Faculty of Engineering of the University of Porto

Combining Classifiers

Analysis and Application to Medical Data

Luís Filipe Barbosa de Almeida Alexandre

Thesis submitted to the Department of Electrical and Computer Engineering for the partial fulfillment of the requirements for the degree of Doctor of Philosophy made under the supervision of Doctor Aurélio Campilho, Associate Professor ('Agregado') at the Faculty of Engineering of the University of Porto, Portugal and Doctor Mohamed Kamel, Professor at the University of Waterloo, Canada

May 2002

I dedicate this work to my daughter, Matilde.

Abstract

One of the chapters of a pattern recognition system is the classification. To improve classification results we can improve the type and number of features used or the classifiers themselves. One of the processes that can be used to improve the classification results is to combine the decisions of several classifiers instead of using only one.

This thesis presents an effort to gain a deeper theoretical understanding of some of the aspects involved in the combination of decisions from several classifiers.

A revision of the state of the art in combining information from several classifiers is presented.

The background on error evaluation and error bounds is explained, and new upper and lower bounds for the empirical classification error are introduced. These are based on a theorem for the proximity among distributions.

Results on simple rules for combining classifiers, such as the sum, the product and the weighted average, are also produced.

The equivalence between the sum and product combination rules is proved for some types of classification problems and a model for the added error in a combination using weighted average is also developed.

Several modular neural network architectures were studied. Bounds for their error and models for the probability of correct classification are given. A unifying view of three of these networks is also presented.

Applications of several of these new results are made to medical problems, involving medical image, physiological signals and other types of data.

Finally, conclusions are drawn and future research directions are proposed.

Resumo

Um dos componentes dum sistema de reconhecimento de padrões é a classificação. Para melhorar os resultados de classificação pode melhorar-se o tipo e número das características ou os classificadores. Uma das formas que pode ser usada para melhorar os resultados de classificação é a combinação de decisões provenientes de vários classificadores em vez da utilização de apenas um único.

Esta tese apresenta um esforço no sentido de aumentar o conhecimento teórico de alguns dos aspectos envolvidos na combinação de decisões de diversos classificadores.

É apresentada uma revisão do estado da arte na combinação de decisões de múltiplos classificadores.

É descrito o conhecimento estabelecido no âmbito da avaliação do erro e seus limites. São apresentados novos majorante e minorante para o erro baseados num teorema para a proximidade de distribuições.

São também introduzidos resultados relativos a regras simples para a combinação das decisões de classificadores, como a soma, o produto e a média pesada.

É provada a equivalência entre as regras de combinação da soma e do produto para alguns problemas de classificação. É também desenvolvido um modelo para o erro adicional de uma combinação de classificadores usando a média pesada.

São estudadas várias redes neuronais modulares. São definidos majorantes para o erro e modelos para a probabilidade de classificação correcta. É também apresentada uma visão unificadora de três destas redes.

Vários destes resultados são aplicados a problemas médicos envolvendo imagem médica, sinais fisiológicos e outro tipo de dados.

Por fim, são retiradas algumas conclusões e são propostas perspectivas de desenvolvimento futuro.

Résumé

Un des composants d'un système de reconnaissance de formes est la classification. L'amélioration des résultats de la classification peut être obtenu en améliorant le type et le nombre des caractéristiques ou encore les classificateurs. Une des procédés qui peut être utilisé dans ce but est la combinaison des décisions provenant de plusieurs classificateurs en lieu et place de l'utilisation d'un seul de ceux-ci.

Cette thèse présente une contribution á l'étude théorique de certains des aspects liés à la combinaison de décisions provenant de divers classificateurs.

Une révision de l'état de l'art de la combinaison de décisions de multiples classificateurs est présentée.

Sont aussi présentés dans cette thèse le contexte et les connaissances prérequises propres à l'évaluation d'erreur et aux marges d'erreur. De nouveaux majorants et minorants pour les erreurs basées un théorème pour les proximités de distributions.

Des résultats relatifs aux règles simples pour la combinaison de décisions de classificateurs, comme la somme, le produit ou la moyenne pondérée sont énoncés.

L'équivalence entre les règles de combinaison par somme et par produit pour certains problèmes de classifications est démontrée. Un modèle de pour l'erreur additionnelle d'une combinaison de classificateurs utilisant la moyenne pondérée est ensuite présenté.

Plusieurs réseaux de neurones modulaires ont été étudiés. Des majorants d'erreur et des modèles pour la probabilité de classification correcte sont ensuite présentés. Une approche unificatrice de trois de ces réseaux est développée.

Plusieurs de ces résultats sont appliqués à des problèmes médicaux impliquant l'imagerie médicale, données physiologiques et autres.

Finalement, sont énoncées quelques conclusions et perspectives futures.

Acknowledgments

First I want to thank my supervisors, Prof. Campilho and Prof. Kamel. Prof. Campilho always supported me even if I constantly changed subjects and showed him different versions of my work. Prof. Kamel for his focused vision and guidance.

Both chest radiographs and knowledge of how to screen them for pathologies were kindly provided by Dr. Mário Júlio Silva, to whom I wish to thank.

I benefited from several discussions, particularly on probabilistic formalisms, that I had with Prof. Helena Ferreira of the Department of Mathematics at Universidade da Beira Interior (UBI).

This work would not have gone so smoothly if it did not had the support of my colleagues at Instituto de Engenharia Biomédica (INEB): Carlos Felgueiras, Fernando Sereno, Jorge Barbosa and Miguel Correia.

I wish to thank Simão de Sousa for his help with the french abstract.

I want to thank UBI for giving me a three year leave and financial support for the PhD fees.

I would also like to acknowledge the support of project number POSI/35590/SRI/2000 approved by the Portuguese Fundação para a Ciência e Tecnologia (FCT) and POSI and partially financed by FEDER.

Contents

List of Figures

List of Tables

Notation

Bold letters represent vectors. If $\mathbf{x}$ is a N dimensional vector then $x_i, i = 1, \ldots, N$ are the vector components. Random variables are denoted by capital letters. If X is a random variable, x denotes a particular value of X.

Symbols

$\emptyset$	empty set
c_i	class i
E_{bay}	Bayes error
E_{add}	added error (difference between the total error and the Bayes error)
$E[X]$	expected value of a random variable X
$f_i^j(x)$	approximation to the posterior probability for c_i, by a classifier j
$\mathcal{I}_A$	indicator function of event A
$p(\centerdot)$	probability density function
$P(\centerdot)$	probability
$P(c_i)$	prior probability
$P(c_i\|x)$	posterior probability
$p(x\|c_i)$	class-conditional probability density function

Abbreviations

AGE	Average Generalization Error
CAD	Computer-Aided Diagnosis
CDA	Costo-Diafragmatic Angle
CTG	Cardiotocography
FHR	Fetal Heart Rate
i.i.d.	independent and identically distributed
k-NN	k-Nearest Neighbour
MLP	Multi-Layer Perceptron
MNN	Modular Neural Network
MSE	Mean Squared Error
NN	Neural Network
OCR	Optical Character Recognition
PCC	Probability of Correct Classification
PR	Pattern Recognition
ROI	Region Of Interest
VC	Vapnik-Chervonenkis

Theories are nets cast to catch what we call 'the world': to rationalize, to explain, and to master it. We endeavour to make the mesh ever finer and finer.

Karl Popper

Chapter 1

Introduction

The main subject of this thesis is the combination of classifiers and its application to medical data. Classification is an important and central chapter of Pattern Recognition (PR), considered as a broad field of knowledge that studies how machines (computers) can recognize patterns. In this context, a pattern is an entity that can be identified and labeled.

This recognition process is inherent to human activity. Humans recognize patterns just by looking around. Humans categorize objects, identify people, read and listen. When humans recognize an object (a car, a book, a door) they place it in categories or classes. This also happens when recognizing people - the input pattern received by the eyes is classified into one of the persons previously known or else into the class of strangers. While reading, before the actual comprehension of what is read takes place, we first have to identify the characters and punctuation. When we hear, we also make use of classification in order to recognize music, talking (the individual words) and other types of sound (including noise). The aim of pattern recognition is to imitate this recognition/categorization capability, using computers.

Figure 1.1 shows a block diagram representation of a pattern recognition system. The first step consists in gathering information from the world (usually by a process of analog to digital conversion). This information is normally pre-processed to remove noise and discard irrelevant information. The next step consists in the extraction and selection of features. Features can be seen as measurements or quantities derived (extracted) from the measurements. Since it is possible to de-

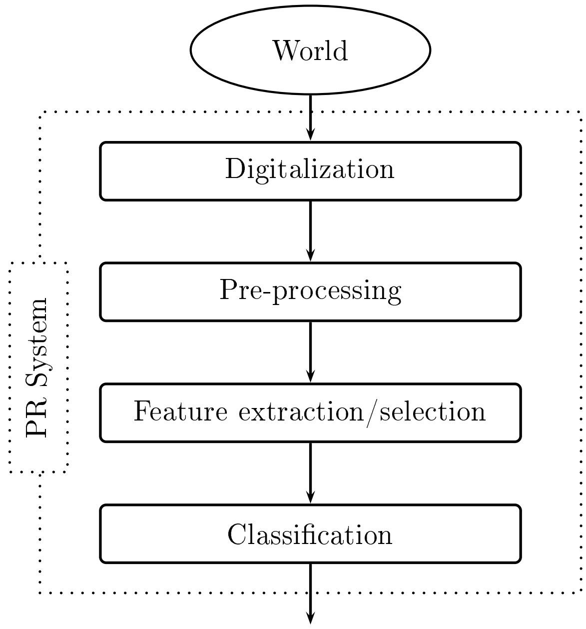

Figure 1.1: Block diagram for a typical pattern recognition system.

fine multiple features for a given pattern, those to be used are selected among the set of possible features. Finally, these features are fed into a classification module that assigns a label to the initial pattern.

As a simple example of this process we can consider the case of an optical character recognition (OCR) system. The patterns are typed or handwritten characters. The input system, a camera or a scanner, captures an image. This image is digitized and fed into a computer. Then the pre-processing step takes place. It includes noise reduction, estimating the background, enhancing the characters and thresholding the image, finally creating a black and white image. From the resulting binary image, several features are calculated. These are highly dependent of the particular problem in consideration. For the case of OCR, the features could be the height of the symbol, its width, the ratio of these two, the area occupied by the symbol and so on. The idea is that a feature should enable the classifier to distinguish among the different types of patterns. Usually a set of features is used (sometimes arranged in a vector form, if these features are scalars), so that the classifier can have the maximum discriminant power (capacity to distinguish between classes). After having calculated the features for a specific pattern, the information is given to a classifier for deciding to which of the 36 possible classes (letters plus digits, excluding punctuation) this pattern belongs to.

The steps involved in a PR system can also be illustrated with a medical example, more precisely an example of a Computer-Aided Diagnosis (CAD) system. Given the delicate nature of medical problems (involving human health and possibly life), these PR systems are used to help the medical doctor to reach a decision. This is achieved by pointing possible pathologies in the exams.

Mammography is a good example of application of a CAD system where there is intensive research. One important issue is to help the detection of microcalcifications in breast tissues. This is one of the most important features for the detection of breast cancer. The first step is image acquisition. Nowadays most mammography is digital, and an image is acquired ready for computer processing (no need for a digitalization step after the acquisition). The pre-processing usually includes noise reduction, gray level equalization and an estimation of the background image. This estimation is normally obtained from the original image using filters that reduce the high frequency components. The image obtained after subtracting the background image from the image after gray level equalization is checked for regions with non-gaussian signals - these are ROIs (regions of interest). Several features are calculated in the ROIs. These can be, e.g., area, average pixel value and several directional gradients. These features are then used for the final classification process which should indicate whether a given ROI is suspect of containing a microcalcification. This information is used to draw the attention of the medical doctor for certain areas of the image that might otherwise go unnoticed.

1.1 The importance of combining classifiers

There are two main approaches to improve the performance of a pattern recognition system: the first is to consider features with high discriminant power; the second is to improve the classification results. These two approaches can, of course, be used simultaneously. In this case, the process would be iterative since the introduction of new features would imply a readjustment of the classifiers. The relevance and the discriminative power of the new features would have to be evaluated since some might be correlated and so a subset of features that yields better classification results should be determined (feature selection). If the classification results are not satisfactory, new features should be introduced and the process starts again.

In this thesis we will focus our attention on the second approach: we investigate one way to improve classification results, namely by taking into account the information from more than one classifier.

The general idea is to use, in the classification module of a pattern recognition system, more than one classifier and combine the information from these several sources to produce the final classification decision.

The issue is how should such a combination be done in order to achieve the best possible results. There is no universal answer to this question and in this thesis we try to make some contributions to the answer.

The motivation for the use of a combination of classifiers is that classifiers with different methodologies or that use different features can complement each other [54]. Evidence has shown that such a combination usually displays a better performance than any of the individual classifiers, although an increase in performance is not guaranteed. This implies that it is important to know when to use a combination of classifiers since such a combination can in certain cases show a worst performance than the isolated classifiers [100].

A combination of classifiers can also be used to reach a determined performance faster than it would take using only one classifier. This happens because with a combination approach, high accuracy in classification can be obtained even if the individual classifiers are not fully optimized (classifier optimization can take a long time) [54].

Another reason to combine the information from several classifiers is to be able to use different types of features and scales [95]. A single classifier is usually trained with only one type of feature as input. In order to use different types of features in a classification task, it would be best if several classifiers were used, one for each type of feature, and their results combined (see [16, 27] and [95]).

A combination of classifiers can also be used to deal with the variance-bias dilemma [18, 60]. The classification error can be decomposed into a bias component plus a variance component. When trying to decrease one of these terms the other usually increases. With a combination of classifiers one can reduce the variance [18]. This way, to reduce the overall error one could use a set of classifiers with low bias and high variance. These classifiers can be, for instance, neural networks

(NNs) larger than necessary to simply learn the problem [92].

A combination can also be justified by Bayesian learning theory that states that, to maximize the accuracy of the predictions, all hypothesis should be taken into account, weighted by their respective posterior probability given the training data [18, 42].

There are two more arguments that support a combination of classifiers instead of the use of 'the best' classifier alone [18]. If after training a set of classifiers only the best one is selected, all the effort put in training the classifiers that were not chosen, is wasted. Also, the best classifier on the training set is not guaranteed to be the best in the test set. This implies that by choosing only one classifier, it is possible to disregard the one with best generalization capabilities.

1.2 Motivation and objectives

Pattern recognition is a field that, due to its numerous applications, has many practitioners and is growing fast. To illustrate this statement we may refer that the International Conference on Pattern Recognition ICPR 2000 had 978 papers published.

Among those, are medical applications. These are of particular interest to the author since he is a researcher at Instituto de Engenharia Biomédica (INEB), where the research is oriented toward medical applications. He is working in a project[1] aiming to build a CAD system, to aid a medical doctor in the process of analyzing chest radiographs.

This consists, essentially, of the development of a PR system, as we have discussed above. Our aim is to obtain a system as accurate as possible. As we have already seen, to increase the accuracy of a PR system, one can improve the features or the classification results. The author focused on the improvement of classification results, particularly, through the use of combinations of classifiers.

This thesis also presents an effort to analyze some classification architectures, namely the decoupled, other-output and cooperative modular neural networks, that are the main subject of chapter 5. The idea was to study the properties of these classification architectures in order to increase our understanding of how they

[1] Project number POSI/35590/SRI/2000 of the Portuguese 'Fundação para a Ciência e Tecnologia'.

work.

The objective of this thesis is to contribute to a deeper theoretical understanding of how methods that combine classifiers work. The results have the immediate goal of application to problems involving medical data, but are general and can be used on any PR problem.

1.3 Contributions

The contributions of this research include:

- reviewed the state of the art in combining information from several classifiers (chapter 2);
- defined new upper and lower bounds for the empirical classification error (chapter 3);
- proved that the sum and product rules are equivalent when combining only two classifiers in a two class problem and that this is the only case where this equivalence occurs (chapter 4) [4];
- showed how the added error varies when combining unbiased and independent classifiers using weighted average (chapter 4) [3];
- in that context (weighted combination of unbiased and independent classifiers), showed that if nothing is known *a priori* about the performance of the classifiers, the simple average should be used (chapter 4) [3];
- showed that in separable problems, a modular neural network (MNN) outperforms an equal capacity neural network (chapter 5) [1] ;
- built probabilistic models for the probability of correct classification for the decoupled, other-output and CMNN modular neural networks (chapter 5);
- validated the models through experiments and presented a unifying view of those MNNs (chapter 5);
- illustrated the findings with medical problems including image and signal data (chapter 6) [5, 6].

1.4 Thesis outline

The next chapter introduces the background, giving the notation and several definitions, and also presents a comprehensive literature review.

Chapter 3 presents the background on error evaluation and error bounds, and introduces new bounds for the empirical error. These are based on an existing result for the proximity among distributions.

