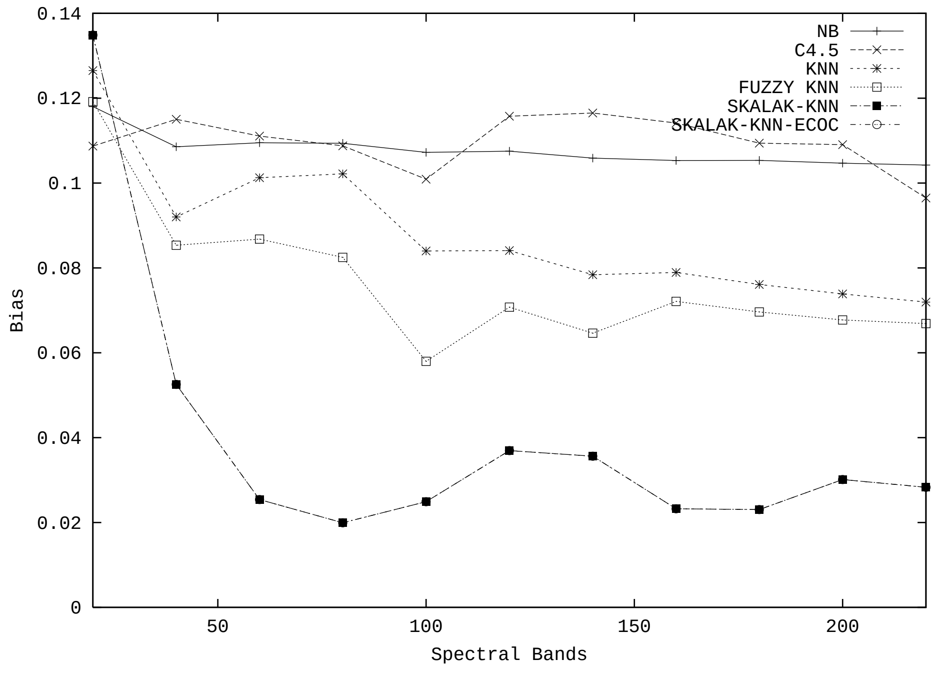


Figure 7.17: Comparison of bias between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands. k=5.

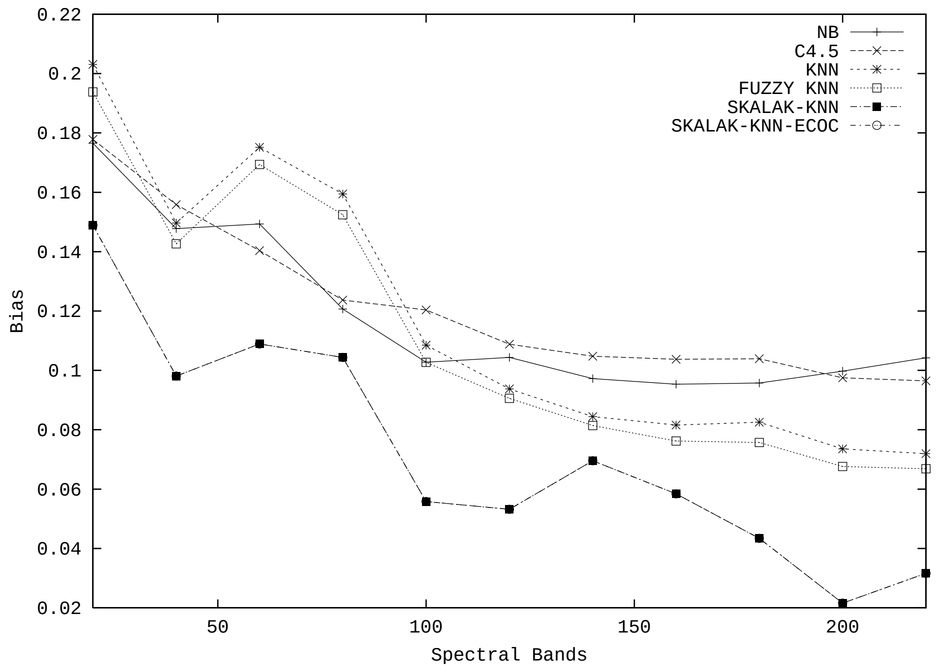


Figure 7.18: Comparison of bias between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands. k=5.

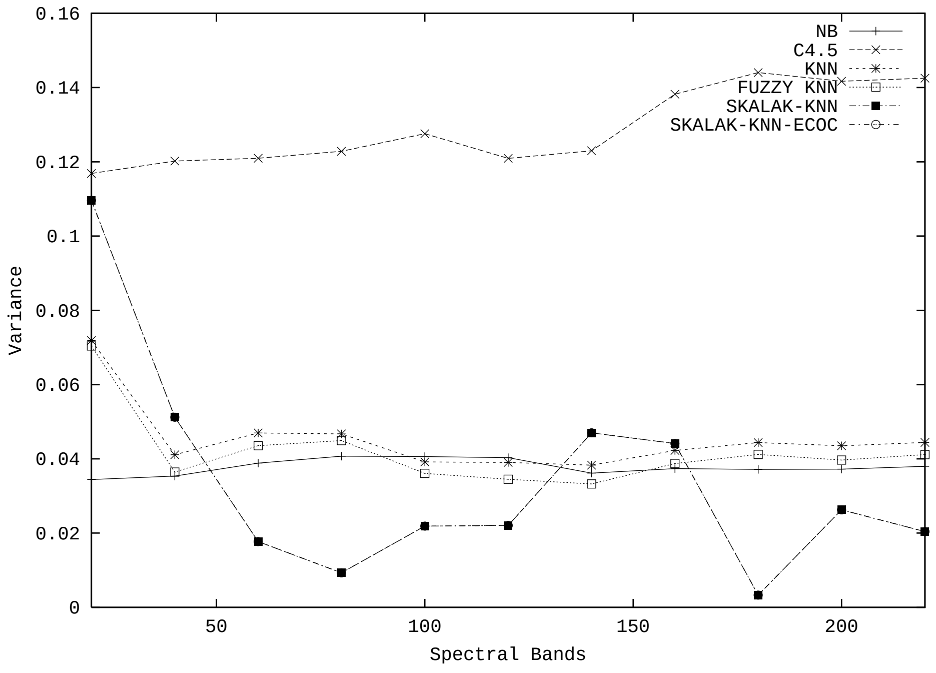


Figure 7.19: Comparison of variance between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands. k=5.

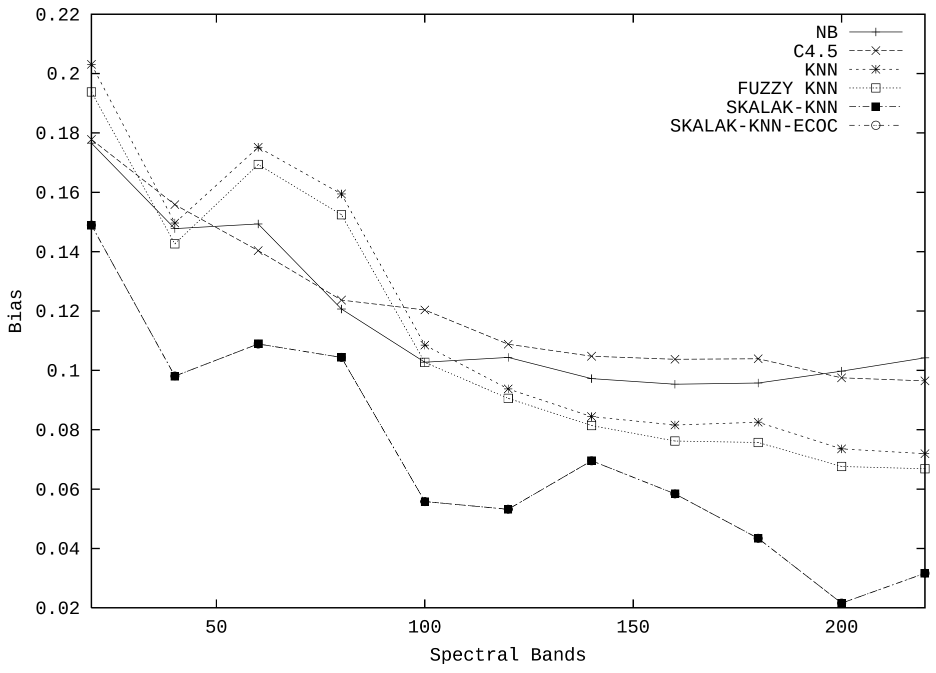


Figure 7.20: Comparison of variance between Naive Bayes (NB), a decision tree C4.5, kNN, Fuzzy kNN, Skalak's kNN, Skalak's kNN with ECOC codes. Results are shown respectively for the last 20 bands, the last 40 bands, ..., the full 220 bands. k=5.

7.3 Summary and Remarks on Nearest Neighbor Classifier Ensembles

In the above section together with Chapter IV, the ensemble methods for nearest neighbor classifiers are reviewed and studied on hyperspectral remote sensing data. Rather than presenting a more complicated classifier or classification architecture, a general study was given.