Chapter 4 presents results on simple rules for combining classifiers, such as the sum, the product and the weighted average. It establishes some equivalences between the sum and product combination rules and develops a model for the added error in a combination using weighted average.

Modular neural network architectures are the subject of chapter 5. Several architectures are described and bounds for their error are developed. Models for the probability of correct classification are also given.

Chapter 6 contains applications of several results of the previous chapters to medical problems, involving image, signal and other data.

The final chapter summarizes the work done, draws conclusions and presents future research directions.

Chapter 2

Definitions and background

This chapter introduces concepts and notation relevant to the thesis. These include the formalization of the classification problem, the notions of loss and risk, the Bayes classifier and the Bayes error. It also explains how to combine classifiers that produce estimates of the posterior probabilities. It presents the multi-layer perceptron (MLP), a type of neural network (NN) that is used extensively in the thesis, introduces the associated notation and how it can be trained to produce the posterior probability estimates (section 2.2).

Section 2.3 presents a thorough literature review on methods used to combine the information from several classifiers and includes a table condensing the main characteristics of these methods.

2.1 Definitions and notation

2.1.1 General framework

A pattern is some entity which can be distinguished and labeled. It can be, e.g., a car in a problem where the goal is to distinguish between different types of cars, a digit in an OCR problem, a physiological signal or a medical image in a diagnostic problem in medicine.

While distinguishing between patterns, they are labeled as belonging to a class $c_i,\ i = 1, \ldots, L$. In fact, as we have seen in chapter 1, the input to a classification system consist in feature vectors and not the patterns themselves. The feature vectors are random vectors, $\mathbf{X}$, containing measurements made on the patterns.

We will use capital letters to denote random variables, such as X, and lowercase to denote their value, x. Boldface will be used to represent vectors. The vectors belong to a feature space F. They are random because a given pattern may be represented by different vectors $\mathbf{X}$ if the measurements are made at different instants. This can happen because the pattern suffered slight changes (the same car may be seen from a different angle or the same symbol in an OCR problem may have been written by a different person), or simply because the measurement process introduces random errors.

To simplify the notation we will consider the features as a simple random variable X instead of a random vector $\mathbf{X}$, but will still call it a vector (in fact, it is a unidimensional vector).

Classification consists on using a function $g(X)$, the classifier, to map a feature vector to a class label, $g(X) : F \rightarrow \{c_1, \ldots, c_L\}$.

Note that $C = g(X)$ is also a random variable, since a given feature vector $X = x$ may correspond to different classes. This multiple correspondence can happen as two different types of cars can have the same length or width, or in the OCR problem, two different symbols can have the same height or width.

A classifier is build from a set of feature vectors with class labels. This is called the training set. The classification problem consists in using the classifier to classify a set of unlabeled feature vectors, the test set.

The probability that a class c_i occurs is denoted $P(C = c_i)$. This is the proportion of cases in which a pattern belongs to c_i in the population under study. We call it the prior probability, and will represent it by $P(c_i)$. The probability that a class c_i occurs, given that the pattern $X = x$ was measured, is called the posterior probability, and is denoted by $P(C = c_i|X = x)$. To simplify the notation, we represent it by $P(c_i|x)$. It plays a central role in classification, as we shall see.

Using the Bayes rule it is possible to relate the posterior probability with the class conditional probability density function $p(X = x|C = c_i)$ (which we represent, again in order to simplify the notation, by $p(x|c_i)$)

$$P(c_i|x) = \frac{p(x|c_i)P(c_i)}{p(x)} \tag{2.1}$$

where $p(x)$ represents the probability density function of the random variable X. This is given by

$$p(x) = \sum_{i=1}^{L} p(x|c_i)P(c_i) \tag{2.2}$$

2.1.2 The Bayes error

When a classifier makes a decision we can consider that it implies a loss, which is zero in the case of correct classification and positive otherwise. A loss function L_{kj} is defined. It gives the loss if the pattern belongs to class c_k and is classified as belonging to class c_j. Generally, L_{kj} is a matrix containing the loss for all the possible combinations of true and assigned class.

The loss can be used to differentiate between different types of error. Most notably, in medical applications it is far more severe to let a pathology go unnoticed (false negative) than it is to diagnose a pathology where there is none (false positive). In this case a higher value of loss should be assigned to a false negative than to a false positive.

Equation 2.3 is the usual definition of the loss function, when it is not possible to estimate the loss of the different possible errors. Notice that it is a function of the input pattern X since the assigned class c_j is a function of X.

$$L_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases} \tag{2.3}$$

If all patterns belong to a given class c_k then the expected value of the loss is

$$E[L_{kj}|C = c_k] = \sum_{j=1}^{L} L_{kj} \int_{R_j} p(x|c_k)dx \tag{2.4}$$

since the class that is going to be assigned, c_j, is a function of X. R_j is the region of input space where X belongs to class c_j.

The expected value of the loss, considering that the pattern may belong to any class, is called the risk

$$
\begin{aligned}
R &= E[L_{kj}] \\
&= \sum_{k=1}^{L} E[L_{kj}|C = c_k]P(c_k) \\
&= \sum_{k=1}^{L}\sum_{j=1}^{L} L_{kj} \int_{R_j} p(x|c_k)dxP(c_k) \\
&= \sum_{j=1}^{L} \int_{R_j} \sum_{k=1}^{L} L_{kj}p(x|c_k)P(c_k)dx
\end{aligned}
\tag{2.5}
$$

To minimize the risk, the integrand in each point x must be minimized. So x should be assigned to class c_j when

$$
\sum_{k=1}^{L} L_{kj}p(x|c_k)P(c_k) < \sum_{k=1}^{L} L_{ki}p(x|c_k)P(c_k),\ \forall i \neq j \tag{2.6}
$$

If the loss function used is the one in 2.3, condition 2.6 can be written as

$$
p(x|c_j)P(c_j) > p(x|c_i)P(c_i),\ \forall i \neq j \tag{2.7}
$$

This can be written, using the definition of posterior probabilities, as

$$
p(c_j|x) > p(c_i|x),\ \forall i \neq j \tag{2.8}
$$

The classifier that minimizes the risk using the loss in 2.3 is called the Bayes classification rule [80] (or Bayes classifier) and consists in

$$
g(X) = j \text{ if } j = \arg \max_{i=1,\ldots,L} p(c_i|x) \tag{2.9}
$$

The Bayes classifier is the optimum classifier in the sense that it minimizes the risk and the error it produces is called the Bayes error.

Figure 2.1 shows, as a shaded region, the Bayesian error region for a one dimensional problem. It also shows the Bayes boundary x^*, which is the optimal boundary between the two depicted classes in the sense that it minimizes the classification error.

It is possible to obtain a classification error smaller than the Bayes error, if the reject option is used. This consists in allowing the classifier to reject some patterns

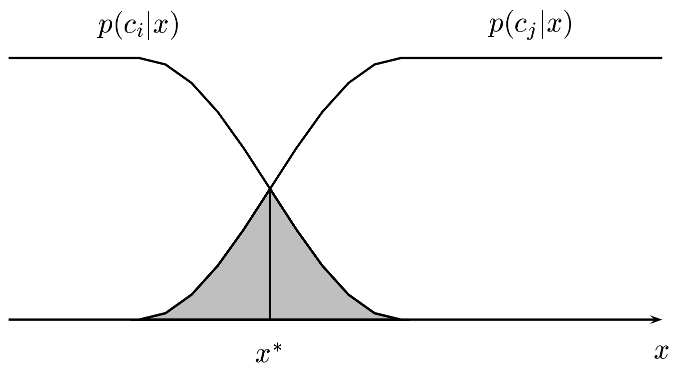

Figure 2.1: The Bayes error.

that it will probably misclassify [40] (the use of the reject option in a combination of classifiers can be seen in [41]). By using a reject option, what happens is that the original problem of classifying N patterns is transformed into another where only $M < N$ patterns are classified. These M patterns are those that are not near the decision boundaries, such that the classifier has low probability of making an error on their classification. So, in fact, the Bayes error is the smallest possible error when considering the classification of all the N patterns. We will not consider the reject option in this thesis.

In order to use this classification rule it is necessary to know the posterior probabilities. In this thesis, we will usually use a neural network (in fact, a combination of neural networks) to estimate these posterior probabilities and then use the Bayes classifier, selecting the class with highest posterior probability as the classification result.

2.1.3 Combining different classifiers

Consider N classifiers whose outputs are expected to approximate the posterior probabilities, $p(c_i|x),\ i = 1, \ldots, L$. The approximation to $p(c_i|x)$ provided by a given classifier j, is

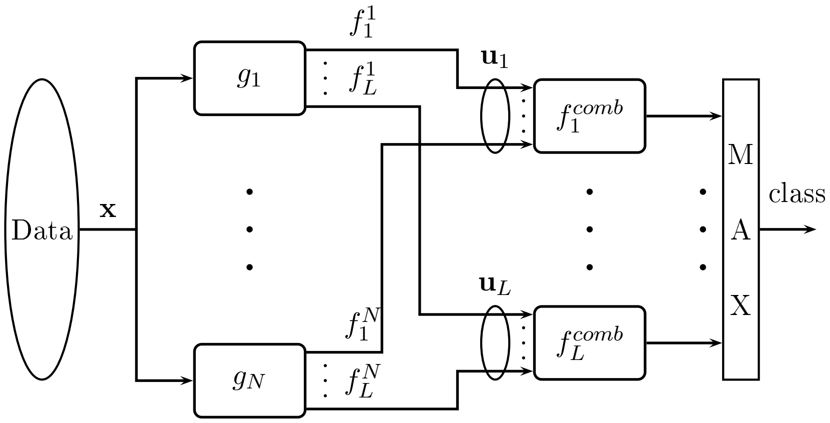

Figure 2.2: The combination process.

$$f_i^j(x) = p(c_i|x) + \epsilon_i^j(x),\ \ i = 1, \dots, L \tag{2.10}$$

where $\epsilon_i^j(x)$ represents the error that the classifier j introduces, when approximating the posterior probability $p(c_i|x)$.

The combination of information from the different classifiers is done by building new predictions for the posterior probabilities from the individual classifiers' predictions. The combined prediction for class c_i is then given by

$$f_i^{comb}(\mathbf{u}_i) = G(\mathbf{u}_i) \tag{2.11}$$

where $\mathbf{u}_i = (f_i^1(x), ..., f_i^N(x))$, 'comb' represents a given combination rule and $G(\centerdot)$ the corresponding function.

This process is illustrated in figure 2.1.3, where the output of the classifiers $g_i,\ i = 1, \dots, N$ is combined and the final class label is obtained using the Bayes classifier, i.e., choosing the class with highest (combined) posterior probability. The input pattern is assigned to class c_j if

$$f_j^{comb} > f_i^{comb}, \forall i \neq j,\ \ i, j \in \{1, .., L\} \tag{2.12}$$

2.1.4 Classification using neural networks

Among the possible classifiers that can be used are (artificial) neural networks (NNs) [50]. These are non-parametric classifiers, i.e., it is not necessary to make any assumptions on the distribution of the data in order to use them.

In this thesis, multi-layer perceptrons (MLPs) are used extensively as classifiers. These are supervised learning algorithms. They are used since they can be trained to approximate the posterior probabilities, as we will discuss later. The training algorithm used is the resilient back-propagation. This is a variant of the classical back-propagation learning algorithm. It eliminates the harmful effect of the small magnitudes of the partial derivatives of the sigmoid transfer functions used in the neurons. The partial derivatives are calculated in the back-propagation algorithm to determine the weight change during the network training. The sigmoid function has a slope that approaches zero for large values of the input. Thus, even when the weights are far from the optimal values, the change in their value in one training epoch can be small, making convergence a very slow process. The resilient back-propagation algorithm considers only the sign of the derivative and not the magnitude, for the weight update, thus avoiding the referred problem [33].

Many other variants of the classical back-propagation training algorithm exist. This variant was chosen based on the experience we had with several of these algorithms. This training algorithm performs well, since it keeps converging when others get very slow or stop converging. It also has a robust behavior since the results obtained with it when repeating an experiment, which involves only a change in the random initialization of the weights of the MLP, are similar. This is not the case with training functions based, for instance, in the conjugate gradient. The results obtained with these training algorithms are more variable, even if sometimes yield better performance than the obtained using resilient back-propagation.

One hidden layer MLPs will be used. This guarantees that they are universal approximators [18, 52], i.e., they can be used to approximate any function with the desired accuracy, given that sufficient hidden layer neurons are available. The topology of the MLP will be referred as $[a : b : c]$ where a is the number of features in the problem, i.e., the dimensionality of the input feature vector, b is the number of neurons in the hidden layer and c the number of neurons on the output layer. Since the MLPs will be used to approximate the posterior probabilities (see below),

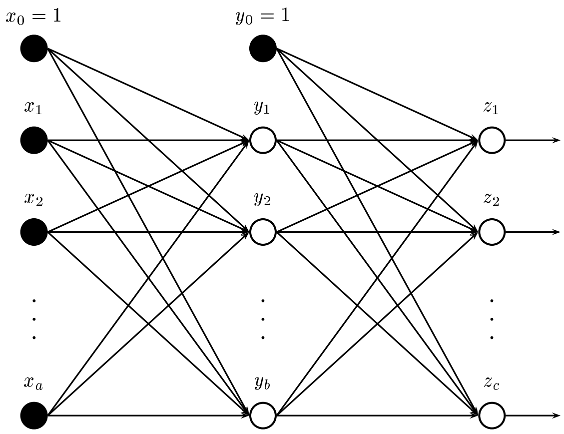

Figure 2.3: The schematic representation of an MLP with topology $[a : b : c]$. Closed circles represent terminals or scalars. Open circles represent neurons.

the number of neurons in the output layer is equal to the number of classes in the problem, i.e., $c = L$.

Each neuron on the hidden layer has as many inputs as the number of features, plus the bias. This gives the total number of weights, for a $[a : b : c]$ MLP that is equal to $(a + 1) \times b + (b + 1) \times c$. Figure 2.3 shows an MLP with the referred characteristics.

In order for the MLP to learn the classification function it is necessary to supply some examples. A set of feature vectors along with their classes is called a training set: $TS = \{(\mathbf{X_i}, C_i), i = 1, \ldots, T\}$. The MLP will try to correctly assign a class label to a pattern not in TS.

Given an MLP, it is possible to train it in such a way that it outputs approximations to the posterior probabilities. This is done by training the MLP in a 1-out-of-L fashion. If the problem has L classes, the MLP is built with L outputs. It is then trained to reach a level of 1 on the output that corresponds to the true class and zero on all other outputs. This ensures that the outputs of the MLP approximate the posterior probabilities [79].

This allows a different approach to classification using neural networks: instead of having the neural network output directly the class label, it outputs approximations to the posterior probabilities. It is then possible to combine the output of several classifiers and use the Bayes classifier on the combined posterior probabilities. This process, along with the definition of the Bayes classifier was explained in section 2.1.2.

Note that it is possible to combine the output of several classifiers even if they do not approximate the posterior probabilities, namely using voting rules as we will see in section 2.2. If the classifiers approximate the posterior probabilities, then there are many more possibilities for combining the information they provide. Also, when classifiers output only class labels they do not provide quantitative information on how probable a class is, nor a rank of preferred classes. This kind of information is present when the classifiers approximate the posterior probabilities.

2.2 A review of classifier combination methods

2.2.1 Introduction

Combining classifiers is not a new idea. According to [49] Laplace, in 1818, was probably the first to use it. Since then, both the statistics and the economics communities have used the idea of combination of classifiers under the names ‘predictor combining’ and ‘combining forecasts’, respectively. According to [17], the idea of combining classifiers was introduced in the machine learning field by Nilsson in 1965, who proposed the committee machines. But the recent expansion in the field started in 1990 with the paper by Hansen and Salamon [48].

2.2.2 Combining methods

Several methods have been proposed to combine classifiers. These methods can be categorized based on different characteristics:

- Some methods use the output from all the classifiers, while others only use the output of some of the classifiers. This feature, which we can call ‘data awareness’ or as in [59], ‘adaptability’, represents the ability of the combining method to adapt to the data and give more importance to some classifiers than to others, depending on the particular pattern to be classified or the region of space from which it comes from.

- The combination methods exhibit a certain architecture. This can be a parallel architecture if all classifiers are consulted simultaneously, serial if some of the classifiers are consulted first and others are consulted afterwords. This characteristic can also be used to distinguish more complex situations such as hierarchical or tree combinations.
- Some methods generate new classifiers by creating different data sets from the existing one(s). The idea behind this approach is to create differences in the classifiers by making them learn from different data sets.
- The output of the classifiers is of different types: [101] divides these types in abstract, rank and confidence. The abstract output is normally only a class label; in the rank output, several class labels are given in a particular order of preference (ranked); with confidence output, the classifiers assign a value that measures how confident they are that the input belongs to a given class.
- Some combining techniques can be trained to improve performance, while others are static. Examples of static techniques are the voting methods. The group of trainable techniques includes the use of an MLP that receives the outputs from the several experts and combines these to produce the final output (two examples of such approach are in [16, 95]).