For an example of complicated classifiers, the fuzzy idea plus the condensed idea may give rise to a so-called "condensed fuzzy nearest neighbor classifier". For a more complicated classification architecture: in the first layer of Fig 4.5, the condensed fuzzy nearest neighbor classifier could be applied to either the base classifier or the complementary classifier, thus producing different kinds of combinations.

As for the voting on CNN, rather than by voting on the condensed nearest neighbor classifiers only, the Multiple Feature Subsets (MFS) [55] idea can be further applied to all the condensed component nearest neighbor classifiers. This could be a dynamic voting over multiple condensed nearest neighbors: both the feature subset number and the feature subset itself become dynamic. An advanced sampling technique, e.g. the weighted random sampling [209] [195], could be instead applied to the MFS method to improve the performance. Besides the simple voting and the weighted voting as used in several ensemble methods, the weighted majority described in Section 2.3.1 of Chapter II could also be applied. In order to further decrease the number of design patterns, the Edit algorithm [96] [129] could be used before condensing the design patterns , but probably at the expense of performance deterioration.

All of these pointed modifications can be a series of further research. For most of the nearest neighbor ensembles, there exist various kinds of combinations between their individual component classifiers, of which combination is the best one is not studied in this dissertation. For such a study, the Orthogonal Experimental Design (OED) method [205] [134] based on orthogonal arrays (OAs) and factor analysis was suggested to find how several simultaneously changing factors affect the classification performance of the entire classification architecture.

7.4 Experiments of Genetic Feature Selector

Due to the problem of long running times using Java code, the implementation of feature selection for the hyperspectral data was turned to C++[2] with the use of the Genetic Algorithms library — *GAlib* [9] from MIT. The random shuffle implementation was borrowed from the repository of free, peer-reviewed C++ libraries — *Boost* [17]. The genetic feature selection tested in the following two experiments belongs to the type of Aggregation Selection with Parameter Variation (Section 6.3.1), the scheme of which is based on the seminal work by Siedlecki and Sklansky [246].

7.4.1 Experiment One

In the first experiment, the genetic feature selection technique is evaluated. For this, the minimal number of obtained features is plotted in function of the number of generations. The parameters t and ξ (see Section 6.3.1 for detail) are set so that the classification error is about 10 %. In figure 7.21, experimental results are displayed for a 3-class problem (classes 2, 3 and 8), with around 100 data points for each class. Several experiments starting with different numbers of bands are conducted. On the plot, the numbers of bands are 50, 100, 150 and 220 respectively, i.e. the first 50 bands, 100 bands, 150 bands of the dataset and the full 220-band data. The population size was 100. The crossover rate usually assumes high values, close or

[2]For the Integrated Development Environment (IDE) for C++, I used the freepackage — the Open Edition of *Kylix* from Borland [18]. Kylix is a versatile C++ development tool with the powerful Borland *Component Library for Cross-platform* (CLX).

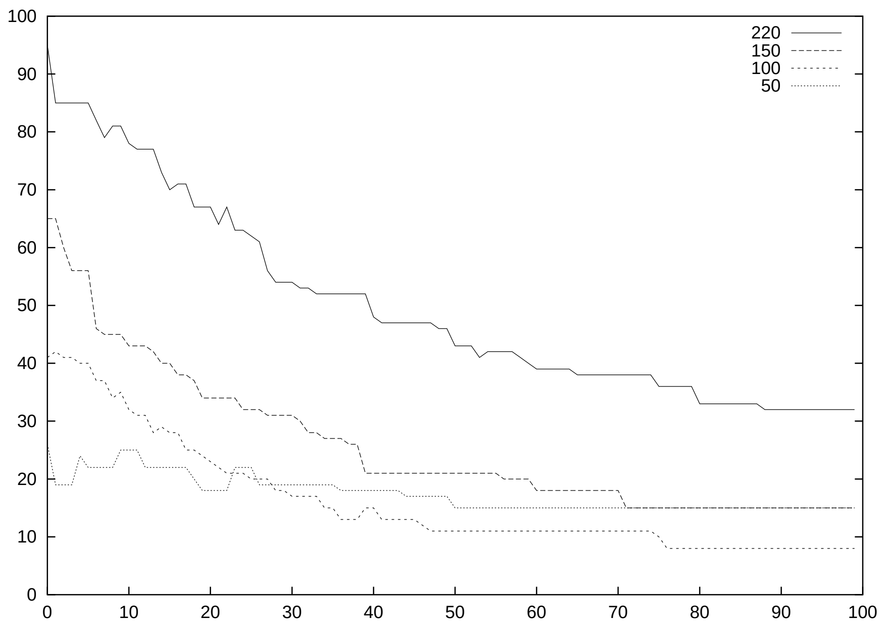


Figure 7.21: Number of obtained features versus number of generations for the genetic feature selection on 50, 100, 150 and 220 bands respectively.

equal to one, while the mutation rate is typically small [246]. The crossover rate is high to allow to produce an offspring that is more optimal than its parents. A 100 % crossover rate however would disrupt any good solution. In our experiment, crossover and mutation rates were set to 90 % and 1 % respectively. For classification, the fuzzy 5NN algorithm was applied. From the plot, one can observe that the number of obtained features decreases with the number of generations. The classification errors were about constant over all the curves (about 10%). In the beginning the reduction is very fast, but after about 50 generations convergence becomes slow.

7.4.2 Experiment Two

There are somewhat controversial reports on the performance comparison between the genetic selection method and the floating search method in the literature. The work of Ferri *et al.* [109] points out that GA and Sequential Floating Forward Search (SFFS) are comparable for moderate size, but that the performance of GA's deteriorates as the dimensionality increases. Jain and Zongker [147] report that the GA approach seems to have a tendency towards premature convergence. In contrast to these reports, the recent work by Kudo and Sklansky [178] suggests that SFFS is suitable for problems with small or medium dimensionality. For very high dimensionality, they tend to favor the use of the GA approach.

The purpose of our experiment is two-fold: (i) To undertake the first study that uses an ensembled learning scheme as the induction algorithm during the selection process with genetic algorithms and demonstrate its superiority over its non-ensembled learning counterpart for genetic feature selection. (ii) To have a pragmatic view of both genetic search and sequential floating forward search on hyperspectral remote sensing data.

Due to the costly time complexity of the *backward* version of sequential floating search (for this, see e.g. footnote 4 in Chapter V), we choose instead the *forward* version of sequential floating search to compare with.

When using genetic algorithms, the first difficult task is how to set up the parameters for GAs. Unfortunately, there is no unified guidance for this. For the sake of simplicity in this experiment, based on our experience, we set the population size as 100, the generation size 50, the crossover probability 0.9, the mutation probability 0.01. The comparison scheme is done as follows. First we run the genetic search and get the number of selected features. We then set the sequential floating forward search to get the same number of selected features as from the genetic search. The comparison was done on error rate and the number of evaluations used respectively.

In order to demonstrate the superiority of the ensembled learning algorithm, we hybridized the feature selection algorithms (both genetic search and sequential floating forward search) with an ensembled learning — the CNN-ECOC and its corre-

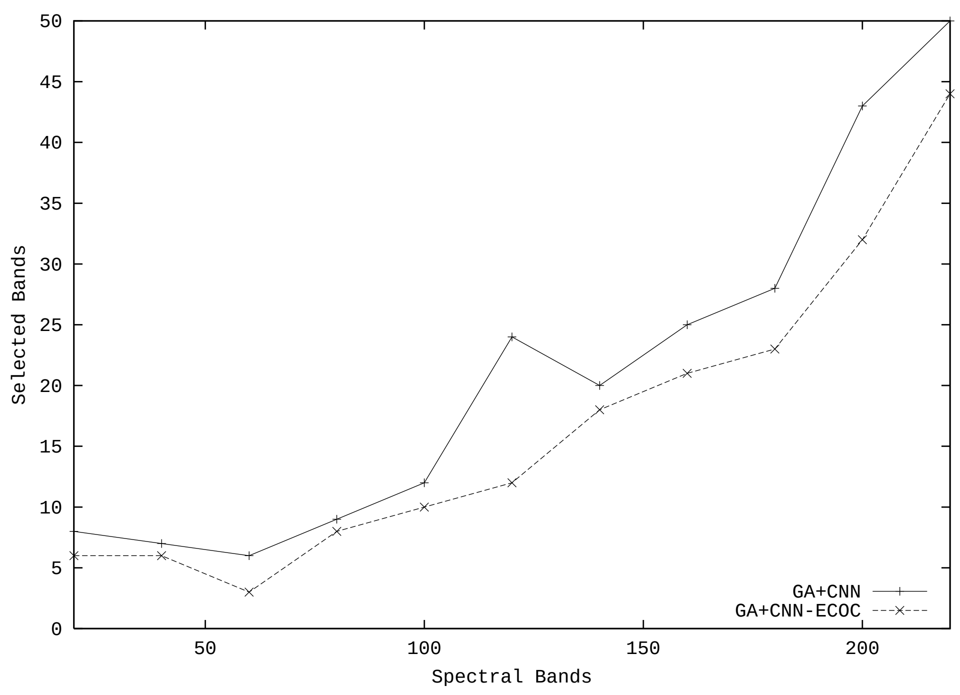


Figure 7.22: Number of obtained spectral bands by running GA with CNN and GA with CNN-ECOC respectively. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

sponding non-ensembled counterpart — the CNN algorithm respectively.