In the rest of this section the combining methods will be grouped by their most unique feature, while reference to the other features will also be made. The methods will be grouped into static, adaptive, methods that generate new classifiers and architectural methods.

2.2.3 Static methods

These are methods that are not trainable, i.e., it is not possible to improve them during the training of the classifiers. They have rigid definitions that do not contemplate the possibility that some of their parameters or characteristics be altered by information gathered from the training data.

Voting rules

The most commonly used combining rule is voting (majority, unanimous). This rule treats classifiers equally without consideration to differences in their performance. That is, it does not take into account the quality of each vote as, for instance, the Dempster-Shafer method (see 2.2.4). Voting expects the classifiers to output a label. If the classifiers output a rank one can consider the class with top rank to be the output and use voting. Also, if the classifiers output measures of confidence, such as posterior probabilities, we can consider the output of the classifier as the class with the largest measure of confidence and apply voting to combine the decisions.

In [69] a weighted majority vote is proposed. The weights are obtained using a genetic algorithm.

A generalization of the majority vote is the Borda count [51]. The Borda count for a class c_j is the sum of number of classes ranked below c_j by each classifier. The magnitude of the Borda count for a given class measures the strength of agreement by the classifiers that the input pattern belongs to that class. The pattern is classified as belonging to the class with largest Borda count. For a two class problem it is equivalent to the simple majority vote.

Sum, product and order rules

If the classifiers estimate the posterior probabilities for the different classes, the following rules can be applied for combining their results: minimum, maximum, median, simple average (or the sum), and product [95]. This can be accomplished using the expression 2.11 and replacing the function $G(\cdot)$ with the desired rule. These rules have been used extensively in the literature [3, 4, 76, 91, 93, 96].

According to [87], the product should be used instead of the average when the estimation of the classifiers presents small errors. But the product rule is very sensitive to estimation errors; when one of the classifiers outputs a probability density of zero then the combination of all the outputs will be zero. A modification to the product rule was proposed in [7, 8] to avoid this problem, adopting the following procedure in a problem with N classifiers. For each class, count the number z of classifiers with an output below a threshold t. If $0 < z < N/2$ the classifiers that have the output smaller than t will have it modified to exactly t. Otherwise nothing is altered. The threshold is found empirically.

The relation between the sum and product rule is studied in depth in chapter 4.

Logistic regression

Logistic regression is a variation of ordinary regression, useful when the observed outcome is restricted to two values, which usually represent the occurrence or non-occurrence of some outcome event, (usually coded as 1 or 0, respectively). It produces a formula that predicts the probability of the occurrence as a function of the independent variables. Logistic regression can be used to combine decisions, using the information from all classifiers. In this case, each classifier outputs a class rank. For a problem with N classifiers, and for a given pattern, a binary variable Y_c is associated to each class. Y_c has the value 1 if c is the true class for that pattern, and zero otherwise. A vector $\mathbf{x} = (x_1, \ldots, x_N)$ containing the rank scores of all classifiers for a given class, is built. The goal is to estimate $\pi(\mathbf{x}) = P(Y_c = 1|\mathbf{x})$. The logit transform of $\pi(\mathbf{x})$ given by $log(\pi(\mathbf{x})/(1 - \pi(\mathbf{x})))$ is approximated by a linear function $\alpha + \sum_{i=1}^{N} \beta_i x_i$. The parameters α, β_i are then estimated using maximum likelihood or weighted least-squares. The recognition is the prediction of the value of Y_c for each class. This method converts the ranks from several classifiers into confidence values (the logit values) for each class. The final decision can be made by choosing the class with highest confidence [51].

Parallel and sequential combination

In [16], a parallel and a sequential combination schemes are tested on handwritten digit recognition tasks. Each of the classifiers outputs a confidence vector with one entry per class. These confidence values belong to $[0, 1]$. There is also a binary signal that is used by the classifier to indicate that it is not able to reliably classify the digit. For the parallel combination, a simple feed-forward neural network (FFNN) with one hidden layer is used. All classifiers are consulted. In the sequential combination, the classifiers are arranged in a chain. The classifiers are combined using another FFNN. A given classifier is consulted only if the combined output of the previous classifiers suggests the rejection of the digit. With this design the combination network cannot recover from substitutions. The sequential combination yields slightly better results than the parallel combination and has a much lower computational complexity. The authors also propose the possibility of boosting the sequential combination, by modifying the classifiers later in the chain using re-sampled versions of the training set. They tested it and obtained the best results. The idea is to train the later classifiers on the digits that were rejected by the first classifiers. This is done at the expense of classification results for the easiest digits, but these were already classified by the first classifiers.

Combination using fuzzy integral

Consider N classifiers that output estimates of the posterior probabilities. Identify each classifier as g_i for $i = 1, \ldots, N$. Let $Y = \{g_1, ..., g_N\}$ and $h : Y \rightarrow [0, 1]$ be a function. The classifiers are labeled such that $h(g_1) \geq h(g_2) \geq \cdots \geq h(g_n)$. The values of $h(g_i)$ are the estimates of the posterior probability of classifier i for a given class. A density d_i is assigned to each classifier which is interpreted as a measure of its importance. These measures of importance can be defined by a human expert or can be induced from the data. The d_i's define the fuzzy density mapping. The knowledge of the d_i allows the calculation of λ by solving

$$\lambda + 1 = \prod_{i=1}^{N} (1 + d_i \lambda) \tag{2.13}$$

and finding the only root greater than -1. λ allows the construction of a d_λ-fuzzy measure, d. This is used to calculate the following fuzzy integral, e, with respect to the fuzzy measure d over Y,

$$e = \max_{i=1}^{N} (\min(h(g_i), d(A_i))) \tag{2.14}$$

where $A_i = \{g_1, ..., g_i\}$. Given that d is a d_λ-fuzzy measure, the values of $d(A_i)$ can be calculated recursively from

$$d(A_1) = d(\{g_1\}) = d_1 \tag{2.15}$$

$$d(A_i) = d_i + g(A_{i-1}) + \lambda d_i d(A_{i-1}), \ 1 < i \leq N. \tag{2.16}$$

The fuzzy integral is calculated for each class. The class with the largest value is the chosen one [29].

This method could be transformed into an adaptive one if the densities d_i were recalculated for each input (one such approach in [96]).

Automatic design of multi-classifier systems

In [44], an automatic method for the design of multi-classifier systems (ADMS) is proposed. The main idea is to select from a set of classifiers a subset containing accurate classifiers that make different errors. This is done in two steps: first an identification of M disjoint subsets is obtained using a hierarchical clustering algorithm. Each subset contains highly error-correlated classifiers, i.e, the classifiers make similar errors. Then a selection of one classifier from each of the M subsets is done. This involves evaluating candidate sets of M classifiers, one from each subset, that were produced during the clustering phase, on a validation data set. The classifiers of the optimal set are then combined by voting.

Parallel consensual neural network (PCNN)

In the PCNN model [17], input data is transformed several times and fed into different NNs, called stage NNs (SNNs). The final output is based on consensus among the SNNs trained on the same original data but with different representations. Each SNN has the same number of output neurons, which is equal to the number of data classes. The SNNs are trained for a fixed number of iterations or until the training procedure converges. The outputs of the SNNs are weighted for the final decision. The authors propose two alternatives for the weights: the first is to give equal weights for all SNNs (average) or use a reliability measure which ranks the SNNs according to its goodness. The second is to find the best weights in an MSE sense.

2.2.4 Adaptive methods

In the adaptive methods, the adaptation can be at two different levels: for each sample, deciding which classifier or which decision combination function to use.

Weighted average combination

Several authors have studied the linear weighted combination of classifiers. The idea is to assign a different weight to the prediction of each classifier. The methods normally differ on the process of finding these weights [2, 46, 49, 55, 75, 96, 94].

The weighted average combination can be considered adaptive since some of the classifiers can be excluded from the final decision by receiving a zero weight.

They can also be seen as trainable methods since the weights are obtained for each data point from estimates based in the training set, such as in the covariance-based optimal combination method, described below.

Some results relative to the use of the weighted average combination are presented in chapter 4.

Dynamic classifier selection by local accuracy

The idea presented in [100] consists on estimating each classifier's accuracy in local regions of feature space surrounding an unknown test sample, and then use the decision of the most locally accurate classifier. The local regions are defined in terms of the k-nearest neighbors (k-NN) in the training data. To estimate the accuracy of the classifiers the authors propose two criteria: the first is the percentage of training samples in the region that are correctly classified - they call this the overall local accuracy. The other is to estimate local accuracy with respect to some output class. This is done by determining the percentage of local training samples assigned to a given class by a classifier that were correctly labeled. This is called the local class accuracy.

Behavior-knowledge space

This method aggregates the decisions obtained by individual classifiers using a Behavior-Knowledge Space (BKS) which can concurrently record the decisions of all classifiers on each learned sample [54]. The BKS is a N-dimensional space where each dimension corresponds to the decision of one classifier. The intersection of the decisions of individual classifiers references one unit of the BKS. Each unit accumulates the number of training samples from each class. To classify a test pattern, the decisions of the classifiers index one unit of the BKS and the pattern is assigned to the class with more training samples in that unit [100].

Dempster-Shafer theory of evidence

This is a generalization of the Bayesian theory and allows the representation of incomplete knowledge. It allows belief assignment to subsets, as well as to individual elements of the hypothesis set. If $P(A)$ represents the probability that the event A is true, then $P(A)$ plus $P(not\ A)$ may not sum to one [54]. It takes the output of several classifiers in confidence level and transforms it in proximity measures of the outputs from a mean vector (calculated using the training data set) for each class and classifier. These proximity measures are then used to build measures of evidence that are combined to yield the evidence for each class given an input pattern. The final classification decision is obtained by assigning the pattern to the class with higher evidence [81].

Mixture of local experts

The idea in [56] is to make a modular version of a supervised multi-layer network. The problem is learned simultaneously by several networks. The final decision is produced by only one of the experts. The decision of which of the experts is going to produce the final classification is made by a gating network that receives the same input as the expert networks. This network, depending on the input, will choose one of the experts to produce the final classification. The gating network learns during training which expert network behave best for certain region of the input space.

Hierarchical mixture of experts

In [57], a hierarchical mixture of experts architecture is described. This architecture contain multiple NNs organized in tree structures. These trees contain two different types of networks: expert networks at the leaves and gating networks in the branches. The tree architecture starts with a complex structure which is pruned on the basis of quantities computed via Gibbs sampling (a general method for probabilistic inference), until a tree-structure is found that matches the data adequately. Inputs that carry useful information are separated from the others using Bayesian hypothesis testing. This is a generalization of the mixture of local experts.

Feature based decision aggregation in MNN

In [97], an architecture for combining decisions is proposed in which the combining rule adapts to the particular pattern, by adjusting the weights given to the output of each classifier. The architecture is composed of several modules: classifiers, error analysis, modification and detector modules (this method could also be placed under the 'architectural methods'). The output of the classifiers is monitorized by an error analysis module which tries to compensate possible systematic errors. Modification modules alter the classifiers output such that all inputs to the aggregation procedure are normalized. The detector modules collect information from the feature space in order to adapt the aggregation procedure to the data. The aggregation procedure uses the modified classifier outputs and the detector outputs to reach the final decision.

Bayesian combination rule

This method takes into consideration for the final decision, the performance of the classifiers on the training set.

Consider N classifiers that output class labels. The problem has L classes plus a reject option. Each classifier g_k has a $L \times (L+1)$ confusion matrix (for the training set), $C^{(k)}$, where $C_{ij}^{(k)}$ is the number of patterns belonging to class i that were classified as belonging to class j, when $j \leq L$. If $j = L$ it represents the number of patterns belonging to class i that were rejected. An estimate for the conditional probability that a pattern x belongs to class i given that the classifier k assigns it to class j is given by

$$P(x \in c_i | e_k(x) = j) = \frac{C_{ij}^{(k)}}{\sum_{i=1}^{L} C_{ij}^{(k)}}. \tag{2.17}$$

Given that for a pattern x the classification results by the classifiers is $e_k(x) = j_k$ for $1 \leq k \leq N$, a belief value that x belongs to class i can be defined as

$$bel(i) = P(x \in c_i | e_1(x) = j_1, \ldots, e_N(x) = j_N).$$

Using the Bayes formula and assuming independence of the classifiers' decisions this can be approximated by [101]

$$bel(i) = \frac{\prod_{n=1}^{N} P(x \in c_i | e_n(x) = j_n)}{\sum_{i=1}^{L} \prod_{n=1}^{N} P(x \in c_i | e_n(x) = j_n)} \tag{2.18}$$

for $1 \leq i \leq L$.

Given an input pattern x, the values $e_n(x) = j_n$ are produced by the classifiers, the estimates in equation 2.17 are used in equation 2.18 and the pattern is classified in class j if $bel(j) \geq bel(i)$ for all $i \neq j$ and $bel(j) > \alpha$. α is a threshold. x is rejected if this is not true or if $e_k(x) = L+1$ for all k [69].

Covariance-based optimal combination

In [75], a method for the calculation of the covariance between different NNs is proposed. It is based on a generalization of the delta method for calculating the Hessian of the network (which is the matrix of second derivatives of the network error relative to the network weights [18]). Using the inverse matrix of this estimate of the covariance matrix between the networks, the authors calculate a pointwise covariance between the networks' predictions. After a normalization procedure with respect to the variance of each network, an estimate of the correlation between networks at each point of the input space is obtained. These estimates use only data from the training set. Different network outputs are combined using a weighted combination, where the weights are 'covariance-based'. This weighted combination is optimal in the sense of minimum expected prediction error if the individual networks are unbiased.

2.2.5 Methods that generate new classifiers

These methods manipulate the training set in order to generate different classifiers which are then combined to form a composite classifier [76].

Option trees

An option tree [25] is a combination of several trees. It has leaf nodes and test nodes as a normal tree. But some of the test nodes are built from several test nodes using a XOR function. Only one test is chosen from the available tests in the node. Each option is a test together with a set of outcome subtrees. To obtain class probabilities for an unclassified pattern one has to pass the example down several branches at once and weight up the resulting probability vectors found at the nodes. The classification process is the averaging of the class probabilities across all nodes traversed, where the class probabilities are stored at each node. A subtree of an option tree can be pruned by turning subtrees to leaves and by removing options from each node so that at most one option is left. This way, a pruned subtree is a normal tree. The author in [25] presents an algorithm for growing option trees, a formula for traversing the tree and for calculating the classification of a pattern.

Bagging

Bagging (bootstrap aggregation) produces replications of the training set by sampling with replacement. Each replication of the training set has the same size as the original set, but some examples can appear more than once while others don't appear at all. A classifier is generated from each replication. All classifiers are used to classify each sample from the test set using a uniform voting scheme [42]. Bagging is applicable to a wide variety of classifiers and is based on the theoretical analysis of the behavior of the composite classifier [76].

The complete procedure is now described.

- Consider a training set TS with N_p cases.
- Define the probability of the nth sample in the training set as $P(n) = 1/N_p$.
- Now sample Np times from the distribution $P(n)$.
- Sample from TS with replacement. This way a re-sampled training set TS' is built. TS' is normally called a bootstrap sample from TS.
- Repeat this procedure to construct a sequence of several independent bootstrap training sets.
- Construct a corresponding sequence of classifiers by using the same classification algorithm applied to each of the bootstrap training sets.
- For the final classification each one of these classifiers votes for each class [21].

In [89] a variation of bagging, called dagging is described. Instead of using bootstrap samples to build the different classifiers, disjoint sets are used. A mixture between bagging and stacked generalization (see below) is also proposed (as in [99]).

Arcing

Arcing stands for 'adaptively re-sample and combine'. It is more successful than bagging in test set error reduction. As in bagging, several classifiers are constructed and vote for the final classification. But now the construction of classifier k depends on the results of the $k-1$ previous classifiers: the construction is sequential. To design a classifier, a probability distribution $P(n)$ exists for each case in the training set TS. A new training set TS' is constructed by sampling N_p times from this distribution. The probabilities are updated depending on how the classifier created with the TS' training set performs on the original training set TS. A factor $\beta > 1$ is defined which depends on the misclassification rate. If the accuracy increases so does β. If the nth case in TS is misclassified by the classifier built on TS' then assign the weight $\beta P(n)$ on that case, otherwise the weight remains $P(n)$. To obtain the new probabilities for the next training set TS'', divide each weight by the sum of the weights. The final classification decision is based on a weighted voting from the created classifiers.

The idea is that the most likely points to be selected for the replicated data are those that are more easily misclassified. The classifiers should focus on the 'hard' points.

Further details can be found in [21], where the details of β updating and the weights used for the final vote are described. Note that Breiman [22] considers boosting (see below) and the combination of weak classifiers (see below) as arcing methods.

Boosting

Boosting was proposed in [83], but the most common version (AdaBoost) was presented in [37]. The method's description follows.

Consider a training set with N_p instances. At each repetition or trial $t = 1, 2, \ldots, T$, instance i has weight w_i^t, where $w_i^1 = 1/N_p$ for all i. For each trial do:

- construct classifier g^t from the training instances using weights w_i^t;
- the error rate of g^t on the training data (ϵ^t) is the sum of the weights w_i^t of the misclassified instances;
- if $\epsilon^t = 0$ or $\epsilon^t \geq 1/2$ the process terminates;
- otherwise the weights w_i^{t+1} for the next trial are set so that the error rate of g^t on the training set using the weights w_i^{t+1} is exactly $1/2$.

The final composite classifier g^* is built combining the g^t classifiers by voting.

Freund and Schapire proved that the error rate of g^* on the unweighted training instances approaches zero exponentially quickly as T increases [77]. In [76] the author concludes that much of the benefits of boosting are caused by over-fitting the training data set. He also says that although boosting generally increases accuracy, it leads to a deterioration on some data sets. Breiman [21] notes that boosting forces the classifier to have a 50% error rate on the re-weighted training instances which causes the learning system to produce a quite different classifier on the following trial. This leads to an extensive exploration of the classifier space. Quinlan [77] did some experiments with boosting C4.5 (a decision tree [78]) and found that the advantages of boosting are lost if there is non-trivial classification noise in the learning sets. This is unexpected since as boosting uses a combination of classifiers it should be robust in the presence of noise. He also believes that the reason why boosting increases the classification accuracy may be unrelated to its convergence properties on the training data. Breiman [21] also says that boosting failure happens more likely with relatively small data sets.

Combination of weak classifiers

A weak classifier is a classifier that can do a little better than random guessing. They are used because they are cheap and easy to obtain. A new weak classifier is created by random sampling the classifier space of all possible linear classifiers [62].

The following method is used to partition the training set. Suppose a combined classifier consists of k weak classifiers. To generate a new weak classifier the training set (N_p samples) is partitioned in two: M samples which contain all the misclassified samples so far and a small fraction of correctly classified samples; and the rest of the samples ($N_p - M$). To select a new weak classifier only the set with M samples is used. The idea behind this partitioning of the training set is to force the new classifier to learn the samples that have not been learned by the existing classifiers. The combined classifier is built using the simple majority vote of the weak classifiers.

2.2.6 Architectural methods

This section describes several methods that develop an architecture for combining the classifiers. These include several NN architectures based on modular networks [12, 53].

Stacked generalization

Stacked Generalization was proposed in [98]. It can be used for two purposes: to combine generalizers (classifiers) or to improve a single generalizer.

Consider a data set with N_p instances (x_i, c_i) where x_i represents the attributes of the ith instance and c_i is a class label. Randomly split the data set into J almost equal parts, $L_1, \ldots, L_J$. Define L_j and $L^{(-j)} = L - L_j$ to be the test and training sets for the jth fold of a J-fold cross-validation. Given N learning algorithms, called *level-0 generalizers*, invoke the nth algorithm on the data in the training set $L^{(-j)}$ to create a model $M_n^{(-j)}$, for $n = 1, \ldots, N$. These are the *level-0 models*. For each instance x_i in L_j (the test set for the jth cross-validation fold), let z_{ni} denote the prediction of the model $M_n^{(-j)}$ on x_i. After the cross-validation, the data from the N models, which is called *level-1 data* is

$$L_{CV} = \{(c_i, z_{1i}, \ldots, z_{Ni}), i = 1, \ldots, N_p\}.$$

Now use a learning algorithm, *the level-1 generalizer* to derive from this data a model $\tilde{M}$ - this is the *the level-1 model*. The classification uses all $N+1$ models: the M_n, $n = 1, \ldots, N$ and $\tilde{M}$. For a new instance, the M_n produce a vector $(z_1, \ldots, z_N)$ which is the input to $\tilde{M}$, whose output is the final classification [88].

In [88], an extension to this method is used in which the output from the level-0 models is a set of class probabilities instead of just a single class prediction. A variant with three levels is proposed in [38].

Cascaded generalization

Cascaded Generalization [42] consists of a reformulation of the input space by the insertion of new features. More specifically, it starts with a training set and a base classifier. The classifier must output a vector of conditional probability distribution $[p_1, \ldots, p_L]$, where p_i represents the posterior probabilities, $P(c_i|x)$, for a data point x and a class i. After the application of the base classifier to the training set each data point is augmented with its vector of conditional probabilities. Thus a new training set is created with the same number of samples but each sample has more features. Now a different classifier will be used on this new training set. This process can be repeated for N classifiers.

This is a sequential composition of classifiers. The author also suggests in [42] the parallel composition of classifiers, where at each level, instead of just one classifier acting on a training set, a group of classifiers could be used.

Cooperative modular neural network

This method divides the feature space using an unsupervised NN, therefore creating several clusters (groups) [11]. A modular neural network (MNN) is used on each group and any supervised algorithm can be used as a learning scheme. Each group can include several classes. Each MNN is trained using information from the other groups too. The output layer of each MNN consists of class outputs c_i representing each class in the group, and group outputs V_i, for each of the remaining MNN.

If a training sample is in one of the module's class, its corresponding output is high $c_i = 1.0$, and all other outputs for the other classes in the group and for the other groups are low $(c_j = 0.0)$.

When a test sample is used the outputs are never strict ones and zeros, but instead vary in the interval $[0, 1]$ according to how near the sample is to the decision boundaries.

Applying a voting scheme that considers the preferences of the different modules gives more importance to the votes of the modules that have less classes. This happens because with the decrease in the number of classes there is an increase in the output values for the existing classes. The authors argue that this is not a bad characteristic since the smaller modules tend to be more accurate, as there is a smaller coupling between their weights.

To prevent single-class groups from manipulating the results they added a rule that prevents having single-class groups for more than 50% of the number of modules.

This architecture is studied with detail in chapter 5.

2.2.7 Conclusions

Table 2.1 summarizes the different characteristics of the reviewed combining methods.

We conclude that, in general, the combination of the output of several classifiers reduces errors in classification when compared to isolated classifiers. But there are cases when a combination of classifiers may yield worst results than a combination of a subgroup of those classifiers [81, 100]. It would be very important to find out when do these situations occur to avoid the unnecessary and even damaging use of a combination of classifiers.

In [100], it is suggested that the greatest potential for classifier combination exists when large data sets with data distribution that are too complex for most individual classifiers, are used.

When possible, not only the opinions of the selected classifiers should be combined but also the certainty of the classifiers in their decisions [13]. This way, more of the produced information is used [42] and a more confident decision can be made. A framework for such type of combination can be the Dempster-Shafer theory.

It is very important to choose which classifiers to use. Several results support this belief. For instance, [81] states that the best results are not necessarily obtained by combining the best individual classifiers. The author says that it seems to be more important to combine 'independent' classifiers than those with better performance.

It is interesting to see that normally the authors are not concerned with the computational cost of their methods - an exception is [62] where ways to combine weak classifiers with polynomial space- and time-complexity are presented.

In terms of applications referred in the articles, the recognition of handwritten digits is by far the most common [16, 28, 51, 54, 65, 95].

An innovative way to increase prediction accuracy is proposed in [23], that consists of generating more data from the existing data so that the feature space becomes more densely populated and the classifier generalizes better. It seems to be a good solution when training data is not abundant.

Finally, it is interesting to see that for Jain and Mao [60] the fundamental challenges in pattern recognition are in pattern representation rather than in classification alone. A similar view is also presented in [43] where the authors say that the fundamental challenges for neural models are representation rather than learning. Perhaps we have reached a point in classification that it is easier to improve accuracy by refining feature extraction than by trying to improve the classifiers.

2.3 Summary

The kind of problems we consider in this thesis are classification problems. As we have already seen, these involve the assignment of an input pattern to one of a set of predefined classes. The combination of opinions from several classifiers can be considered for a different type of problems: the regression problems. These are function approximation problems where the input can be assigned to any value in a real interval. We are not going to consider this type of problems.

Finally, we wish to stress that the framework we are dealing with has the following characteristics:

- we are considering a classification and not a regression problem;
- we are not considering the reject option, i.e., all feature vectors are to be classified;
- classifiers output approximations to the posterior probabilities, except if otherwise stated, as it will happen when we mention some voting rules;
- the loss function is the one in expression 2.3 except if otherwise stated, as is the case in chapter 6;
- when using NNs, we will use MLPs trained with resilient back-propagation; other classifiers, such as k-nearest neighbors (k-NN) will also be used.

In this chapter the notation was introduced along with basic probability definitions. The basic notions required for the understanding of the following chapters were also presented. These include the use of NNs for estimating posterior probabilities and the general form used to combine these estimates in order to produce a classification result.

A literature review of methods used to combine the opinions of classifiers along with a summary table were also included.

Table 2.1: Characteristics of the combination methods

Method	Gen. new	Architecture	Trainable	Adaptive	Outputs	References
ADMS	N	parallel	N	N	abstract	[44]
Arcing	Y	serial/parallel	N	N	abstract	[21, 22]
Bagging	Y	parallel	N	N	abstract	[22, 76, 89]
Bayesian combination	N	parallel	N	Y	abstract	[69, 96, 101]
BKS	N	parallel	N	Y	abstract	[54, 100]
Boosting	Y	parallel	Y	Y	abstract	[16, 35, 76]
Borda count	N	parallel	N	N	rank	[51, 96]
Cascaded generalization	Y	serial or serial/parallel	Y	N	confidence	[42]
CMNN	N	parallel + gate	Y	Y	confidence	[11, 12, 13]
Comb. weak classifiers	Y	parallel	Y	N	abstract	[62]
Dempster-Shafer	N	parallel	N	Y	confidence	[27, 81, 101]
Dynamic classifier selection	N	parallel	N	Y	abstract	[100]
Feat. based agg.	N	parallel + ...	Y	Y	confidence	[97]
Fuzzy integral	N	parallel	N	N/Y	confidence	[29, 96, 101]
Hierarchical m. experts	Y	tree	Y	Y	confidence	[57]
Logistic regression	N	parallel	N	N	rank	[51, 61]
Mixture of experts	N	parallel + gate	Y	Y	abstract	[56]
Option trees	Y	tree	Y	N	confidence	[25]
PCNN	N	parallel	N	N	confidence	[17]
Stacked generalization	Y	parallel+serial	N	N	abstract	[98]
Sum, product, order	N	parallel	N	N	confidence	[3, 4, 66, 76, 91, 93, 96]
Voting	N	parallel	N	N	abstract	[9, 13, 15, 27, 38, 62, 96]
Weighted sums	N	parallel	Y	Y	confidence	[2, 46, 49, 55, 75, 94, 96]

The next chapter presents the background on error evaluation and error bounds. It also introduces new bounds for the empirical error. These are based on an existing result for the proximity among distributions.

Chapter 3

Classification error estimation

This chapter focuses on the measurement and estimation of the classification error. We distinguish here between the true error that a classification problem, defined by a training and test set along with a classifier, implies, and the empirical error which is an estimate of the true error. In section 3.1, methods for the determination of estimates of the error of a classifier are presented.

When it is not possible to estimate directly the classification error or only an idea of what it may be, without running the actual tests, is desired, bounds for the error can be used. In section 3.2 several bounds will be presented. Section 3.3 introduces new upper and lower bounds for the classification error.

3.1 Error counting methods

In this section we analyze how can the performance of a classifier be evaluated in practice.

The most basic method to determine an estimate for the error of a classifier is the resubstitution method. This consists of using the same data set for training and testing the classifier. The resultant estimate will normally be optimistic and will not extend to a different data set.

To avoid the optimistic bias of the resubstitution method, cross-validation methods should be used. These methods are all based on the same idea: there should be no common data in the learning and test sets. One of these methods is the holdout method. It consists of partitioning the data set in two independent subsets - one is used for training and the other for testing. This gives an estimate for the error. Another estimate can also be obtained by averaging the values obtained by considering this definition of the sets and the reverse definition: using the original training set as the new test set and using the original test set as the new training set.

Another method for evaluation of the performance of a classifier is the leave-one-out method. For an N instances data set, N partitions with $N-1$ instances are created. The classifier is then evaluated on the instance that was left out of each partition. The N performance results are averaged. This method gives a better estimate of the true performance than the previous one since the classifier is trained with almost all of the data set. This method is a special case of the more general leave-k-out method which is also called balanced m-fold cross-validation [80], with $m = N/k$. The term 'balanced' is used because, generally, the m-fold cross-validation does not require that the partitions have the same size.

The leave-k-out method, instead of using only one instance for testing, uses k. The remaining $N-k$ instances are used for training. This process is repeated N/k times until all the partitions have been used for testing. The final estimate is the average of the individual N/k results.

Note that this generalizes both the leave-one-out method (setting $k = 1$) and the holdout method (setting $k = N/2$). It also reduces the bias of the holdout method and the computational effort of the leave-one-out method [20].

How should the value of k be chosen ? According to [80], using a small value for k gives rather variable estimates but also the most accurate. Large k gives larger bias but smaller variance. A compromise must be found and should take into account the actual size of the data set used.

A method that creates new sets is the bootstrap method [20, 70]. Several (typically between 50 and 200 [86]) learning sets are obtained by sampling with replacement from the original data set. These are used for several experiments where the test set is the original data set. The results are averaged. The performance estimate of this method has been demonstrated to be statistically equivalent to the leave-one-out method [39] and the variance similar to the resubstitution method [82].

The main characteristics of these methods are in table 3.1.

In this thesis, the leave-k-out method will be used to estimate the empirical

Table 3.1: Error counting methods for a data set with N points.

Methods	Train Points	Test Points	Tests
Resubstitution	N	N	1
Holdout	$N/2$	$N/2$	1 or 2
Leave-one-out	N-1	1	N
Leave-k-out	N-k	k	N/k
Bootstrap	$m < N$	N	$> 50, < 200$

classification error and this value will be given along with the standard deviation that results from the N/k independent tests made.

3.2 Error bounds

In this section we present several bounds for the classification error. These bounds can be used to give an idea of how close are the empirical error and the true error.

The following theorem introduces four probabilistic inequalities [64].

Theorem 3.1 *Let $X_1, \ldots, X_m$ be a sequence of m independent Bernoulli trials, with probability of success $E[X_i] = p$. Let $S = X_1 + \cdots + X_m$ be a random variable indicating the total number of successes, so $E[S] = pm$. Then, for $\gamma \in [0, 1]$, the following bounds hold*

$$P(S > (p + \gamma)m) \leq e^{-2m\gamma^2} \tag{3.1}$$

$$P(S < (p - \gamma)m) \leq e^{-2m\gamma^2} \tag{3.2}$$

$$P(S > (1 + \gamma)pm) \leq e^{-\gamma^2 pm/3} \tag{3.3}$$

$$P(S < (1 - \gamma)pm) \leq e^{-\gamma^2 pm/2} \tag{3.4}$$

These bounds are normally called Chernoff bounds. The first two inequalities are called the additive form and are credited to Hoeffding. The other two are the multiplicative form and are credited to Chernoff.

They can be used in a classification context to provide bounds for the error, as we will now discuss.

The outcome of a classification experiment is a random variable with Bernoulli distribution. It can only have two outcomes: correct or incorrect classification.

The outcome of each experiment will be denoted X_i and will be one if the result was incorrect and zero otherwise. The probability of error will be $E[X_i] = p$.

We define $\widehat{error}$ as the random variable that indicates the proportion of errors, that is, $\widehat{error} = S/m$. This is known as the empirical error and is an unbiased estimate of the true error, since $E[\widehat{error}] = p$. The bounds in theorem 3.1 can now be restated for $\widehat{error}$

$$P(\widehat{error} > (p+\gamma)) \leq e^{-2m\gamma^2} \tag{3.5}$$

$$P(\widehat{error} < (p-\gamma)) \leq e^{-2m\gamma^2} \tag{3.6}$$

$$P(\widehat{error} > (1+\gamma)p) \leq e^{-\gamma^2 pm/3} \tag{3.7}$$

$$P(\widehat{error} < (1-\gamma)p) \leq e^{-\gamma^2 pm/2} \tag{3.8}$$

All these bounds involve the true error probability p in the event to be bounded. This means that, in general, they give the probability that the empirical error is at a certain distance from the true error.

A lower bound for the error can be obtained, for certain values of the error, using the following inequality (from [34])

Lemma 3.1 (Slud's inequality) *Let B be a binomial (m, p) random variable with $p \leq 1/2$ and suppose that*

$$mp \leq k \leq m(1-p). \tag{3.9}$$

Then,

$$P(B \geq k) \geq P\left(Z \geq \frac{k - mp}{\sqrt{mp(1-p)}}\right) \tag{3.10}$$

where Z is a normal random variable with zero mean and unit standard deviation.

We can rewrite this inequality to establish a lower bound for the empirical error, as follows.