Figure 7.22 shows the numbers of selected features by a run of GA with CNN and GA with CNN-ECOC respectively. One observation from this figure is that, the number of obtained features by GA with CNN-ECOC is, generally speaking, smaller than that by GA with CNN. This suggests one potential advantage of using an ensembled learning: the smaller the number of selected features is, the more simplified model it may result in.

The classification performance by the different search methods are plotted respectively as a function of the numbers of spectral bands in Figure 7.23. One can observe a systematic classification performance improvement (up to around 5%) by the use of CNN-ECOC over CNN in conjunction with genetic algorithms. For low spectral bands, the difference of their classification performance is small, but when the dimensionality exceeds 140, the difference tends to grow. This demonstrates the superiority of ensembled learning over non-ensembled learning for the use with genetic algorithms. Therefore we suggest that when using a specific classifier with genetic algorithms to do pre-processing, one should extend the choice of classifiers by taking into account the use of the corresponding ensembled counterpart classification algorithms.

At a first glance, one may see that the classification performance of (SFFS+CNN-ECOC) slightly dominates that of (SFFS+CNN) over all spectral bands, but the difference between them is not significant (the difference is less than 2%).

The difference of the classification performance between genetic search and SFFS is up to 10 % between spectral bands of 80 — 140, but using CNN-ECOC with GAs reduces this difference. For example, when the dimensionality tends to be higher than 140, the difference becomes smaller (less than 5%). Another remedy to this is to use an *optimal* parameter settings of GA, which was not addressed in this dissertation.

In Figure 7.24, we didn't compare directly the execution time used to complete the search task by both genetic search and sequential floating search, but instead we compared the number of intermediate subset evaluations, since execution time usually varies from one implementation to another. For example, the GAlib [9] we used is a highly optimized library, few people can write the implementation code of the sequential floating search in that optimized manner. The use of GAlib with a higher number of subset evaluations could still need less running time than a not-highly-optimized (or even a poorly optimized) implementation of sequential floating search with a smaller number of subset evaluations, so directly comparing the execution time alone is not fair. The number of intermediate subset evaluations reflects

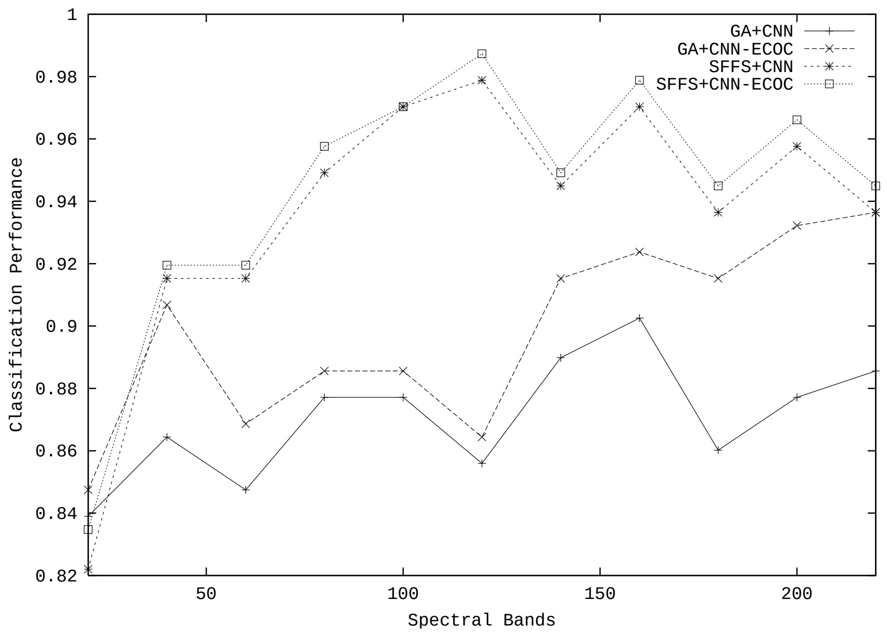


Figure 7.23: Classification performance of GA with CNN, GA with CNN-ECOC, SFS with CNN and SFS with CNN-ECOC. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

the time complexity: the larger the number of evaluations is, the higher the time complexity of the search algorithm.

From Figure 7.24, one can see that the number of evaluations of the genetic search grows very slowly with the dimensionality, but that of the sequential floating search grows very quickly with the dimensionality. The time complexity of floating search could be 3 to 5 times than that of genetic search in this case (with the current parameter settings of GAs). When the dimensionality exceeds 180, the number of evaluations by (SFFS+CNN-ECOC) is becoming much larger than that by (SFFS+CNN).

7.5 Summary and Remarks on Genetic Feature Selector

In the above section together with chapter VI, the use of the ensembled learning with genetic algorithms to perform feature selection is initiated and applied to high

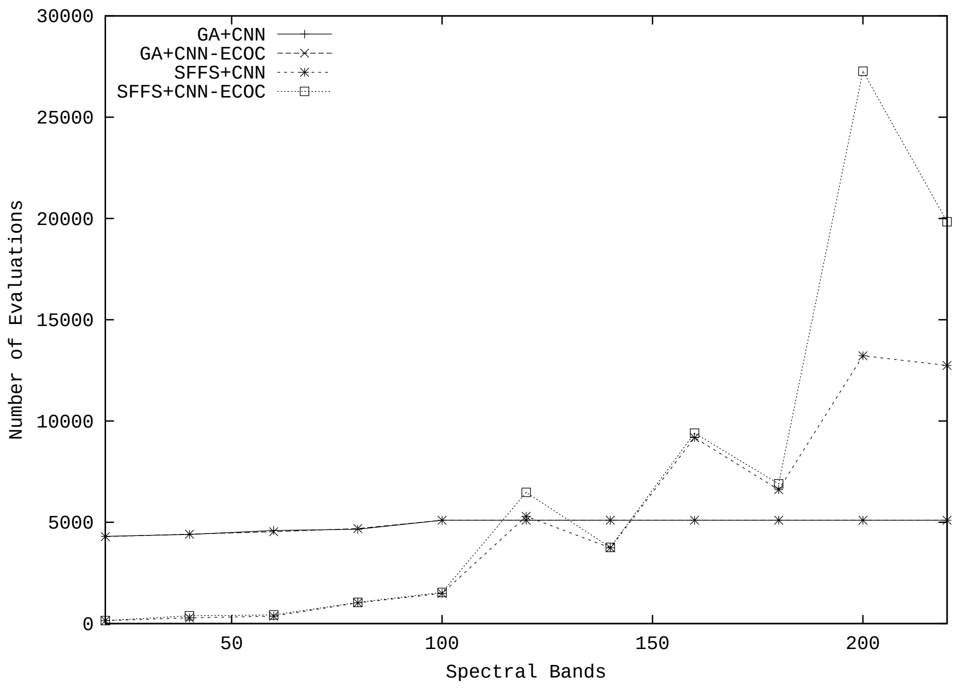


Figure 7.24: Comparison of number of evaluations between GA with CNN, GA with CNN-ECOC, SFS with CNN and SFS with CNN-ECOC. Results are shown respectively for the first 20 bands, the first 40 bands, ..., the full 220 bands.

dimensional remote sensing data. We demonstrated that its use not only reduces the number of selected features, but also improves the classification performance. By comparing genetic feature selection and sequential floating forward selection, we can conclude that the sequential floating forward feature selection is suited for small and medium sized databases, but for larger databases, one should definitely take into account the use of GAs in terms of both classification performance and (*especially*) running time.

Due to the inherent parallel nature of genetic algorithms, there are many parallel genetic algorithms ready for us to use [74] while there is no parallel version of floating search at this time. Although we didn't do experiments on parallel GAs, we believe they can further improve the computation efficiency, especially when dimensionality increases. In terms of time complexity, the floating search method is then no longer a rival to genetic search. But for low dimensionality, the merit of floating search is straightforward: it only needs a small number of subset evaluations to finish the search. From this point of view, our experimental results support the claim in the work by Kudo and Sklansky [178]: the GA approach is well suited for time-critical tasks.

In order to rival the parallel GA approach *to some extent*, the floating forward search method needs at least to have a parallel version, especially in the case of high dimensional data. Another practical advantage of genetic search over floating forward search is that it is very robust in the sense that its parameters are easily tunable, in order to meet one's computational needs.

CHAPTER VIII

Conclusion and Further Work

8.1 Conclusion

In this dissertation, two important aspects of pattern recognition are studied on hyperspectral remote sensing data: classifier ensemble problems, in particular nearest neighbor classifier ensembles, and feature selection problems, in particular genetic feature selectors.