Consider a classification problem with a theoretical error probability $p \leq 1/2$. Consider also a constant $\alpha = k/m$ such that,

$$p \leq \alpha \leq (1-p). \tag{3.11}$$

The estimate of the true error is $\widehat{error} = B/m$. Then, from Slud's inequality, we have

$$P(\widehat{error} \geq \alpha) \geq P\left(Z \geq \frac{\alpha - p}{\sqrt{p(1-p)}}\right) \tag{3.12}$$

In a classification problem, this expression gives a lower bound for the probability that the empirical error is larger than a given constant α, if the true error is less than 50% and the constant α obeys 3.11.

In the next section we introduce new upper and lower bounds for the probability that the empirical error is larger (or smaller) than a fixed value α. This means that these new bounds do not involve the true error probability in the event to be bounded as the above mentioned Chernoff's and Hoeffding's bounds, and they do not restrict, *a priori*, the range of values of α for which they are valid, as in the lower bound based in Slud's inequality.

3.3 New error bounds

In this section we propose new upper and lower bounds for the empirical error, that are based on Sanov's theorem.

The outcome of a classification experiment, as we have referred, is a random variable with a Bernoulli distribution. The outcome of a series of such experiments, as are normally done in classification problems, is a random variable with binomial distribution.

Bounds such as Hoeffding's [34] (or the additive form of Chernoff's bounds [64]) are established for the difference between the empirical error and the true error, i.e., bound the following probability $P(\widehat{error} - \textit{true error} \geq \alpha)$ (or $P(\widehat{error} - \textit{true error} \leq \alpha)$) with $\alpha \in (0, 1)$. These involve the true error and are only upper bounds. Combinations of absolute and relative error such as the bound by Haussler [80] also involve the true error.

Sanov's theorem [104] allows the establishment of bounds on the probability that a given distribution belongs to a set of distributions.

It will be used for establishing upper and lower bounds on $P(\widehat{error} \geq \alpha)$.

The next section introduces the mathematical results that allow the establishment of the above mentioned bounds. Section 3.3.2 shows how those bounds can be used in the classification context and in section 3.3.3 a discussion on the applicability of the bounds is presented. Section 3.3.4 contains the conclusions.

3.3.1 Sanov's theorem

Sanov's theorem is the foundation for the rationale proposed here, enabling the definition of upper and lower bounds on the probability that a distribution belongs to a set of distributions. This theorem needs the definitions of the Kullback-Leibler divergence [104] and type, given below.

Definition 3.1 (Kullback-Leibler divergence) *Given two probability density functions, $P_A(x)$ and $P_B(x)$, the Kullback-Leibler divergence $D(P_A||P_B)$ is defined by*

$$D(P_A||P_B) = \sum_x P_A(x)\log_2 \frac{P_A(x)}{P_B(x)}.$$

This is a measure of distance between two probability distributions. In fact, it is not a distance metric because it lacks the commutativity of its arguments, i.e., in general $D(P_A||P_B) \neq D(P_B||P_A)$.

Let $X_1, \ldots, X_N$ be a sequence of symbols from an alphabet $H = \{a_1, \ldots, a_m\}$, with m different symbols.

Definition 3.2 (Type (empirical distribution)) *[31] The type $P_{\mathbf{X}}$ (or empirical probability distribution) of a sequence $\mathbf{X} = X_1, \ldots, X_N$ is the relative proportion of occurrences of each symbol of H, i.e., $P_{\mathbf{X}}(a) = K(a|\mathbf{X})/N$ for all $a \in H$, where $K(a|\mathbf{X})$ is the number of times the symbol 'a' occurs in the sequence $\mathbf{X} \in H^N$.*

Theorem 3.2 (Sanov's theorem) *Let $X_1, ..., X_N$ be i.i.d. from a probability distribution $P_s(x)$ with support size J and E be any closed set of probability distributions. Let $P(T \in E)$ be the probability that the type T of a sample sequence lies in the set E. Then:*

$$(N+1)^{-J} 2^{-ND(P^*||P_s)} \leq P(T \in E) \leq (N+1)^{J} 2^{-ND(P^*||P_s)}$$

where $P^* = \arg\min_{T \in E} D(T||P_s)$ *is the distribution in* E *that is closest to* P_s *in terms of the Kullback-Leibler (K-L) divergence.*

Proof. The proof can be found in [104].□

The use of this theorem can be illustrated with a very simple example from [31], that will provide the motivation for the application to the classification error.

Suppose we have a fair coin and want to estimate the probability of having more than 700 heads in 1000 tosses.

The set E consists of the probability distributions for which $P(heads) \geq 0.7$ with $P(heads) + P(tails) = 1$. The distribution that generates the samples is $P_s(heads) = P_s(tails) = 0.5$ since the coin is fair. The closest distribution to P_s in E (in terms of the K-L divergence) is $P^*(heads) = 0.7, P^*(tails) = 0.3$. The corresponding K-L divergence is $D(P^*||P_s) = 0.119$. The alphabet (support) size for this problem is $J = 2$ since there are only two possible outcomes for each toss (heads or tails). Using Sanov's theorem the probability of having more than 700 heads in 1000 tosses is less than $2^{-119} \times 1001^2$.

3.3.2 Application to classification error

In a classification problem, the alphabet size is also $J = 2$ since there is only a correct or incorrect classification, independently of the number of classes, dimension of the input space, etc.

Let E be the set of Bernoulli distributions with error probability larger or equal to α.

Since the *a priori* error rate for a given classification problem (data set and classifier) is unknown, an estimate of P_s can be obtained by evaluating the classifier on a set, thus getting $\hat{P}_s$. This set can be a subset of the set that is used later, when performing the experiment whose result is to be bounded.

There is no *a priori* constraints on the size of this set, but it should be large enough to allow a good estimation of P_s. Its actual size can be found as a function of the proximity of the estimate to the true value of P_s. A formula for the size of a sample when estimating proportions can be found, for instance, in [30].

It is possible to use an estimate for P_s since in the formulation of Sanov's theorem, the true error shows up in the bound and not in the event for which we want to bound the probability, as is usual with other bounds such as Chernoff's.

This is a similar situation to the one we would get if in the coin example the fairness of the coin was not known *a priori* and we made a set of tosses and used the estimated type $\hat{P}_s$ instead of P_s.

The closest distribution to $\hat{P}_s$ in E, in terms of the Kullback-Leibler divergence, is $P^*(error) = \alpha$, if $\alpha \geq \hat{P}_s$, as should normally be the case.

Sanov's theorem can be used to bound the empirical classification error:

Consider E to be the set of Bernoulli distributions with error probability larger or equal to α, make $J = 2$ and use $\hat{P}_s$ as an estimate of P_s. Consider the empirical error to be a type of $\hat{P}_s$.

Let P^* represent a Bernoulli distribution with $P^*(error) = \alpha$ (and $P^*(correct) = 1 - \alpha$). Let $\hat{P}_s$ represent a Bernoulli distribution with $\hat{P}_s(error)$ equal to the empirical error obtained while using a classifier g on a data set A (and $\hat{P}_s(correct) = 1 - \hat{P}_s(error)$).

The probability that the classifier g makes an average error larger than α while classifying N points on data set B, that obey the same distribution as the points in A, is bounded by

$$LB \leq P(\widehat{error} \geq \alpha) \leq UB \tag{3.13}$$

with the lower bound, LB, given by

$$LB = (N+1)^{-2} 2^{-ND(P^* || \hat{P}_s)} \tag{3.14}$$

and the upper bound, UB, given by

$$UB = (N+1)^{2} 2^{-ND(P^* || \hat{P}_s)} \tag{3.15}$$

Note that A may be a subset of B.

It would be possible to use Sanov's theorem to bound $P(\widehat{error} \leq \alpha)$ instead of $P(\widehat{error} \geq \alpha)$, simply by defining the set E as the set of the Bernoulli distributions with error smaller or equal to α. But this case could convey erroneous information since any empirical error is lower bounded by the Bayes error, and it would be possible to set an α smaller than the Bayes error and still get information from the bounds when there should not be any.

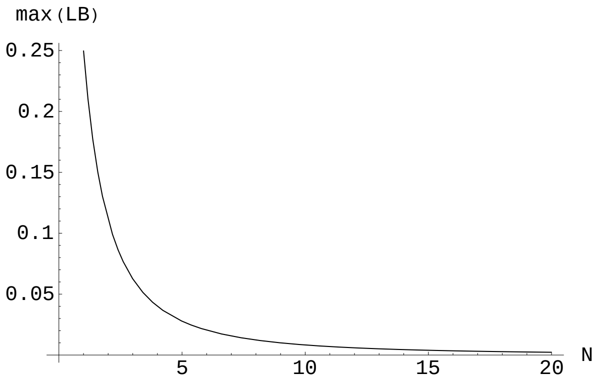

Figure 3.1: The maximum value of the lower bound as a function of N.

3.3.3 Discussion

Lower bound

To study the applicability of the lower bound it is important to find its maximum value for all the possible cases of a classification problem, that is, the maximum of LB for any N, with $P^*, P_s \in (0, 1)$.

Note that LB is a decreasing function of D. To find the maximum of LB we search the minimum of D. Since D is a distance it is easy to see that its minimum value will be zero and it is reached when $\hat{P}_s(error) = \alpha = P^*(error)$.

The maximum value of LB is thus given by $(N+1)^{-2}$. Figure 3.1 represents this function.

We see that this bound does not provide much information if we are using more than 20 points for testing the system: its maximum has a value of approximately 2×10^{-3} for $N = 20$.

We conclude that the applicability of this bound is limited to small values of N (typically smaller than 20).

Upper bound

Figure 3.2 shows UB for $\alpha = 0.35$ as a function of N.

For small values of N, the region of the problem space for which the bound gives useful information (the bound is < 1) is small. Also, the region where the UB does not provide useful information ($UB \geq 1$) is centered around $\hat{P}_s = \alpha$.

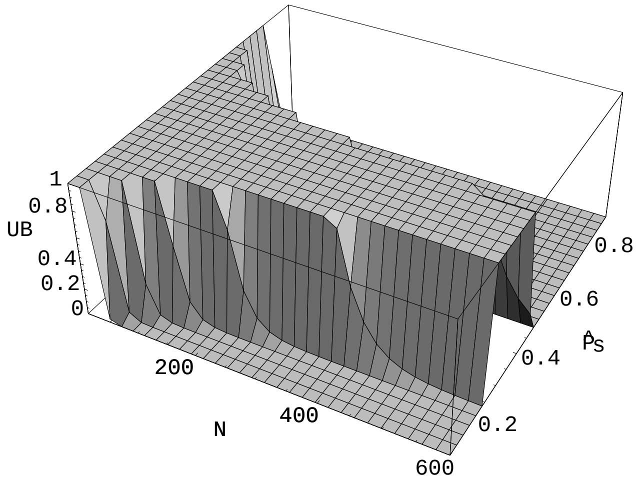

Figure 3.2: The upper bound as a function of N and $\hat{P}_s$ for $\alpha = 0.35$.

This makes the applicability of the upper bound specially adequate for the cases where $\hat{P}_s$ is somewhat distant from α.

For the case in figure 3.2, it would not be of any use to find the upper bound for the probability of making an error larger than 0.35 on a test involving 500 data points if the estimated $\hat{P}_s$ is approximately in the range $[0.3, 0.4]$.

This can be illustrated by another way of representing the UB. In figure 3.3, the UB is represented as a function of N and $dif = \hat{P}_s - P^*$. The use of dif allows an easier visualization of the behavior of the UB as the distance between the estimated error and the error with which we want to compare varies.

Application 1

One application of the bound is to set a good threshold α from which it is not probable that the error of a particular classifier on a experience with N data points will go beyond.

Unfortunately it is not possible to find the analytical expression of $\alpha(UB, \hat{P}_s, N)$. An example follows. Consider that, for the case in figure 3.2 the estimated $\hat{P}_s$ is 0.2. We want to find a value of α such that $P(\widehat{error} \geq \alpha) \leq 0.001$ when using $N = 300$ data points. Using numeric simulation one finds that $\alpha = 0.35$ solves the problem. This way we could say that for this data set and classifier, when using 300 points, it is not probable that the error will be larger than 35%.

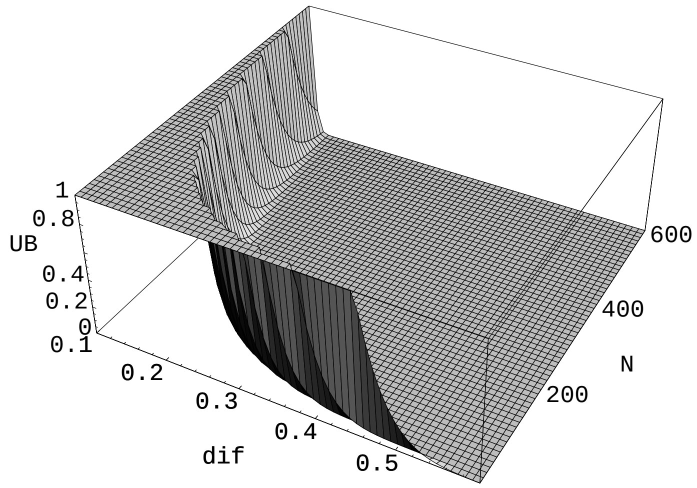

Figure 3.3: The upper bound as a function of N and dif.

Application 2

Another application of the bound is to provide a lower bound for the number of points N necessary to ensure that a classifier will make an error on a given data set smaller than α with a high probability.

This is analogous to the first application, but now we search for N instead of α. Once again we cannot solve the UB explicitly for N, i.e., it is not possible to find analytically $N(\alpha, UB, \hat{P}_s)$.

Example: consider again the case in figure 3.2. Consider also that the estimated $\hat{P}_s = 0.2$. We want to know how many data points N should the test set have such that the probability of the empirical error being larger than 0.35 is at most 0.001. After numerical calculation we get the value of $N = 300$.

3.3.4 Conclusions

We presented bounds based on Sanov's theorem that are for the empirical error and have the form of $P(\widehat{error} \geq \alpha)$ with $\alpha \in (0, 1)$. These are both upper and lower bounds and only involve an estimate of the true error. It is possible to use an estimate since in the formulation of Sanov's theorem, the true error shows up in the bound and not in the event, as it happens with other bounds (Hoeffding's or Chernoff's).

The use of these bounds can be seen as complementary to the other bounds, since the information their use provides is of a different nature.

3.4 Summary

This chapter focused on the evaluation of the classification error. Several methods for counting the error were described and the method that will be used in the thesis, the leave-k-out cross-validation, was presented.

Section 3.2 presented Chernoff's and Hoeffding's bounds for the classification error. These bounds are dependent of the true error, i.e., they provide a measure of how distant an estimate of the error is from the true error. It also presented a lower bound based on Slud's inequality. We introduced new lower and upper bounds, in section 3.3, independent of the true error. These give an estimate of the probability that the error measured is at a certain distance from a given value.

The next chapter presents results on simple rules for combining classifiers, such as the sum, the product and the weighted average. It shows in what conditions the sum and product combination rules are equivalent and develops a model for the added error in a combination using weighted average.

Chapter 4

Sum, product and weighted average

In this chapter, the use of simple rules such as the sum and the product, to combine several classifiers, is analyzed.

In section 4.1 it will be shown that the sum and the product are equivalent rules under some assumptions.

In section 4.2, the use of the sum rule (or simple average) will be confronted with the combination using a weighted average. It will be shown in what extent is the added error changed when using the weighted average instead of the simple average. It will also be shown that, if there is no *a priori* knowledge about the capabilities of the classifiers used in the combination, the simple average should be used instead of the weighted average.

4.1 Combining using sum and product

In this section we present new results on the equivalence of the sum and product combination rules [4].

4.1.1 Introduction

There has been some interest on the comparative performance of the sum and product rules (or the arithmetic and geometric means) [66, 67, 87]. The arithmetic mean is one of the most frequently used combination rules since it is easy to implement and normally produces good results.

In [66], the authors show that for combination rules based on the sum, such as the arithmetic mean, and for the case of classifiers working in different feature spaces, the arithmetic mean is less sensitive to errors than geometric mean.

[87] found experimentally that the combination rules based on the product give better results when all classifiers produce small errors. If at least one of the classifiers makes large errors then the arithmetic mean rule gives better results.

We show that when working in a classification problem with two classes, and using two classifiers that give estimates of the posterior probabilities that sum to one, such as k-NN classifiers, the combination rules arithmetic mean (or the sum) and the geometric mean (or the product) are equivalent, i.e., they have exactly the same error rates. We also show that this is the only case when these rules are equivalent when using this type of classifiers.