The fact that the research on feature selection dates back to the 70's, not only demonstrates that this topic is a traditional and fundamental problem, but also suggests it still remains, despite more than three decades' research efforts by numerous researchers, a very challenging task. Therefore the idea of feature selection is very old, but its application to hyperspectral remote sensing data — ”band selection”, is rather new, mainly due to the recent advances in sensor technology. The adoption of feature selection to hyperspectral data was in the hope that it alleviates the so-called Hughes phenomenon [139], that is, the classification performance of hyperspectral data improves up to a limited point as additional features are added, and then deteriorates.

As already pointed out in Chapter II, the idea that aggregating the opinions of a committee of experts to improve accuracy is not new, rather it has many real world applications which might reflect direct connections to our daily life. But the idea of applying aggregation to classification algorithms is rather new. While most of the current research on classifier ensembles focuses on the general principles and methodologies, in this dissertation the specific nearest neighbor classifier ensembles were focused on.

In summary, the roadmap of the research conducted in this dissertation is the following:

First we studied the classifier ensembles in general. We then presented a method called CNN-ECOC, as our first main contribution, which takes advantage of the dynamic nature of the Condensed Nearest Neighbors (CNN) algorithm in conjunction with the technique of Error Correcting Output Codes (ECOC). The ECOC technique is a distributed scheme which decomposes *a multi-class classification problem* into a series of distributed *two-class problems*. By adopting the ECOC with CNN, we demonstrated that this method further improves the classification performance and at the same time decreases the storage requirements. Another variant called kNN-ECOC-RS, which takes advantage of the dynamic nature of randomly selected features in conjunction with the ECOC was suggested as a by-product.

As our second main contribution, we initiated and advocated the idea of using ensembled learning in conjunction with genetic algorithms to perform feature selection. As an example of this, we demonstrated the superiority of the proposed ensembled learning (CNN-ECOC) with genetic algorithms over its non-ensembled counterpart (CNN).

8.2 Further work

1. Part of this thesis work deals with the feature selection problem for hyperspectral remote sensing data, but the idea of band selection was under criticism from leading scientist in the remote sensing community, arguing that ”to pick up a subset of bands and completely ignore the rest may ignore useful diagnostic”. Therefore further research is needed to investigate e.g. if feature extraction would be more useful than feature selection for hyperspectral data.

2. Another evolutionary feature selector, a multi-layer neural network feature selector, which uses the scheme of EDA (Estimation of Distribution Algorithm) [206] as the search engine in conjuntion with a multi-layer neural network [57] (used as the models for representing the probability distribution of a set of candidate solutions), is suggested as another topic of research.

3. The use of a population (even without using crossover) can by itself be advantageous for function optimization, as has been investigated theoretically [152]. A theoretical study on the use of a population alone which uses neither crossover nor mutation, may be considered as an interesting subject.

NEDERLANDSE SAMENVATTING

This is dutch summary. It is being translated and will appear on the final print-out version.

LIST OF PUBLICATIONS

- **Shixin Yu**, Steve De Backer, Paul Scheunders *Genetic Feature Selection Combined with Fuzzy kNN for Hyperspectral Satellite Imagery.* Proc. of IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2000), 24-28, July, 2000, Honolulu, Hawaii, U.S.A. Vol. 2, pp. 702 -704.

- **Shixin Yu**, Steve De Backer, Paul Scheunders. *Genetic Feature Selection Combined with Composite Fuzzy Nearest Neighbor Classifiers for High-dimensional Remote Sensing Data.* Proc. of IEEE Interntemal Conference on Systems, Man and Cybernetics, 8-11, October 2000, Nashville, TN, USA, pp. 1912-1916.

- **Shixin Yu**, Paul Scheunders. *A Fuzzy Markov Chain Approach to Feature Selection for High-Dimensional Remote Sensing Data*, Proc. of IEEE International Geoscience and Remote Sensing Symposium, 9-13, July, 2001, Sydney, Australia. Vol. 7, pp. 3306-3308.

- **Shixin Yu**, Paul Scheunders. *Feature Selection for High-Dimensional Remote Sensing Data by Maximum Entropy based Optimization*, Proc. of IEEE International Geoscience and Remote Sensing Symposium, 9-13, July, 2001, Sydney, Australia. Vol. 7, pp. 3303-3305.

- **Shixin Yu**, Steven De Backer, Paul Scheunders. *Genetic Feature Selection Combined wth Composite Fuzzy Nearest Neighbor Classifiers for Hyperspectral*

Satellite Imagery. Pattern Recognition Letters, 23, 183-190, 2002.

- **Shixin Yu**, Paul Scheunders. *On Combining Nearest Neighbor Classifiers: An Empirical Study on Hyperspectral Remote Sensing Data.* submitted to Pattern Recognition Letters, 2002.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] ftp://ftp.cs.orst.edu/put/tgd/programs/ecoc-codes.tar.gz. 41

[2] ftp://ftp.uwasa.fi/cs/report94-1/. 58

[3] http://dynamo.ecn.purdue.edu/ ˜biehl/multispec/documentation.html. 2

[4] http://dynamo.ecn.purdue.edu/˜landgreb/publications.html. 5

[5] http://evonet.dcs.napier.ac.uk/. EvoNet - the European Network of Excellence in Evolutionary Computing. 57

[6] http://home.ptd.net/˜olcay/feature-selection.html. Online Bibliography on Feature Selection. 44

[7] http://iris.usc.edu/Vision-Notes/bibliography/contents.html. Annotated Computer Vision Bibliography, check 14.1.4.2 Multiple Classifiers, Combining Classifiers, Combinations. 11

[8] http://iris.usc.edu/Vision-Notes/bibliography/contents.html. Annotated Computer Vision Bibliography, check 14.1.3 Feature Selection in Pattern Recognition or Clustering. 44

[9] http://lancet.mit.edu/ga/. GAlib - the Genetic Algorithm library. 84, 88

[10] http://makalu.jpl.nasa.gov/aviris.html. Main Homepage of AVIRIS at JPL of NASA. 1

[11] http://surf.de.uu.net/encore/www/. The Hitch-Hiker's Guide to Evolutionary Computation. 57

[12] http://tilde-hoschek.home.cern.ch/˜hoschek/colt/index.htm. COLT: A Java Package for High Performance Computing. 66

[13] http://www-fp.mcs.anl.gov/otc/guide/optweb/index.html. 49

[14] http://www.aaai.org. AAAI - American Association for Artificial Intelligence. 5

[15] http://www.aic.nrl.navy.mil/galist/. The Genetic Algorithm Archive. 58

[16] http://www.boosting.org. Online Resources on Boosting Research. 19, 30

[17] http://www.boost.org. Boost - the repository of free, peer-reviewed C++ libraries. 84

[18] http://www.borland.com/kylix/. 84

[19] http://www.cs.waikato.ac.nz/˜ml/weka/. WEKA: A Java Machine Learning Package. 66

[20] http://www.eclipse.org. 66

[21] http://www.genetic-programming.org/. Genetic Programming Online. 57

[22] http://www.inf.fu-berlin.de/˜behnke/papers/nn98/node8.html. 14

[23] http://www.isgec.org. The International Society for Genetic and Evolutionary Computation (ISGEC). 57

[24] http://www.jeo.org. Evolutionary Optimization - An International Journal on the Internet. 57

[25] http://www.kernel-machines.org. Primary Resources on the Kernel Machines Related Research. 11

[26] http://www.lania.mx/~ccoello/EMOO/. Repository on Evolutionary Multiobjective Optimization, mirror sites also at: www.jeo.org/emo/ and http://delta.cs.cinvestav.mx/~ccoello/EMOO/. 64

[27] http://www.mlnet.org. MLnet - The Machine Learning Network Online Information Service. 5

[28] http://www.ncsa.uiuc.edu/Apps/CMP/RNG/www-rng.html. Random Numbers on the Web. 66

[29] http://www.optimization-online.org/. Optimization Online - an eprint site for the optimization community. 49

[30] http://www.ph.tn.tudelft.nl/PRInfo/books.html. A list of research monographs on pattern recognition. 3

[31] http://www.public.asu.edu/~huanliu/fsbook/appendixa.html. 44

[32] http://www.recursive-partitioning.com. Online Bibliography on Recursive Partitioning. 23

[33] http://www.rsinc.com/envi. ENVI - the Environment for Visualizing Images, remote sensing software of Research Systems Inc. 26

[34] http://www.rulequest.com/see5-info.html. 56

[35] http://wwws.sun.com/software/sundev/jde/. 66

[36] http://www.trw.com. 2

[37] http://www.vtt.fi/tte/research/tte1/tte14/virtual/. Remote Sensng WWW Virtual Library. 1

[38] Pattern recognition group at Delft University of Technology. v

[39] Personal communication, 2001. 79

[40] G. Abousleman, M. Marcellin, and B. Hunt. Hyperspectral image compression using entropy-constrained predictive trellis coded quantization. *IEEE-IP*, 6(4):566–573, Apr. 1997. 2

[41] D. Aha and R. Banket. A comparative evaluation of sequential feature selection algorithms. In *Proc. of the 5th International Workshop on Artificial Intelligence and statistics*, pages 1–7, Menlo Park, CA, 1994. AAAI. 51, 53, 55

[42] David W. Aha. *Lazy Learning*. Kluwer Academic Publishers, Dordrecht, Jun. 1997. 27

[43] D.W. Aha and R.L. Bankert. Cloud classification using error-correcting output codes. Technical Report AIC-96-024, NCARAI, 1996. 32, 39, 40, 66, 73

[44] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991. 27, 35, 39

[45] H. Almuallim and Dietterich. T.G. Learning with many irrelevant features. In *Proc. of the 9th National Conference on Artificial Intelligence*, pages 547–552, San Jose, CA, 1991. AAAI Press. 50, 53

[46] E. Alpaydin. *Neural Models of Incremental Supervised and Unsupervised Learning.* PhD thesis, Department d'Informatique, Ecole Polutechnique Fédérale de Lausanne, Lausanne, Switzerland, 1990. No. 869. 35

[47] E. Alpaydin. Voting over multiple condensed nearest neighbors. *Artificial Intelligence Review*, 11(1-5):115–132, 1997. 30, 32, 35, 68

[48] T. Back. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996. 57

[49] T. Back, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In R.K. Belew and L.B. Booker, editors, *Proc. of the 4th International Conference on Genetic Algorithms*, pages 2–9. San Mateo, CA: Morgan Kaufmann, 1991. 57

[50] Thomas Back, editor. *Handbook of Evolutionary Computation.* IOP Publishing Ltd. and Oxford University Press, 1997. 57

[51] Dennis Bahler and Laura Navarro. Combining heterogeneous sets of classifiers: Theoretical and experimental comparison of methods, 2000. 21

[52] Allen L. Barker. *Selection of Distance Metrics and Feature Subsets for k-Nearest Neighbor Classifiers.* PhD thesis, Dept. Computer Science, University of Virginia, May, 1997. 27

[53] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 24(3):173–202, 1999. 15, 16, 17

[54] W. Baxt. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4:772–780, 1992. 12, 30

[55] S.D. Bay. Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, 3(3):191–209, 1999. 30, 32, 34, 73, 83

[56] R.E. Bellman. *Adaptive Control Processes.* Princeton University Press, 1961. 5

[57] Samy Bengio and Yoshua Bengio. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Trans. Neural Networks*, 11(3), May 2000. 94

[58] K. P. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. 2002. To be published in the Proc. of KDD'02, avaliable at http://www.rpi.edu/˜demira/assemble.ps.gz. 11

[59] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981. 28

[60] James C. Bezdek. *Fuzzy Logic and Neural Network Handbook*, chapter 2. McGraw-Hill Companies, Inc., 1996. 25

[61] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997. 46

[62] A.L. Blum and R.L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127, 1992. 45

[63] D. Boyce, A. Farhi, and R Weischedel. *Optimal Subset Selection.* Springer-Verlag, Berlin, Germany, 1974. 44

[64] P. S. Bradley, U. M. Fayyad, and O. L. Mangasarian. Mathematical programming for data mining: formulations and challenges. *INFORMS Journal on Computing*, 11(3):217–238, 1999. 44

[65] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996. 17, 30

[66] L. Breiman. Bias, variance and arching classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996. 18, 41

[67] L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996. 21

[68] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Belmont, CA: Wadswoth, 1984. 56

[69] H.J. Bremermann, J. Rogson, and S. Salaff. Global properties of evolution processes. In H.H. Pattee, editor, *Natural Automata and Useful Simulations*, pages 3–42. 1966. 57

[70] C.E. Brodley. Dynamic automatic model selection. Technical Report 92-30, Department of Computer Science, University of Massachusetts, Amherst, MA, 1992. 23

[71] S.P. Brumby, Theiler, S. Perkins, N.R. Harvey, and J.J. Szymanski. Genetic programming approach to extracting features from remotely sensed imagery. In *Proc. of FUSION-2001*, 2001. 57

[72] Traina Jr. Caetano, Traina Agma, Leejay Wu, and Christos Faloutsos. Fast feature selection using fractal dimension. In *XV Brazilian Symposium on Databases (SBBD)*, 2000. 44

[73] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27, 1974. 53

[74] Erick Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Volume 1 of Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, 2001. 59, 90

[75] C. Cardie. Using decision trees to improve case-based learning. In *Proc. of the 10th International Conference on Machine Learning*, pages 25–32, Amherst, MA, 1993. Morgan Kaufmann. 53

[76] B. Cestnik. Estimating probabilities: a crucial task in machine learning. In *Proc. of the European Conference on Artificial Intelligence*, pages 147–149, Stockholm, Sweden, 1990. 78

[77] Sung-Hyuk Cha and Sargur N. Srihari. A fast nearest neighbor search algorithm by filtration. *Pattern Recognition*, 35(2):515–525, Feb. 2002. 27

[78] Philip K. Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta learning on partitioned data. In *Proc. of 12th International Conference on Machine Learning*, pages 90–98, 1995. 22

[79] Ke Chen, Lan Wang, and Huisheng Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(3):417–445, 1997. 20

[80] M. Chen, J. Han, and P. Yu. Data mining: an overview from database perspective. *IEEE Trans. Knowledge and Data Engineering*, 8(6):866–883, 1996. 44

[81] K. Cherkauer. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In P. Chan, editor, *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models,*, pages 15–21. 1996. 20

[82] Woogon Chung and Evangelia Micheli-Tzanakou. Classifiers: An overview. In E. Micheli-Tzanakou, editor, *Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence*, Industrial Electronics Series, pages 3–60. CRC Press Inc., Boca Raton, FL, 2000. 11

[83] A. Ciampi. Constructing predictions trees from data: the recpam approach. In *Computational Aspects of Model Choice*, pages 52–105. Physica-Verlag, Hedelberg, 1992. 56

[84] A. Ciampi, C.-H. Chang, S. Hogg, and S. McKinney. Recursive partition: a versatile method for exploratory data analysis in biostatistics. In I.B. MacNeil and G.J. Umphrey, editors, *Biostatistics*, pages 23–50. Reidel, Dordrecht, 1987. 23

[85] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999. 64

[86] Carlos A. Coello Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2):109–143, 2000. 61, 62, 64

[87] David W. Coit, Alice E. Smith, and David M. Tate. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, 8(2):173–182, 1996. 62

[88] Marquis J.A. Condorcet. Sur les elections par scrutiny. *Histoire de l'Academie Royale des Sciences*, pages 31–34, 1784. 12

[89] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993. 26

[90] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, IT-13:21–27, 1967. 25, 30

[91] Joao Manuel Portela da Gama. *Combining Classification Algorithms*. PhD thesis, Universidade do Porto, 1999. 12

[92] B.V. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Lost Alamitos, CA, 1990. 25, 27

[93] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis - An International Journal*, 1(3), 1997. 48

[94] T. Denoeux. A k nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE Trans. Systems, Man and Cybernetics*, 25(5):804–813, 1995. 25

[95] J. Deogun, S. Choubey, V. Raghavan, and H. Sever. Feature selection and effective classifiers. *Journal of ASIS*, 49(5):423–434, 1998. 44

[96] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall Inc., London, 1982. 73, 83

[97] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Applications of Mathematics. Springer-Verlag, 1996. 3

[98] T.G. Dietterich. Machine learning research: four current directions. *AI Magazine*, 1997. Available at: http://www.aaai.org/AITopics/html/machine.html. 13, 19, 30

[99] T.G. Dietterich. Ensemble methods in machine learning. In *Proc. of MCS'2000*, Lecture Notes in Computer Science, pages 1–15. New York: Springer Verlag, 2000. 14

[100] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 2000. 17

[101] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 31, 38

[102] S. Dimitrios and Andreas Stafylopatis Frossyniotis. A multi-SVM classification system. In *Proc. of the 2nd International Workshop on Multiple Classifier Systems, MCS 2001, Cambridge, UK*. 12

[103] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical Report CSE-92-18, Department of Computer Science and Engineering, University of Carlifornia, 1992. 51

[104] Pier Luigi Dragotti, Giovanni Poggi, and Arturo R.P. Ragozini. Compression of multispectral images by three-dimensional SPIHT algorithm. *IEEE Trans. Geoscience and Remote Sensing*, 38(1), Jan. 2000. 2

[105] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. New York: Wiley, 1973. 25, 30

[106] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks: A review. *Pattern Recognition*, 2002. To appear. 2

[107] Christos Emmanouilidis, Andrew Hunter, John Macintyre, and Chris Cox. Selecting features in neurofuzzy modelling by multiobjective genetic algorithms. In *Proc. of ICANN99, the 9th International Conference on Artificial Neural Networks*, volume 2, pages 749–754, Edinburgh, UK, Sep. 1999. 64

[108] B. Everitt. *Cluster Analysis*. New York: Halsted Press, 1974. 26, 27

[109] F.J. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice*, volume IV, pages 403–413. Elsevier Science B.V., 1994. 85