4.1.2 Averaging combination rules

We are interested in comparing the performance of two combination rules that involve averaging: the arithmetic and the geometric means. To make the explanation clearer, the figure introducing the combination process, that was presented in section 2.1.3, is reproduced again in this chapter, in figure 4.1.2. These rules are implemented using the following expressions for the function $G(\mathbf{u}_i)$ (defined in section 2.1.3)

$$f_i^{am}(\mathbf{u}_i) = \frac{1}{N}\sum_{j=1}^{N}\mathbf{u}_i(j) \tag{4.1}$$

for the an arithmetic mean and

$$f_i^{gm}(\mathbf{u}_i) = \left(\prod_{j=1}^{N}\mathbf{u}_i(j)\right)^{\frac{1}{N}} \tag{4.2}$$

for the geometric mean. $\mathbf{u}_i(j)$ denotes the j-th component of vector $\mathbf{u}_i$.

We prove that under some conditions these two combination rules are equivalent and then show that those are the only conditions for which the equivalence occurs.

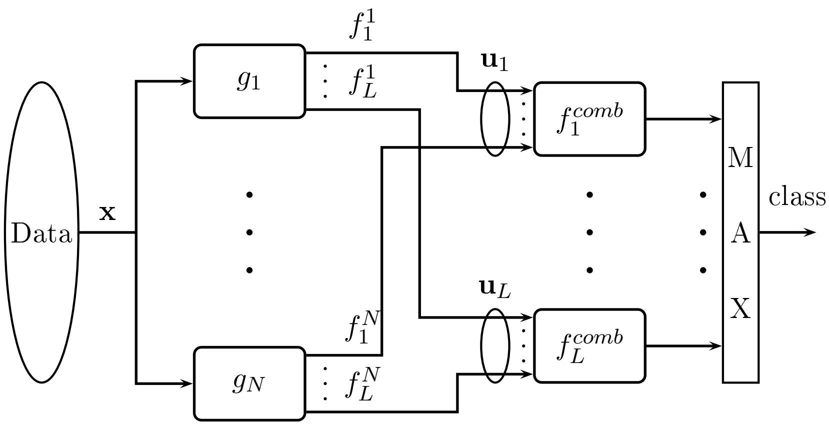

Figure 4.1: The combination process.

4.1.3 Equivalence of sum and product with $L = N = 2$

We state the first result:

Proposition 4.1 *Consider a two class classification problem and two classifiers that approximate the posterior probabilities in a way that for each classifier the sum of its outputs equals one. Then, the classification obtained when combining the posterior probabilities of these classifiers using the sum and the product rule, and choosing the class with the highest posterior probability as the classification result, is exactly the same.*

Proof. Since $L = 2$, there are two vectors of posterior probabilities: $\mathbf{u}_1$ and $\mathbf{u}_2$. And given that $N = 2$, these vectors have two coordinates: $\mathbf{u}_i(j),\ i, j \in \{1, 2\}$. We are considering classifiers whose estimates of the posterior probabilities sum to one, i.e.

$$\mathbf{u}_1(i) + \mathbf{u}_2(i) = 1,\ i = 1, 2. \tag{4.3}$$

There are two cases in which the sum and the product disagree in their predicted class.

Case 1:

$$\mathbf{u}_1(1)\mathbf{u}_1(2) > \mathbf{u}_2(1)\mathbf{u}_2(2)\ \wedge\ \mathbf{u}_1(1) + \mathbf{u}_1(2) < \mathbf{u}_2(1) + \mathbf{u}_2(2) \tag{4.4}$$

Given 4.3 we can rewrite the first part of expression 4.4 as

$$(1 - \mathbf{u}_2(1))(1 - \mathbf{u}_2(2)) > \mathbf{u}_2(1)\mathbf{u}_2(2) \tag{4.5}$$

or

$$\mathbf{u}_2(1) + \mathbf{u}_2(2) < 1 \tag{4.6}$$

The second part of expression 4.4 can also be rewritten using 4.3

$$1 - \mathbf{u}_2(1) + 1 - \mathbf{u}_2(2) < \mathbf{u}_2(1) + \mathbf{u}_2(2) \tag{4.7}$$

or

$$\mathbf{u}_2(1) + \mathbf{u}_2(2) > 1 \tag{4.8}$$

It is impossible that a point $\mathbf{u}_2$ satisfies both 4.6 and 4.8 simultaneously.

The second case when the sum and the product disagree in their predicted class is

Case 2:

$$\mathbf{u}_1(1)\mathbf{u}_1(2) < \mathbf{u}_2(1)\mathbf{u}_2(2) \;\wedge\; \mathbf{u}_1(1) + \mathbf{u}_1(2) > \mathbf{u}_2(1) + \mathbf{u}_2(2) \tag{4.9}$$

As in case 1, a point now would have to satisfy conditions 4.6 and 4.8, which is impossible. This proves that it is not possible to find two points that obey 4.3 and make the sum and product disagree, for a two class problem.

This makes the arithmetic and the geometric mean equivalent when combining two classifiers that obey 4.3 in any two class problem.□

General case

In this section we prove that this equivalence of the combination rules does not generalize for other values of (L, N) other than $(2, 2)$.

Proposition 4.2 *Consider a classification problem with $N \neq 2$ classifiers and $L > 2$ classes. The classification obtained when combining the posterior probabilities of these classifiers using the sum and the product rule, and choosing the class with the highest posterior probability as the classification result, may disagree.*

Proof. In general, equation 4.3 becomes

$$\sum_{i=1}^{L} \mathbf{u}_i(j) = 1, j = 1, \ldots, N. \tag{4.10}$$

Note that for the case $(L = 2, N = 3)$ the following two points make the product and the sum rule disagree:

$$\{\mathbf{u}_1(0; 0.8; 0.8), \mathbf{u}_2(1; 0.2; 0.2)\}. \tag{4.11}$$

This disagreement occurs because the arithmetic mean of $\mathbf{u}_1$ is greater than the arithmetic mean for $\mathbf{u}_2$ making class 1 the chosen for the arithmetic mean combination rule. Simultaneously, the geometric mean of $\mathbf{u}_1$ is smaller than for $\mathbf{u}_2$ making class 2 the chosen one for the geometric mean.

For the case $(L = 3, N = 2)$ the following three points make the combination rules disagree.

$$\{\mathbf{u}_1(0.1; 0.65), \mathbf{u}_2(0.4; 0.3), \mathbf{u}_3(0.5; 0.05)\}. \tag{4.12}$$

We now describe a way to create from these points other points that make those rules disagree for values of (L, N) other than $(2, 2)$.

To create a new set of points that make the combination rules disagree for (L, N) from a set of points that make the combination rules disagree for $(L-1, N)$ it is enough to consider a new point $\mathbf{u}_L$ with all coordinates equal to zero. Example: from the points in 4.11, create a new set of points that make the product and the sum rule disagree for $(L = 3, N = 3)$. The new set of points is then

$$\{\mathbf{u}_1(0; 0.8; 0.8), \mathbf{u}_2(1; 0.2; 0.2), \mathbf{u}_3(0; 0; 0)\}. \tag{4.13}$$

This process guarantees that the new points obey to 4.10 and they do not change the initial decisions of the classifiers since what happens is that the new point will have the smallest sum and the smallest product (both zero) and since the decision is made using the maximum it will not interfere with the initial conflicting decision.

To create a new set of points that make the combination rules disagree for (L, N) from a set of points that make the combination rules disagree for $(L, N-1)$ it is enough to add to each point a new coordinate with value equal to $1/L$.

Example: from the points in 4.11, create a new set of points that make the product and the sum rule disagree for $(L = 2, N = 4)$. The new set of points is then

$$\{\mathbf{u}_1(0; 0.8; 0.8; 0.5), \mathbf{u}_2(1; 0.2; 0.2; 0.5)\}. \tag{4.14}$$

This process guarantees that the new points obey to 4.10 and they do not change the initial decisions of the classifiers since what happens is that both points will have their sum increased with the same amount $(1/L)$ and their product multiplied by the same amount (also $1/L$), thus not changing the initial conflicting decision.

By using these two procedures and the sets of points in 4.11 and 4.12, one can find points that make the combination rules disagree for all values of (L, N) other than $(2, 2)$. (Note that the cases with $N = 1$ are not combinations of classifiers and the cases with $L = 1$ are not classification problems since all points belong to one single class.)□

Proposition 4.3 *Consider a classification problem with N classifiers and L classes. The classification obtained when combining the posterior probabilities of these classifiers using the sum and the product rule, and choosing the class with the highest posterior probability as the classification result, is the same only when $N = 2$ and $L = 2$.*

Proof. This proposition results directly from propositions 4.1 and 4.2.□

4.1.4 Estimates of the posterior probabilities

Depending on the type of classifier used to produce the estimation we face two scenarios: either the outputs of the classifier sum to one thus obeying equation 4.10 (scenario A) or the outputs do not sum to one (scenario B).

Scenario A

Examples of cases that fall into scenario A are those produced by a k-nearest neighbors (k-NN) classifier. It is possible to produce estimates of the posterior probabilities by dividing the number of the k-NN belonging to a given class, by k, as in

$$f_i(\mathbf{x}) = \frac{nc_i(\mathbf{x})}{k}. \tag{4.15}$$

with nc_i being the number of the k-NN of $\mathbf{x}$ belonging to class c_i [18]. This way,

$$\sum_{i=1}^{L} f_i(\mathbf{x}) = \sum_{i=1}^{L} \frac{nc_i(\mathbf{x})}{k} = 1 \tag{4.16}$$

which makes k-NN classifiers' predictions of the posterior probabilities fall into scenario A.

Scenario B

Cases that fall into scenario B are those produced, for instance, by a neural network. In this case there are no guarantees that the outputs of each NN will sum to one.

Another interesting characteristic of these classifiers is the fact that the error may push the estimate of the posterior probabilities out of the $[0, 1]$ interval. This can have a major effect especially on the geometric mean, since the product of an odd number of negative estimates produces a negative estimate which will not be chosen even against the worst positive estimation. This may have a severe impact on the accuracy of the geometric mean combination estimates.

We will conduct experiments illustrating these two scenarios and complete the discussion afterwards.

4.1.5 Experiments

In this section we present several experiments that illustrate the previous results.

Data Sets

We used three data sets from the UCI repository [19]. Details of these data sets are presented in table 4.1. The first column lists the reference we use for a given data set, the second lists their names, the third the number of patterns, the fourth the number of features, the fifth the number of classes in the problem and the last the value of k used in the leave-k-out cross-validation. The value for k was chosen to be approximately 10% of the total number of points in the data sets. This makes the process a 10-fold cross-validation, which is a typical choice for data sets that are not very small (if the data sets are very small the preferred method is the leave-one-out) [18].

Table 4.1: Characteristics of the data sets.

Ref.	Name	#Points	#Features	#Classes	k
DS1	Diabetes	768	8	2	64
DS2	Ionosphere	351	34	2	26
DS3	Sonar	208	60	2	16

Scenario A

In these experiments, k-NN classifiers using euclidean distance are used to produce estimates of the posterior probabilities adopting equation 4.15. These estimates are combined using arithmetic and geometric means. Experiments were made with two, three and four k-NN classifiers. Table 4.2 presents the errors for each data set, and for the combinations: 'AMi' means 'arithmetic mean combination of the first i classifiers' and 'GMi' means 'geometric mean combination of the first i classifiers'.

Table 4.2: Classification errors (in percentage) using the k-NN classifiers.

Classifiers	DS1	DS2	DS3
1NN	32.03	13.43	17.31
3NN	30.60	15.14	18.27
5NN	28.52	15.43	17.31
7NN	27.21	16.29	23.08
AM2	32.03	13.43	17.31
GM2	32.03	13.43	17.31
AM3	32.03	13.43	17.31
GM3	32.03	13.43	17.31
AM4	30.08	14.00	19.23
GM4	32.03	13.43	17.31

Scenario B

We made experiments combining two, three and four MLPs only differing in their topology and in the weight initialization (which is done randomly). The classifiers should approximate posterior probabilities, and these neural networks do so [79], as mentioned in section 2.1.4. The classifiers are trained for 100 epochs using resilient backpropagation [33]. Their topology is presented in table 4.3. Tables 4.4, 4.5 and 4.6 present the average errors for each data set along with the respective average standard deviations.

Table 4.3: Topology of the MLPs.

Classifiers	DS1	DS2	DS3
MLP1	[8 12 2]	[34 20 2]	[60 10 2]
MLP2	[8 14 2]	[34 22 2]	[60 12 2]
MLP3	[8 16 2]	[34 24 2]	[60 14 2]
MLP4	[8 18 2]	[34 26 2]	[60 16 2]

Table 4.4: Average classification errors and standard deviation (in percentage) for $N = 2$.

Classifiers	DS1	DS2	DS3
MLP1	23.96 (2.17)	5.13 (1.72)	14.90 (4.46)
MLP2	26.82 (1.33)	7.69 (2.98)	15.38 (3.29)
AM	23.83 (1.35)	7.12 (2.46)	13.94 (4.21)
GM	24.09 (1.31)	8.26 (3.05)	14.42 (4.15)

Table 4.5: Average classification errors and standard deviation (in percentage) for $N = 3$.

Classifiers	DS1	DS2	DS3
MLP1	28.39 (2.72)	5.41 (2.16)	12.50 (3.02)
MLP2	25.78 (2.82)	4.56 (1.51)	12.98 (1.88)
MLP3	27.21 (6.30)	6.84 (2.81)	15.38 (3.48)
AM	24.61 (1.99)	5.41 (2.16)	9.62 (2.07)
GM	25.52 (2.56)	5.13 (2.22)	11.54 (3.00)

Table 4.6: Average classification errors and standard deviation (in percentage) for $N = 4$.

Classifiers	DS1	DS2	DS3
MLP1	28.39 (2.39)	6.84 (2.56)	13.94 (2.69)
MLP2	27.86 (1.81)	4.84 (1.64)	10.58 (2.95)
MLP3	25.26 (1.62)	7.69 (2.33)	10.58 (2.34)
MLP4	27.73 (1.76)	7.12 (2.36)	11.06 (2.81)
AM	23.57 (1.99)	4.84 (1.79)	7.21 (2.26)
GM	26.95 (2.10)	8.55 (3.19)	11.54 (2.89)

The columns in tables 4.4, 4.5 and 4.6 show the average error and standard deviations, in percentage, for the isolated classifiers and for the combinations, for each data set.

There was not a special care in tuning the individual classifiers, since the important issue was comparative performance.

4.1.6 Discussion

As expected, in scenario A, the combination of two classifiers provides equal results for all data sets, since they are all two class problems. Notice that in the case $N = 3$, the combination rules also present the same error. But, as seen in section 4.1.3, this is not always true. In the combination of 4 classifiers, the geometric mean gives better results than the arithmetic mean for data sets DS2 and DS3.

In scenario B, we see that the combination rules do not have the same performance, not even for the case of $N = 2$. This is due to the fact that these classifiers do not obey expression 4.10, as mentioned before.

The arithmetic mean outperforms the geometric mean, with the exception of the combination of 3 MLPs for DS2. In this case the geometric mean performance was slightly better than the arithmetic mean performance: less than 0.3%, the smallest difference among these two combination rules for all the experiments.

The performance gain of the arithmetic mean relative to the geometric mean is in agreement with results from [66], where it was noted that the product rule is more sensitive to error than the sum rule. Note that an eventual negative output for an estimate of the posterior probabilities can be easily compensated in the arithmetic mean by the other positive values, since we are considering a sum. In the geometric mean, an odd number of negative estimates makes the result negative. The bad performance of the geometric mean is connected with the fact that a possible negative value for the posterior probabilities estimates by only one classifier can make the result, for a given class, smaller (negative) than all the other possible not so good classes but that had no negative output for any of their posterior probabilities estimates. Another justification for the bad performance of the geometric mean is that a given classifier can make the combination result zero or very small by giving a zero or very small estimate for the posterior probabilities even if all other classifiers give high values for the posterior probabilities.

4.2 Combination using simple and weighted averages

In this section we introduce new results on the relation between simple average and weighted averages [2].

4.2.1 Introduction

In [93], the error reduction for the simple average, considering unbiased and independent classifiers was presented. Unbiased classifiers are those in which the expected value of their predictions is the same as the true value. Two classifiers are independent if their errors are not correlated, i.e., if they make 'different 'errors. We extend this result by showing what is the error reduction for the weighted average, also considering unbiased and independent classifiers. We also show that when considering classifiers with these properties and if no information about the relative performance of the classifiers is known, the weights used should be $\frac{1}{N}$ for N classifiers, which means that the simple average should be used.

We present experiments illustrating the theoretical results. We compare the performance of the combination of averaging and weighted averaging of four identical classifiers. In the weighted averaging the coefficients are randomly selected. This is used to illustrate that the simple average is better than any other weight combination, for this type of classifiers.

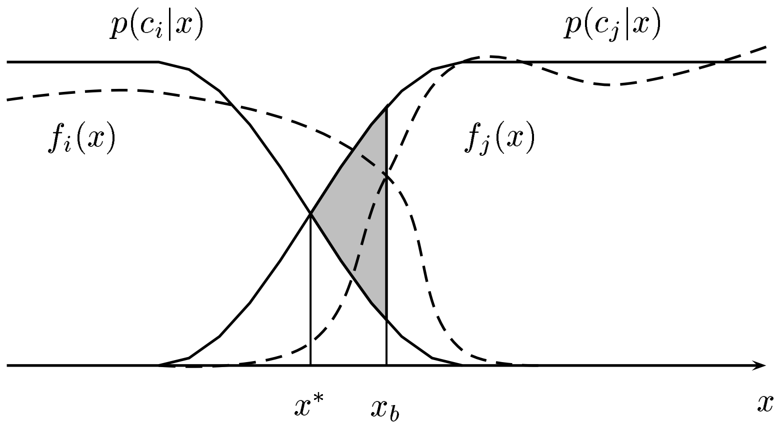

Figure 4.2: The Bayes boundary, the posterior probabilities and its approximations.

4.2.2 Basic concepts

Consider a single classifier whose outputs are expected to approximate the posterior probabilities. The decision boundaries produced by this classifier are expected to be close to the Bayesian decision boundaries. We consider the problem of overlapping classes. We will also consider that the input to the classifier is one-dimensional (without loss of generality [92]).

The total classification error, E_{tot}, is the sum of the Bayes error, E_{bay}, (which is the theoretical limitation to the performance of a classifier or combination of classifiers [15]) with an added error, E_{add}: $E_{tot} = E_{bay} + E_{add}$.

The boundary between classes i and j is the point x^* such that $p(c_i|x^*) = p(c_j|x^*)$. We call x^* the Bayes boundary. The approximation to the posterior probability $p(c_i|x)$ provided by the classifier is $f_i(x) = p(c_i|x) + \epsilon_i(x)$ where $\epsilon_i(x)$ represents the error introduced by the classifier. The boundary produced by the classifier is the point x_b such that $f_i(x_b) = f_j(x_b)$. The difference between the obtained and the real boundaries is $b = x_b - x^*$.

The added error region due to the error of the classifier is

$$A(b) = \int_{x^*}^{x^*+b} (p(c_j|x) - p(c_i|x))dx. \tag{4.17}$$

This is the shaded region in figure 4.2. The expected added error region $E[A(b)]$, which we call the added error E_{add}, is

$$E_{add} = E[A(b)] = \int_{-\infty}^{\infty} A(b) f_b(b) db. \tag{4.18}$$

$f_b(x)$ is the probability density for the variable b. This is unknown.

4.2.3 Average of N classifiers

Tumer in [93] showed that the added error using a simple average of N classifiers, E_{add}^{ave}, is given by

$$E_{add}^{ave} = \frac{E_{add}}{N}. \tag{4.19}$$

This expression is valid for unbiased and independent classifiers. In the following section we show the result for the weighted average, again for unbiased and independent classifiers.

4.2.4 Weighted average of N classifiers

In this derivation we will follow closely [93], although, as we mentioned before, these are new results since they contemplate the weighted average case instead of the simple average. Consider that w_j is a positive value used to weight the contribution of classifier j to the combined posterior probability. The approximations to the posterior probabilities are now given by

$$f_i^{wa}(x) = \sum_{j=1}^{N} w_j f_i^j(x) \tag{4.20}$$

where $f_i^j(x)$ represents the approximation to the posterior probability $p(c_i|x)$ given by classifier j.

We make the following normalization assumption on the weights $\sum_{j=1}^{N} w_j = 1$. The approximation to the posterior probabilities given by the combination of classifiers is

$$f_i^{wa}(x) = \sum_{j=1}^{N} w_j(p(c_i|x) + \epsilon_i^j(x)) \tag{4.21}$$

where the superscript wa stands for 'weighted average'.

Splitting the error term in two terms, the bias β and a random noise η, the error becomes

$$\epsilon_i^j(x) = \beta_i^j + \eta_i^j(x). \tag{4.22}$$

Considering that there is no bias in the classifiers' estimation we have $\beta_i^j = 0$ and

$$f_i^{wa}(x) = \sum_{j=1}^{N} w_j(p(c_i|x) + \eta_i^j(x)) = p(c_i|x) + \eta_i^{wa}(x) \tag{4.23}$$

with

$$\eta_i^{wa}(x) = \sum_{j=1}^{N} w_j \eta_i^j(x). \tag{4.24}$$

The weighted average boundary is called b^{wa}. The E_{add}^{wa} is the expected value of the added error region $A(b^{wa})$, which is

$$A(b^{wa}) = \int_{x^*}^{x^*+b^{wa}} (p(c_j|x) - p(c_i|x))dx. \tag{4.25}$$

This gives E_{add}^{wa} as

$$E_{add}^{wa} = E[A(b^{wa})] = \int_{-\infty}^{\infty} A(b^{wa}) f_{b^{wa}}(b^{wa}) db^{wa}. \tag{4.26}$$

Finding $A(b^{wa})$

To find an expression for $A(b^{wa})$ it will be necessary to make an approximation to the posterior probability around x^*. Using the Taylor expansion,

$$p(c_k|x) = p(c_k|x^*) + (x - x^*)p'(c_k|x^*) + \frac{1}{2}(x - x^*)^2 p''(c_k|x^*) + \dots \tag{4.27}$$

where p' represents the derivative of p in relation to x, $\partial p/\partial x$ and p'' represents $\partial^2 p/\partial x^2$. Ignoring 2nd order and higher terms, we have a linear approximation to $p(c_k|x)$. This way it is possible to calculate $A(b^{wa})$,

$$A(b^{wa}) = \int_{x^*}^{x^*+b^{wa}} (p(c_j|x) - p(c_i|x))dx$$
$$= \int_{x^*}^{x^*+b^{wa}} (x - x^*)(p'(c_j|x^*) - p'(c_i|x^*))dx = \frac{s(b^{wa})^2}{2}, \tag{4.28}$$

with

$$s = p'(c_j|x^*) - p'(c_i|x^*). \tag{4.29}$$

We used the fact that in the boundary, $p(c_j|x^*) = p(c_i|x^*)$, by definition of boundary.

Finding the added error

Using the relation for $A(b^{wa})$, the expression for the added error becomes

$$E_{add}^{wa} = \int_{-\infty}^{\infty} \frac{s(b^{wa})^2}{2} f_{b^{wa}}(b^{wa})db^{wa}. \tag{4.30}$$

Also, we can write the expression for the second moment of b^{wa}

$$M_2(b^{wa}(x)) = E\left[(b^{wa})^2(x)\right] = \int_{-\infty}^{\infty} (b^{wa})^2 f_{b^{wa}} db^{wa}. \tag{4.31}$$

We can use this expression to substitute in (4.30),

$$E_{add}^{wa} = \frac{sM_2(b^{wa})}{2}. \tag{4.32}$$

The second moment can be written as a function of the variance of b^{wa}, $var(b^{wa})$, as

$$M_2(b^{wa}) = var(b^{wa}) + M_1^2(b^{wa}). \tag{4.33}$$

Since the classifiers are unbiased, the first moment of b^{wa} is zero. This means that $M_2(b^{wa}) = var(b^{wa}) = \sigma_{b^{wa}}^2$. $\sigma_{b^{wa}}$ is the standard deviation of b^{wa}.

Finally, we can write the added error as

$$E_{add}^{wa} = \frac{s}{2}\sigma_{b^{wa}}^2. \tag{4.34}$$

We want now to compare this added error with the added error in the case of the single classifier, which is given by [93]

$$E_{add} = \frac{s}{2}\sigma_b^2. \tag{4.35}$$

Relation between E_{add} and E_{add}^{wa}

A relation between $\sigma_{b^{wa}}^2$ and σ_b^2 must be found. Tumer [93] found the following expression for b

$$b = \frac{\epsilon_i - \epsilon_j}{s} \tag{4.36}$$

Since we are considering the case of unbiased classifiers, this can be reduced to

$$b = \frac{\eta_i - \eta_j}{s} \tag{4.37}$$

This gives

$$\sigma_b^2 = \frac{\sigma_{\eta_i}^2 + \sigma_{\eta_j}^2}{s^2} = \frac{2\sigma_{\eta_i}^2}{s^2} \tag{4.38}$$

since the noise is i.i.d.

For the case of b^{wa} we define $x_b = x^* + b^{wa}$ and we have

$$f_i^{wa}(x_b) = f_j^{wa}(x_b) \Leftrightarrow \tag{4.39}$$

$$p(c_i|x^*) + b^{wa}p'(c_i|x^*) + \eta_i^{wa}(x_b) = p(c_j|x^*) + b^{wa}p'(c_j|x^*) + \eta_j^{wa}(x_b) \Leftrightarrow \tag{4.40}$$

$$b^{wa} = \frac{\eta_i^{wa}(x_b) - \eta_j^{wa}(x_b)}{s}, \tag{4.41}$$

again using (4.29).

To find $\sigma_{b^{wa}}^2$ we start by noting that since the noise has zero mean, the second moment of b^{wa} is its variance. To find the second moment, we start by calculating

$$(b^{wa})^2 = \frac{1}{s^2}((\eta_i^{wa})^2 + (\eta_j^{wa})^2 - 2\eta_i^{wa}\eta_j^{wa}). \tag{4.42}$$

Since $E\eta_i^{wa} = 0$ (we are considering noise with zero mean) and the η_k are independent (this independence allows the expected value of the product to be the product of the expected values) we get

$$\sigma_{b^{wa}}^2 = E(b^{wa})^2 = \frac{1}{s^2}(\sigma_{\eta_i^{wa}}^2 + \sigma_{\eta_j^{wa}}^2) \tag{4.43}$$

Now we must rewrite $\sigma^2_{\eta_i^{wa}}$ as a function of σ^2_{η}, which is the same as to say, as a function of $\sigma^2_{\eta_i}$.

Relation between $\sigma^2_{\eta_i^{wa}}$ and $\sigma^2_{\eta_i}$

Consider (4.24). Since η_i^j is noise, its expected value is zero. This implies that the expected value of η_i^{wa} is also zero. This way the variance of η_i^{wa} is equal to its second moment

$$\sigma^2_{\eta_i^{wa}} = M_2(\eta_i^{wa}). \tag{4.44}$$

To calculate the second moment, we start by finding

$$\begin{aligned}(\eta_i^{wa})^2 &= (\sum_{j=1}^{N} w_j\eta_i^j)^2 = (w_1\eta_i^1 + \cdots + w_N\eta_i^N)^2 \\ &= (w_1\eta_i^1)^2 + \cdots + (w_N\eta_i^N)^2 + 2w_1\eta_i^1 w_2\eta_i^2 + \cdots \\ &\quad + 2w_1\eta_i^1 w_N\eta_i^N + \cdots + 2w_{N-1}\eta_i^{N-1} w_N\eta_i^N.\end{aligned} \tag{4.45}$$

Since we are considering i.i.d. noise, the expected value of the product equals the product of the expected values, and since the expected values of the noise are zero we have

$$M_2(\eta_i^{wa}) = E(\eta_i^{wa})^2 = w_1^2 E(\eta_i^1)^2 + \cdots + w_N^2 E(\eta_i^N)^2. \tag{4.46}$$

This leads to

$$\sigma^2_{\eta_i^{wa}} = \sum_{j=1}^{N} w_j^2 \sigma^2_{\eta_i^j}. \tag{4.47}$$

Relation between the added errors

Since the classifiers are independent and the noise is i.i.d., $\sigma^2_{\eta_i^j} = \sigma^2_{\eta_i}$. The added error in the case of a single classifier is given by (4.35), and using (4.38) we find

$$E_{add} = \frac{\sigma^2_{\eta_i}}{s}. \tag{4.48}$$

We can now rewrite (4.34) using (4.43) and (4.47).

$$E_{add}^{wa} = \frac{s}{2}\sigma_{b^{wa}}^2 = \frac{1}{s}\sigma_{\eta_i^{wa}}^2 = \frac{1}{s}\sum_{j=1}^{N} w_j^2\sigma_{\eta_i}^2. \tag{4.49}$$

Finally, using (4.48) we get the relation between the added error when combining using a weighted average and the added error for the single classifier

$$E_{add}^{wa} = E_{add}\sum_{j=1}^{N} w_j^2. \tag{4.50}$$

We can see that in the case of equal weights, $\frac{1}{N}$, we get the result for the simple average (see (4.19)).

4.2.5 What is the best choice for the combination weights ?

Given the previous section results it would be interesting to find out what should the choice for the weights w_j be, such that the smallest added error would result.

This is a constraint problem: the E_{add}^{wa} is proportional to the sum of the squared weights. The minimum of the sum of squared weights must be found, given that they should sum to 1. Let us define

$$F(w_1, ..., w_N) = \sum_{j=1}^{N} w_j^2 \tag{4.51}$$

The constraint is

$$G(w_1, \ldots, w_N) = \sum_{j=1}^{N} w_j = 1 \tag{4.52}$$

We build the function $u = F + \lambda G$. The λ is the Lagrange multiplier. The system of $N + 1$ equations

$$\begin{cases} \frac{\partial u}{\partial w_j} = 0, \quad j = 1, \ldots, N \\ G = 1 \end{cases} \tag{4.53}$$

gives the minimum of F at the point

$$w_j = \frac{1}{N}, \; j = 1, ..., N \tag{4.54}$$

In fact, the method finds an extremum. We know it is the minimum by testing F on other points.

Table 4.7: Average classification errors and standard deviation (in percentage)

Classifier	Diabetes	Ionosphere	Sonar
MLP1	29.30(2.35)	7.69(2.47)	10.58(2.47)
MLP2	27.08(1.78)	6.27(1.68)	13.46(2.40)
MLP3	28.78(1.94)	5.41(2.24)	16.83(3.06)
MLP4	29.04(2.00)	5.70(2.21)	14.42(2.72)
SA	25.39(1.28)	4.56(1.85)	11.06(3.24)
WA	28.52(2.33)	7.12(1.71)	14.42(3.27)

This means that the simple average is the best (in the sense that it minimizes the added error) method for doing a weighted average of several unbiased independent classifiers. For correlated classifiers, consult [49]. The derivation therein is not applicable to our case (unbiased and independent classifiers) since the expressions require the inversion of singular matrices.

4.2.6 Experiments

In this section we present some experiments that confirm our findings. Since the results obtained on the previous sections considered classifiers with i.i.d. noise, we used 4 identical MLPs only differing in the weight initialization (which is done randomly). The other requisite was that the classifiers should approximate the posterior probabilities, and these neural networks do so [79].

We used the data sets listed in table 4.1.

The topologies of the MLPs for each dataset are [8:12:2], [34:20:2] and [60:10:2], respectively. The values for the k in the leave-k-out are also those in table 4.1. The MLPs were trained for 100 epochs using resilient backpropagation.

Table 4.7 presents the average errors and standard deviations, for the experiments.

The columns show the average error and the average standard deviation for the average for the four isolated classifiers and the two combinations: simple average (SA) and weighted average (WA), for the three data sets.

For each data point, the weights for the weighted average combination were generated randomly.

We can see that the best result is given by the simple average, as derived in section 4.2.5.

Note that we did not spend much time in tuning the individual classifiers, since the important issue was comparative performance. Even so, the results are compatible with recent ones [71].

4.3 Summary

This chapter introduced results on the combination of classifiers using the sum, the product and the weighted average.

First we showed that, for a problem with two classes, and when using two classifiers that give posterior probabilities values that sum to one, such as k-NN classifiers, these rules have exactly the same performance. This means that in these cases more than two classifiers should be combined in order to have different performances from these combination rules. We also showed that this is the only case when this equivalence holds. When more than two classifiers are combined the geometric mean showed better performance than the arithmetic mean in our experiments.

We also studied the behavior of these rules when using classifiers that do not give estimates of the posterior probabilities that sum to one, such as MLPs. We concluded, experimentally, that in this case the arithmetic mean outperforms the geometric mean.

The second part of this chapter focused on averaging: simple and weighted. We showed how the added error varies when combining unbiased and independent classifiers using a weighted average. We also showed that, for these classifiers, the weights used for a weighted average should be $\frac{1}{N}$ for a combination of N classifiers. This means that the simple average should be used. We also presented experiments that illustrate the results.

The next chapter focuses on modular neural network architectures. Four architectures are described. Bounds for the classification error are developed in one case and models for the probability of correct classification are introduced for the other three.

Chapter 5

Modular neural network architectures

This chapter focuses on MNNs architectures. The first section introduces the subject, and explains the advantages the MNNs have when compared to non-modular NNs. In section 5.2, the performance of a modular architecture is compared with the performance of a non-modular NN with the same number of weights. Sections 5.3, 5.4 and 5.5 present the probabilistic models for the correct classification of the decoupled, the other-output MNN and the Cooperative MNN (CMNN), respectively. In section 5.6, a unifying view of the decoupled, other-output and CMNN architectures is presented, in which, the models for the probability of correct classification for these architectures are seen as variants of a single expression. Section 5.7 contains experiments where the models' estimates of the errors are confronted with the actual error measurements.