[110] E. Fix and J. Hodges Jr. Discriminatory analysis - nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, 1951. Project 21-49-004 Report. 25

[111] D.B. Fogel. *System Identification Through Simulated Evolution: A Machine Learning Approach to Modelling*. Needham, MA: Ginn Press, 1991. 57

[112] Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: A unified formulation. *IEEE Trans. Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, 1998. 64

[113] C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995. 64

[114] J. Fox. *Applied Regression Analysis*. Sage Publications, Inc., 1997. 5

[115] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. of the 13th International Conference on Machine Learning*, pages 148–156, 1996. 19

[116] J.H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987. 5

[117] J. Fukunaga and R Beauregard. An optimal global nearest neighbor metric. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(3):314–318, 1984. 26

[118] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, California, 1990. 30

[119] K. Fukunaga and P.M. Narendra. A branch and bound algorithm for computing k nearest neighbors. *IEEE Trans. Computers*, C-24:750–753, 1975. 27

[120] P.D. Gader, M.A. Mohamed, and J.M. Keller. Fusion of handwritten word classifiers. *Pattern Recognition Letters*, 17:577–584, 1996. 30

[121] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–48, 1992. 41

[122] Joydeep Ghosh. Multiclassifier systems: Back to the future. In Fabio Roli and Joser Kittler, editors, *Proc. of 3rd International Workshop on Multiple Classifier Systems (MCS2002)*, volume 2364 of *Lecture Notes in Computer Science*, pages 1–15, Cagliari, Italy, Jun.24-26 2002. Springer. 11

[123] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989. 57, 59

[124] I. Guyon. Welcome and introduction to the problem of feature/variable selection. In *NIPS 2001 Workshop on Feature/Variable Selection*, 2001. 46

[125] Mark A. Hall. *Correlation-Based Feature Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, Apr. 1999. 44

[126] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990. 11, 12, 30

[127] P.E. Hart. The condensed nearest neighbor rule. *IEEE Trans. Information Theory*, 14:515–516, 1968. 34, 35

[128] N.R. Harvey, S.P. Brumby, R.B. Porter S.J. Perkins, J. Theiler, A.C. Young, J.J. Szymanski, and J.J. Bloch. Parallel evolution of image processsing tools for multispectral imagery. In *Proc. of Imaging Spectrometry IV*, volume SPIE-4132, pages 72–80. Intl. Soc. for Opt. Eng., 2000. 57

[129] Kazuo Hattori and Masahito Takahashi. A new edited k nearest neighbor rule in the pattern recognition problem. *Pattern Recognition*, 33(3):521–528, Mar. 2000. 83

[130] D. Heath, S. Kasif, and S. Salzbery. *Committees of Decision Trees*. 1996. 19

[131] F. Herrera and M. Lozano. Fuzzy genetic algorithms: Issues and models. Technical report, Depertment of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain, 1998. 59

[132] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991. 6

[133] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proc. of 26th International Conference on Very Large Data Bases (VLDB)*, Cairo, 2000. Downloadable from the site - http://www.acm.org/sigmod/. This is the comprehensive site of the Special Interest Group on Management of Data (SIGMOD) of ACM. 27

[134] S.Y. Ho, C.C. Liu, and S. Liu. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters*, 2002. 84

[135] T.K. Ho, J.J. Hull, and S.N. Srihar. Decision combination in multiple classifier systems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:66–75, 1994. 30

[136] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975. 58

[137] J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intellgence.* MIT Press, Cambridge, MA, 1992. 2nd Edition. 59

[138] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proc. of the 1st IEEE Conference on Evolutionary Computation*, pages 82–87, 1994. 64

[139] G.F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Information Theory*, IT-14(1):55–63, 1968. 5, 7, 92

[140] H. Ichihashi and *et al.* Neuro-fuzzy ID3. *Fuzzy Sets and Systems*, 81:157–167, 1996. 56

[141] Manabu Ichino and Jack Sklansky. Optimum feature selection by zero-one integer programming. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-14(5), 1984. 44

[142] Piotr Indyk. *High-dimensional Computational Geometry.* PhD thesis, Dept. Computer Science, Stanford University, 2000. 27

[143] Piotr Indyk. Approximate nearest neighbor algorithms for frechet distance via product metrics. In *Proc. of Symposium on Computational Geometry*, 2002. 27

[144] H. Ishibuchi and T. Nakashima. Multi-objective pattern and feature selection by a genetic algorithm. In *Proc. of Genetic and Evolutionary Computation Conference*, pages 1069–1076, Las Vegas, Nevada, U.S.A., July 8-12, 2000. 58, 63

[145] Raj Dharmarajan Iyer Jr. An efficient boosting algorithm for combining preferences. Technical Report MIT-LCS-TR-811, M.I.T., 1999. 30

[146] William G. Jacoby. *Statistical Graphics for Visualizing Multivariate Data.* Sage Publications Inc., 1998. 5

[147] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997. 48, 58, 60, 86

[148] A.K. Jain and Chandrasekaran R. Dimensionality and sample size consideration in pattern recognition practice. In P.R. Krishaniah and L.N. Kanal, editors, *Handbook of Statistics*, volume II, pages 835–855. North-Holland, Amsterdam, The Netherlands, 1982. 44

[149] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:4–37, 2000. 11, 14

[150] G. James and T. Hastie. Generalizations of the bias/variance decomposition for prediction error. htttp://stat.stanford.edu/ ~gareth. 41

[151] Cezary Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Trans. Systems, Man, and Cybernetics*, 28(1):1–14, 1998. 56

[152] Thomas Jansen. On the utility of populations. Technical Report CI-102/00, Department of Computer Science, University of Dortmund, 11 2000. 94

[153] B. Jeng and *et al.* FILM: a fuzzy inductive learning method for automated knowledge acquisition. *Decision Support Systems*, 21:61–73, 1997. 56

[154] Byeungwoo Jeon and David Landgrebe. Partially supervised classification using weighted unsupervised clustering. *IEEE Trans. Geoscience and Remote Sensing*, 37(2):1073–1079, March 1999. 5

[155] George H. John. *Enhancements to the Data Mining Process*. PhD thesis, Department of Computer Science, Stanford University, Mar. 1997. 46

[156] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine learning*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann. 46, 50, 51

[157] A. Jozwik. A learning scheme for a fuzzy k-nn rule. *Pattern Recognition Letters*, 1:287–289, July 1983. 28

[158] Arto Kaarna and Jussi Parkkinen. Wavelet compression of multispectral images. In *Proc. of IASTED International Conference Computer Graphics and Imaging (CGIM'98)*, 1998. 2

[159] H. Kargupta, W. Huang, Krishnamoorthy, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems Journal*, 3(4):422–448, 2000. 11

[160] Yi ke Guo and Janjao Sutiwaraphun. Distributed classification with knowledge probing: A new framework for distributed data mining. In Hillol Kargupta and Philip Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*. MIT/AAAI Press, Sep. 2000. 11

[161] J.M. Keller, R. Gray, and JR. J.A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Trans. Systems, Man and Cybernetics*, 15(4):580–585, 1981. 28

[162] J.D. Kelly and L. Davis. Hybridizing the genetic algorithm and the k nearest neighbors classification algorithm. In *Proc. of the 4th International Conference on Genetic Algorithms and their Applications (ICGA'91)*, pages 377–383, 1991. 60

[163] K. Kira and L. Rendell. A practical approach to feature selection. In *Proc. of 9th International Conference on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1992. Morgan Kaufmann. 50, 53

[164] J. Kittler. A framework for classifier fusion: Is it still needed? Pierre Devijver Award Lecture 2000. Available at: http://www.ph.tn.tudelft.nl/Organisation/TC1/pda/pda.html. 12

[165] J. Kittler. Feature set search algorithm. In C.H. Chen, editor, *Pattern Recognition and Signal Processing*, pages 41–60. Sithof and Noordhoff, Alphen aan den Rjin, The Netherlands, 1978. 44

[166] J. Kittler. Feature selection and extraction. In Tzay Y. Young and King-Sun Fu, editors, *Handbook of Pattern Recognition and Image Processing*, pages 59–83. Academic Press, 1986. 47

[167] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998. 12, 30

[168] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997. 45, 78

[169] R. Kohavi and D.H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proc. of the 13th International Conference on Machine Learning*, 1996. 41, 43

[170] D. Koller and M. Sahami. Toward optimal feature selection. In *Proc. of the 13th International Conference on Machine Learning*, pages 284–292, Bari, Italy, 1996. Morgan Kaufmann. 53

[171] E.B. Kong and T.G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proc. of the 12th National Conference on Artificial Intelligence*, 1996. 40, 41, 72

[172] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proc. of the 7th European Conference on Machine Learning*, 1994. 50

[173] John R. Koza. Genetic programming. In James G. Williams and Allen Kent, editors, *Encyclopedia of Computer Science and Technology*, volume 39, pages 29–43. Marcel-Dekker, 1998. 57

[174] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G.Tesauro and *et al*, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. Cambridge MA:MIT Press, 1995. 12, 30

[175] F.A. Kruse, A.B. Lefkoff, J.W. Boardman, K.B. Heidebrecht, A.T. Shapiro, J.P. Barloon, and A.F.H. Goetz. The spectral image processing system(sips) - interactive visualization and analysis of imaging spectrometer data. *Remote Sensing of Environment*, 44:145–163, 1993. 26

[176] M. Kuat, D. Flotzinger, and G. Pfurtscheller. Discovering patterns in EEG-signals: Comparative study of a few methods. In *Proc. of the 6th European Conference on Machine Learning*, pages 366–371, Heidelberg, 1993. Springer-Verlag. 53

[177] M. Kudo, P. Somol, P. Pudil, M. Shimbo, and J. Sklansky. Comparison of classifier-specific feature selection algorithm. In F. J. Ferri, J. M. Inesta, A. Amin, and P. Pudil, editors, *Proc. of Joint IAPR International Workshops SSPR2000 and SPR2000*, volume 1876 of *Advances in Pattern Recognition, Lecture Notes in Computer Science*, pages 677–686, Alicante, Spain, Aug./Sep. 2000. Springer. 49

[178] Mineichi Kudo and Jack Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000. 48, 86, 91

[179] L. I. Kuncheva and L. C. Jain. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20:1149–1156, 1999. 58, 63

[180] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. Decision template for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001. 30

[181] Ludmila I. Kuncheva and Christopher J. Whitaker. Using diversity with three variants of boosting: Aggressive, conservative and inverse. In *Proc. of MCS'02*, Lecture Notes in Computer Science. Springer-Verlag, 2002. To appear. 19

[182] L. Lam and C.Y. Suen. Optimal combination of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995. 12, 30

[183] David Landgreb. Personal communication, 2001. 2

[184] D. Landgrebe. Information extraction principles and methods for multispectral and hyperspectral data. In C.H. Chen, editor, *Information Processing for Remote Sensing*. World Scientific, USA, 2000. 2

[185] P. Langley and S. Sage. Induction of selective bayesian classifiers. In *Proc. of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406, Seattle, WA, 1994. Morgan Kaufmann. 55

[186] P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 113–117, Seattle, WA, 1994. AAAI Press. 55

[187] N.Y. Lao and F.C. Wong. Hyperspectral imagery market forecast: 2000-2005. Technical report, Economic and Market Analysis Center, Systems Engineering Division, Engineering and Technology Group, December 2000. Available at http://www.aero.org/emac/PK0444vl.pdf. 2

[188] Tjen-Sien Lim, Sei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. *Machine Learning*, 1999. preprint available at http://www.recursive-partitioning.com/mach1317.pdf, and appendix containing complete tables of error rates, ranks, and training times at http://www.recursive-partitioning.com/appendix.pdf. 11

[189] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987. 16

[190] N. Littlestone and M. Warmuth. The weighted majority algorithm. Technical Report UCSC-CRL-91-28, Department of Computer Engineering and Information Sciences, University of California Santa Cruz, 1991. 16

[191] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 4:212–261, 1994. 16

[192] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proc. of 13th International Conference on Machine Learning*, pages 319–327, Bari, Italy, 1996. Morgan Kaufmann. 53

[193] Huan Liu and Hiroshi Motota. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, 1998. ISBN 0-7923-8198-X. 44

[194] Nicholas M. Short, Sr. http://rst.gsfc.nasa.gov/front/tofc.html. 1

[195] Dimuthu Prasanna Makawita, Kian-Lee Tan, and Huan Liu. Sampling from databases using B+-Trees. In *CIKM*, pages 158–164. 2000. 83

[196] Tassos Markas and John Reif. Multispectral image compression algorithms. In James A. Storer and Martin Cohn, editors, *DCC '93 : Data Compression Conference*, pages 391–400, Apr. 1993. 2

[197] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Modeling and Computer Simulation*, 8(1):3–30, Jan. 1998. 66

[198] G. Mercier, M. C Mouchot, and G. Cazuguel. Joint classification and compression of hyperspectral images. In *Proc. of 1999 International Geoscience and Remote Sensing Symposium (IGARSS)'99*, volume 4, pages 2035–2037, Piscataway, NJ, 1999. IEEE Service Center. 2

[199] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolutionary Programs.* Springer-Verlag, 1996. 59

[200] A.J. Miller. *Subset Selection in Regression.* Chapman and Hall, Washtington D.C., 1990. 44

[201] D. Miller, A.V. Rao, K. Rose, and A. Gersho. A global optimization technique for statistical classifier design. *IEEE Trans. Signal Processing*, 44(12):3108–31022, 1996. 11

[202] M. Mitchell. *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press, 1996. 59

[203] M. Mocciardi. A comparison of seven techniques of choosing subsets of pattern recognition. *IEEE Trans. Computers*, C-20:1023–1031, 1971. 44, 48

[204] A. Moore and M. Lee. Efficient algorithms for minimizing cross validation error. In *Proc. of the 11th International Conference on Machine Learning*, pages 190–198, San Francisco, CA, 1994. Morgan Kaufmann. 55

[205] T. Mori. *Taguchi Techniques for Image and Pattern Developing Technology.* New Jersey, Prentice-Hall, 1995. 84

[206] H. Muhlenbein and G. Paab. From recombination of genes to the estimation of distributions. binary parameters. In Voigt H.M. et al, editor, *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, pages 178–187. 1996. 94

[207] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. Computer*, C-26(9):917–922, 1977. 44

[208] S.L. Nitzan and J. Paroush. *Collective Decision Making*. Cambridge University Press, 1985. 12

[209] F. Olken and D. Rotem. Random sampling from databases: A survey. *Statistics & Computing*, 5(1):25–42, Mar. 1995. 83

[210] D. Opitz and D. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. 12, 30

[211] D. Opitz and J. Shavlik. Generating accurate and diverse members of a neural network ensemble of classifiers. In G.Tesauro and *et al*, editors, *Advances in Neural Information Processing System*, volume 8, pages 535–541. Cambridge MA:MIT Press, 1996. 12, 30

[212] B. Parmanto, P. Munro, and H.R. Doyle. Improving commmitte diagnosis with resampling techniques. In D.S. Touretzky, Mozer M.C., and M.E. Hesselmo, editors, *Advances in Neural Information Processing*, volume 8, pages 882–888. MIT Press, Cambridge, MA, 1996. 19

[213] Terry R. Payne. *Dimension Reduction and Representation for Nearest Neighbor Learning*. PhD thesis, Department of Computing Science, University of Aberdeen, 1999. 56

[214] Terry R. Payne and P. Edwards. Survey of work on feature selection. Draft Copy Only, 1996. 56

[215] Carlos Andres Pena-Reyes. *Coevolutionary Fuzzy Modeling*. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, 2002. 57

[216] M.A. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, George Mason University, 1997. 57

[217] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994. 48, 49, 77

[218] W.F. Punch, E.D. Goodman, Min Pei, Lai Chia-Shun, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In *Proc. of the International Conference on Genetic Algorithms and their Applications (ICGA'93)*, pages 557–564, 1993. 60

[219] R.C. Purshouse and P.J. Fleming. The multi-objective genetic algorithm applied to benchmark problems: An analysis. Technical Report 796, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, Aug. 2001. 64

[220] S-E. Qian, A. Hollinger, D. Williams, and D. Manak. 3D data compression of hyperspectral imagery using vector quantization with NDVI-based multiple codebooks. In *Proc. of IGARSS'1998*, 1998. 2

[221] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 37, 50, 56

[222] J. R. Quinlan. *Programs for Machine Learning*. Morgan Kaufmann, San Marteo, CA, 1993. 38, 56

[223] R. Quinlan. Bagging, boosting and C4.5. In *Proc. of 13th American Association for Artificial Intelligence*. AAAI Press, 1996. 17

[224] V. Ramasubramanian and Kuldip K. Paliwal. Fast nearest neighbor search algorithms based on approximation-elimination search. *Pattern Recognition*, 33(9):1486–1510, Sep. 2000. 27

[225] P.J. Rauss, J.M. Daida, and S. Chaudhary. Classification of spectral imagery using genetic programming. In et al D. Whitley, editor, *Proc. of GECCO-2000*, pages 726–733. Morgan Kaufmann, 2000. 57

[226] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evolutionary Computation*, 4:164–171, 2000. 60

[227] Michael L. Raymer, William F. Punch, Erik Goodman, Paul C. Sanschagrin, and Leslie A. Kuhn. Simultaneous feature extration and selection using a masking genetic algorithm. Technical report, Gentic Algorithms Research and Applications Group, Department of Computer Science, Michigan State University, East Lansing, MI 48824. 60

[228] F. Ricci and D.W. Aha. Extending local learners with error-correcting output codes. Technical Report AIC-97-001, Navy Center for Applied Research in Artificial Intelligence, 1997. 30

[229] J.T. Richardson, M.R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In *Proc. of the 3rd International Conference on Genetic Algorithms*, pages 191–197. Morgan-Kauffman, 1989. 63

[230] R. E. Roger and M. C. Cavenor. Lossless compression of imaging spectometer data. Technical report, Department of Electrical Engineering, Australian Defence Force Academy, Canberra AACT 2600, Australia, 1993. This report may be obtained by anonymous FTP from evans.ee.adfa.oz.au in directory pub/reports/remsen. 2

[231] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7:777–781, 1994. 12, 30

[232] Thomas Philip Runarsson and Xin Yao. Constrained evolutionary optimization: The penalty function approach. In R. Sarker, M. Mohammadian, and X. Yao, editors, *Evolutionary Optimization*, chapter 4, pages 97–113. Kluwer Academic Publishers, USA, 2002. 62

[233] M. Ryan and J. Arnold. The lossless compression of AVIRIS images by vector quantization. *IEEE Trans. Geoscience and Remote sensing*, 35(3):546–550, May 1997. 2

[234] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983. 26

[235] Hanan Samet and Gisli R. Hjaltason. Similarity searching: Indexing, nearest neighbor finding, dimensionality reduction and embedding methods for applications in multimedia databases. In *ICPR 2002*, 2002. Tutorial. 27

[236] Michael Schaepman and Walter Debruyn. http://www.apex-esa.org/. 1

[237] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. 18, 36

[238] Michail I. Schlesinger and Vaclav Hlavac. *Ten lectures on Statistical and Structural Pattern Recognition.* Kluwer Academic Publishers, 2002. 3

[239] J.C. Schlimmer. Efficiently inducing determinations: A complete and efficient search algorithm that uses optimal pruning. In *Proc. of the 10th International Conference on Machine Learning*, pages 284–290, Amherst,MA, 1987. Morgan Kaufmann. 53

[240] Stephen Scott. Feature vector selection. Lecture Note of CSCE970: Pattern Recogntion. Department of Computer Science and Engineering, University of Nebraska, Lincoln, Nebraska. Available from http://www.cse.unl.edu/˜sscott/CSCE970/. 48

[241] Rudy Setiono. Neural network feature selector. *IEEE Trans. Neural Networks*, 8(3):654–662, 1997. 44

[242] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976. 21

[243] Amanda J. C. Sharkey, editor. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems (Perspectives in Neural Computing).* Springer Verlag, Apr. 1999. 23

[244] Amanda J. C. Sharkey. Types of multinet system. In *Proc. of MCS'2002*, 2002. To appear. 14

[245] W. Siedlecki and J. Skansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988. 48

[246] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10:335–347, 1989. 58, 60, 62, 84, 85

[247] W. Siedlecki and J. Sklansky. Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In C.H. Chen, L.F. Pau, and P.S.P. Wang, editors, *Handbook of Pattern Recognition & Computer Vision*, pages 108–123. World Scientific Publishing Co. Pte. Ltd., P.O. Box 128, Farrer Road, Singapore 9128, 1995. 62

[248] M. Singh and G.M. Provan. A comparison of induction algorithms for selective and non-selective bayesian classifiers. In *Proc. of the 12th International Conference on Machine Learning*, pages 497–505, Lake Tahoe, CA, 1995. Morgan Kaufmann. 55

[249] M. Singh and G.M. Provan. Efficient learning of selective bayesian network classifiers. In *Proc. of the 13th International Conference on Machine Learning*, pages 453–461, Bari, Italy, 1996. Morgan Kaufmann. 53

[250] Samer Singh, John Haddon, and Markos Markou. Nearest neighbor classifiers in natural scene analysis. *Pattern Recognition*, 34(8):1601–1612, Aug. 2001. 25

[251] D. B. Skalak. *Prototype Selection for Composite Nearest Neighbor Classifiers.* PhD thesis, University of Massachusetts, Amherst, MA, 1997. 21, 24, 30, 32, 35, 37, 55

[252] Alice E. Smith and David W. Coit. Penalty function. In Thomas Baeck, David Gogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation.* A Joint Publication of Oxford University Press and Institute of Physics Publishing, 1995. 62

[253] P. Somol, P. Pudil, J. Novovicova, and P. Paclik. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20:1157–1163, 1999. 49

[254] C. Stanfill. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1986. 27

[255] S.D. Stearns. On selecting features for pattern classifiers. In *Proceedings of the 3rd International Conference on Pattern Recognition*, pages 71–75, Coronado, CA, 1976. 44

[256] Hisashi Tamaki, Hajime Kita, and Shigenobu Kobayashi. Multi-objective optimizaiton by genetic algorithms: A review. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proc. of the 1996 International Conference On Evolutionary Computation*, pages 517–522, Nagoya, Japan, 1996. 64

[257] K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary algorithms for multi-objective optimizations: Performance assessments and comparisons. *Artificial Intelligence Review*, 17(4), Jun. 2002. 64

[258] Stephen R. Tate. Band ordering in lossless compression of multispectral images. In *Proc. of Data Compression Conference, Snowbird, Utah*, pages 311–320, 1994. 2

[259] Sergios Theodoridis. *Pattern Recognition*. Academic Press, San Diego, CA, Jan. 1999. 3

[260] H.H. Thodberg. A review of bayesian neural networks with an application to near infrared spectroscopy. Technical report, The Danish Meat Research Institute, 1995. 30

[261] R. Tibshirani. Bias, variance and prediction error for classification rules. Technical report, Department of Statistics, University of Toronto, 1996. 41

[262] K.M. Ting and L.H. Witten. Stacked generalization: When does it work? In *Proc. of International Joint Conference on Artificial Intelligence*, 1997. 21

[263] D. Tretter, N. Memon, and C. Bouman. Multispectral image coding. In Alan Bovik, editor, *The Image and Video Processing Handbook*. Academic Press. To appear. 2

[264] Kagan Tumer and Joydeep Ghosh. Classifier combining: Analytical results and implications. In *National Conference on Artificial Intelligence*. Portland, August 1996. 14

[265] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3 & 4):385–404, December 1996. Special issue on combining artificial neural networks: ensemble approaches. 14

[266] Kagan Tumer and Joydeep Ghosh. Theoretical foundations of linear and order statistics combiners for neural pattern classifiers. Technical Report TR-95-02-98, 1996. 14

[267] M. Umanol and *et al.* Fuzzy decision trees by fuzzy ID3 and its application to diagnosis systems. In *Proc. of the IEEE International Conference on Fuzzy Systems*, pages 2113–2118, 26-29, Jun. 1994. 56

[268] P.E. Utgoff. Perception trees: A case study in hybrid concept representations. *Connection Science*, 1:377–391, 1989. 23

[269] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 18

[270] V.N. Vapnik. *The Nature of Statistical Learning Theory*. John Wiley, New York, 1998. 11

[271] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Air Force Institute of Technology, Jun. 1999. 64

[272] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Air Force Institute of Technology, Wright Paterson AFB, Oct. 1998. 64

[273] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000. 64

[274] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002. 22

[275] S. Voget and M. Kolonko. Multi-criteria optimization with a fuzzy genetic algorithm. *Journal of Heuristic*, 1998. 59

[276] S.S. Wilks. *Mathematical Statistics*. Wiley, New York, 1963. 47

[277] D. Wilson and T Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997. Journal of Artificial Intelligence Reseach (JAIR) is an online journal which can be freely accessible from: http://www.jair.org. 26

[278] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, Complex Systems Group, Theoretical Division, and Center for Non-linear Studies, MS B213, LANL, Los Alamos, NM, 1990. 21, 36

[279] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19:405–410, 1997. 12, 30

[280] Yingquan Wu, Krassimir Ianakiev, and Venu Govindaraju. Improved k nearest neighbor classification. *Pattern Recognition*, 35(10), 2002. 25

[281] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Trans. Systems, Man and Cybernetics*, 22:418–435, 1992. 30

[282] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In *Proc. of the Second Conference on Genetic Programming*, 1997. 58, 63

[283] Mary M. Yang. Hyperspectral image compression and client/sever software. Available at http://www.reisys.com/ehb/proposal.pdf. 2

[284] Shixin Yu, Steven De Backer, and Paul Scheunders. Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. *Pattern Recognition Letters*, 23:183–190, 2002. 57, 60, 61, 79

[285] Shixin Yu and Paul Scheunders. On combining nearest neighbor classifiers: An empirical study on hyperspectral remote sensing data. *Pattern Recognition Letters*. submitted. 30

[286] X. Zhang, J.P. Mesirov, and D.L. Waltz. Hybrid system for protein secondary structure prediction. *Journal of Molecular Biology*, 225:1049–1063, 1992. 30

[287] Ye Zhang and Mita D. Desai. Hyperspectral image compression based on adaptive recursive bidirection prediction/JPEG. *Pattern Recognition*, 33(11):1851–1860, Nov. 2000. 2

[288] Zijian Zheng. Naive bayesian classifier committees. In *Proc. of ECML'98*, pages 196–207. Springer Verlag, 1998. 23

[289] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999. Tik-Schriftenreihe nr. 30. 61, 62

[290] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. Technical Report TIK-Report No. 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology, Zurich, Dec. 1999. Revised Version. 64