5.1 Introduction

The basic idea behind an MNN architecture is the combination of several small networks (called experts) that are trained to solve a specific part of the full problem. The output of these experts is then analyzed and one of them is selected to produce the final decision of the MNN.

It is known that there is not any unique classification model or architecture that outperforms all others in every problem [57, 63]. It is thus important to know when to use a particular architecture: this can be viewed as a 'model selection' problem. Model selection can be seen from different angles: choosing the optimum number of hidden neurons in a neural network, how much pruning to do on a decision tree or how to define the degree of a polynomial for a fit problem [63, 84].

Classification systems based on modular neural networks (MNN) have been used now for some time. Meanwhile, little theoretical insight has been provided to support their application. Intuitively, an MNN architecture should perform better than a single network for problems that can be separated into several subproblems if each subproblem is to be solved by a single network. This happens since there is a decoupling of the neurons (and weights) used for learning each subproblem when compared to the case of using a single network to solve the entire problem.

Several modular architectures have been proposed [12, 32, 56, 57, 97]. Some of their reported advantages over the single neural network are: they overcome the high internal interference of the single NN that results from the coupling of the hidden-layer weights; they decompose a large size and high complexity task into several simpler subtasks; they are able to deal with complex tasks that introduce overlap between classes.

5.2 Mixture of experts architecture

This section presents the comparison of the performance of one of the basic modular architectures, the 'mixture of experts' architecture, with a single multi-layer perceptron (MLP), with equal number of weights [1]. The idea of comparing architectures with the same number of weights is to insure that the performance differences are due exclusively to the way the weights are connected and not to their number.

Our goal is to show some of the circumstances in which the modular architecture should be used instead of the single network architecture, and thus contribute to solving the 'model selection' problem.

5.2.1 Modular architecture evaluation

The obvious territory for the application of modular neural networks consists of problems which have groups of classes, such that it is possible to train each module of the network to learn a subproblem consisting of a group of classes.

To allow the development of some theoretical results, we restrict the problems we will address to 'separable problems'. This simplifies the following analysis. First, the notion of 'separable problem' is introduced.

Definition 5.1 (Separable problem) *A problem is separable if there is at least one hyper-plane in the feature space that completely separates the classes into two groups.*

Note that a separable problem does not imply that all classes are separable, just that at least one group of classes can be separated from the rest of the classes by a hyper-plane. In each group, the classes can have any degree of overlapping (see the examples in section 5.2.2).

Theoretical error estimation

We state an upper bound for the average generalization error (AGE) of a classification system. This bound is due to Gu and Takahashi [47]. Usual estimates for the generalization error are 'worst case' bounds. By finding an 'average generalization error' the predictions will be less pessimistic and, in this case, are given by a simple expression that involves interesting quantities such as the number of weights of the NN, the training error and the number of training patterns. The price to pay is that the bound obtained is an average bound and thus, there are cases where the estimated error may be smaller than the true error.

The AGE bound of a classification system with α training error ($\alpha \in [0,1]$), with d weights and m training points is given by [47]

$$AGE \leq \alpha + \frac{1}{2}\sqrt{\frac{d}{m}} \tag{5.1}$$

assuming that the loss function has values in $[0,1]$. This is the case in classification, if the loss is one when the pattern is misclassified and is zero otherwise. This is the loss function in expression 2.3.

The modular architecture

We propose the comparison of the AGE of a single neural network to the AGE of a modular NN architecture such as the one in figure 5.1. It is composed of several 'expert' neural networks, $g_i(X)$, $i = 1, \ldots, N$, that only learn a subtask

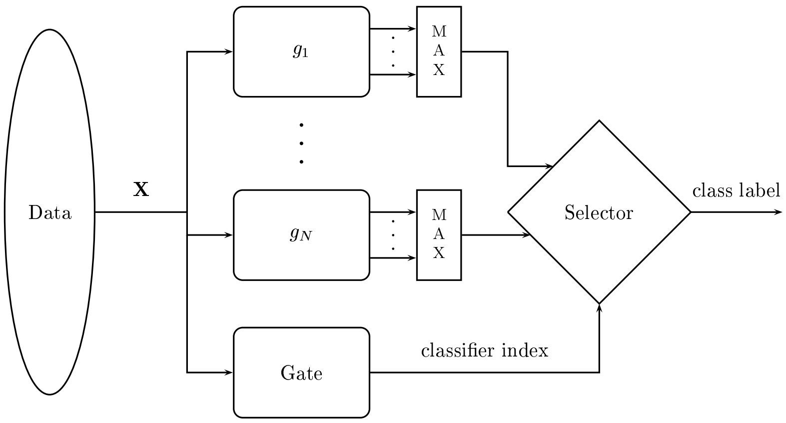

Figure 5.1: The mixture of experts architecture.

corresponding to the classification problem for a subgroup of classes. These expert networks can output a class label or, as represented in the figure, approximations to the posterior probabilities. In this case, the modules labeled ‘MAX’ receive these approximation and output the label of the class with larger posterior probability. The MNN also has a gating network. This network is trained to implement the hyper-plane that separates the groups. It chooses the expert network that makes the final decision. The selector in figure 5.1 represents this selection process and acts like a switch. It allows the output of only one of the experts (the one chosen by the gate) to be the output of the MNN. This figure represents the MNN when set up for testing. To train this MNN, the first step is training the gate. When the gate is able to implement the hyperplane that separates the groups of classes, these are then used to train the individual experts.

To simplify the model we will use in the derivations bellow two assumptions. The first is to consider that the gating network does not make errors. This might seem a strong assumption but the fact that the problem is defined as separable insures that it is not very far from reality. The second assumption is that this gating network does not use any weights. This is a fair assumption since a single perceptron is used for the gate: it implements the required hyper-plane. It uses exactly as many weights as the input features, plus one (the bias). Usually the total number of weights of the architecture is much larger than this, and hence the assumption of considering the number of weights of the gate to be equal to zero.

The single MLP will be called architecture A and the MNN will be called architecture B. We will assign the corresponding index, A or B, to the parameters for each case.

From equation 5.1, we have

$$AGE_A \leq \alpha_A + \frac{1}{2}\sqrt{\frac{d_A}{m_A}} \tag{5.2}$$

and

$$AGE_B \leq \sum_{i=1}^{N} P_i \left(\alpha_i + \frac{1}{2}\sqrt{\frac{d_i}{m_i}} \right) \tag{5.3}$$

where P_i is the probability that the input pattern belongs to a class in group i and N is the number of groups of classes. In fact, P_i is the probability that the gating network sends the pattern to group i, but, since we assume that it makes no errors, they are equal.

The pattern must belong to one of the N groups, hence

$$\sum_{i=1}^{N} P_i = 1 \tag{5.4}$$

There is the same number of training patterns for both cases, so

$$m_A = \sum_{i=1}^{N} m_i. \tag{5.5}$$

By construction of the networks we can ensure that

$$d_A = \sum_{i=1}^{N} d_i. \tag{5.6}$$

This only happens because we made the initial assumption that the gating network uses no weights. This is the same as to say that we are comparing architectures that in principle have the same learning capacity since they use the same number of weights.

The general case

We now investigate the relation between the various parameters that has to be verified such that $AGE_B \leq AGE_A$.

Using expressions 5.2 and 5.3, we have

$$\sum_i P_i \alpha_i \leq \alpha_A - \frac{1}{2}\left(\sum_i P_i \sqrt{\frac{d_i}{m_i}} - \sqrt{\frac{d_A}{m_A}}\right) \tag{5.7}$$

If this expression holds, then $AGE_B \leq AGE_A$ and the modular architecture is better, on average, than the single network architecture.

Using this expression we can build a modular architecture that will use the same number of weights as a single network but will have a better generalization performance. These are the steps to achieve it:

- train a single NN and find its α_A
- calculate the right side of expression 5.7
- train the expert NNs such that their training error verifies expression 5.7

This last step might not always be possible: if this happens then it is possible that architecture A outperforms architecture B.

Note that, given a data set it is possible to determine the values of the P_i, m_A and m_i. Also, by deciding on one particular single NN, the value of d_A is obtained. The values of d_i can be found by dividing the number of weights of the single architecture, d_A, by the several expert networks according to the degree of complexity of their respective subtasks. This can be done 'by hand' for some problems but for others will require the use of an optimization algorithm for searching among the possible distributions of weights between the individual MNNs, one that yields a small classification error.

An approximate bound for AGE_B

Expression 5.3 can be rewritten as

$$AGE_B \leq \sum_{i=1}^{N} P_i \alpha_i + \sum_{i=1}^{N} \frac{P_i}{2} \sqrt{\frac{d_i}{m_i}} \tag{5.8}$$

Using the relation

$$\sum_{i=1}^{N} a_i b_i \leq \sum_{i=1}^{N} a_i \sum_{i=1}^{N} b_i \tag{5.9}$$

for a_i and b_i positive or zero [24], on the second term on the right side of equation 5.8, we obtain

$$\sum_{i=1}^{N} \frac{P_i}{2} \sqrt{\frac{d_i}{m_i}} \leq \sum_{i=1}^{N} \frac{P_i}{2} \sum_{i=1}^{N} \sqrt{\frac{d_i}{m_i}} \tag{5.10}$$

Using the relation

$$\sum_{i=1}^{N} \sqrt{a_i} \leq \sqrt{N \sum_{i=1}^{N} a_i}, \tag{5.11}$$

for a_i positive or zero [24], and replacing the a_i by $\frac{d_i}{m_i}$, the right hand side of expression 5.10 can be upper bounded

$$\sum_{i=1}^{N} \frac{P_i}{2} \sum_{i=1}^{N} \sqrt{\frac{d_i}{m_i}} \leq \sum_{i=1}^{N} \frac{P_i}{2} \sqrt{N \sum_{i=1}^{N} \frac{d_i}{m_i}} \tag{5.12}$$

Finally, combining expressions 5.4, 5.8 and 5.12, an approximation to the bound AGE_B is found

$$\widetilde{AGE}_B \leq \sum_{i=1}^{N} P_i \alpha_i + \frac{1}{2} \sqrt{N \sum_{i=1}^{N} \frac{d_i}{m_i}} \tag{5.13}$$

where the tilde means 'approximate'.

This bound for the performance of the modular architecture is less tight than the one in 5.3, but will allow some insight into the behavior of the modular architecture for problems with different number of groups.

Given expressions 5.2 and 5.13, if $\widetilde{AGE}_B \leq AGE_A$ then

$$N \leq \frac{4 \left(\alpha_A + \frac{1}{2} \sqrt{m_A / d_A} - \sum_{i=1}^{N} P_i \alpha_i \right)^2}{\sum_{i=1}^{N} \frac{d_i}{m_i}} \tag{5.14}$$

Since $\sum_{i=1}^{N} P_i \alpha_i \geq 0$, when $\sum_{i=1}^{N} \frac{d_i}{m_i} > 1$, it is possible to write

$$N \leq 4 \left(\alpha_A + \frac{1}{2} \sqrt{m_A / d_A} \right)^2 \tag{5.15}$$

Since equation 5.15 presents an upper bound for N, we conclude that for a fixed number of training samples, m_A, and for a fixed number of weights, d_A, to be used in the architecture, there is a number of groups N^* such that, for $N > N^*$ it does not pay off to use this type of modular architecture. This is somewhat unexpected because it could be thought that as the number of groups increases the more likely it would be that the modular architecture would outperform an equally powerful single NN. An explanation for this behavior is proposed in section 5.2.3.

Notice that equation 5.15 is always true for $N = 1$, since the number of weights d_A is smaller than the number of training patterns m_A. This should happen because $N = 1$ is the case of a non-modular architecture.

5.2.2 Experiments

In this section we use two artificial problems and one real problem to illustrate the application of the previous results.

Experiment 5.1

Artificial data was generated for a 2 feature, 5 class problem. Class 1 has a uniform distribution and all other have normal distributions. 1000 points for each class were generated such that the data set is linearly separable into two groups of 2 and 3 classes respectively. The data set is represented in figure 5.2 (only 100 points per class are represented to facilitate the visualization). The gate's decision boundary for one of the repetitions of the experiment is plotted in figure 5.3.

Architecture A consisted on a single MLP with topology [2:18:5], having thus 149 weights. Architecture B consisted of two MLPs with with topologies [2:12:2] and [2:14:3], that were used respectively for group 1 (classes 1 and 2) and group 2 (classes 3, 4 and 5). The total number of weights is the same as in architecture A. All networks were trained using resilient back-propagation for 100 epochs.

The leave-k-out cross-validation method, with $k = 500$ was used to measure the error rates. This means that there were 10 repetitions, since the data set had 5000 points.

Table 5.1 shows the mean and average standard deviation for the experiment.

The results in table 5.1 show that architecture B has smaller generalization error (test set error) than architecture A (8.10% versus 10.30%) in this problem.

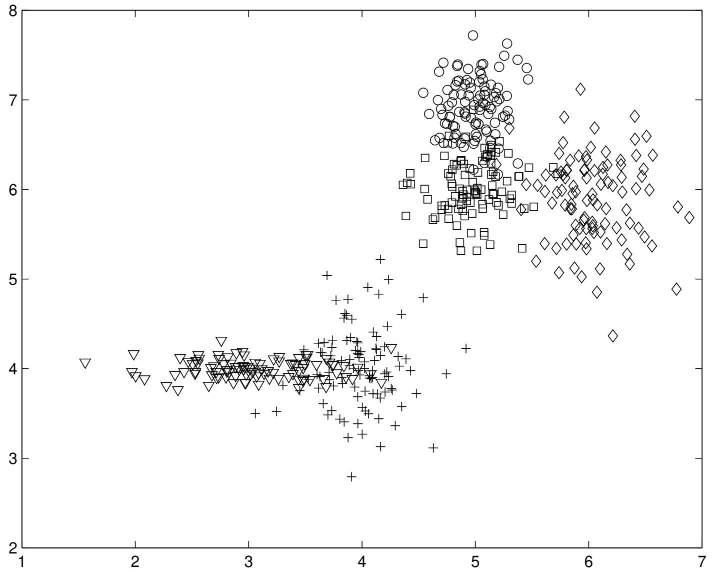

Figure 5.2: Data set for experiment 5.1.

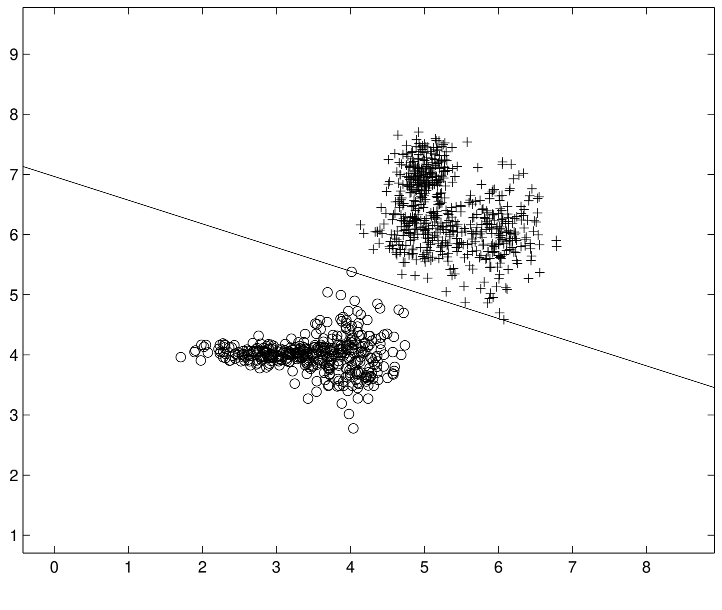

Figure 5.3: The gate's decision boundary for one of the repetitions of experiment 5.1.

Table 5.1: Generalization errors and AGE bounds with standard deviation (in percentage) for experiment 5.1.

Classifiers	*AGE*	Gen. error
Arch. A	19.36 (1.50)	10.30 (1.66)
Arch. B	17.03 (0.25)	8.10 (0.43)

It can also be observed in table 5.1 that the bound AGE_B is smaller than AGE_A and thus predicts correctly that architecture B outperforms architecture A in this problem. This supports the conclusion that these bounds can be used to select the best architecture before the test phase.

Experiment 5.2

Artificial data was generated for a 3 feature, 10 class problem such that it is separable into 3 groups of classes. 1000 points for each class were generated. The data set is represented in figure 5.4 (only 60 points per class to facilitate the visualization). The data set can be separated into three groups of 2, 3 and 5 classes, by two planes, as shown in figure 5.5. Architecture A consists on an MLP with topology [3:24:10], which uses 346 weights. Architecture B consists of 3 MLPs with topologies [3:13:2] (80 weights), [3:15:3] (108 weights) and [3:17:5] (158 weights). The total number of weights is the same as in architecture A. The first network was used in group 1 (2 classes), the second in group 2 (3 classes) and the third in group 3 (5 classes). All networks were trained using resilient back-propagation for 100 epochs.

The experiment was a leave-k-out cross-validation, with $k = 1000$. This means that there were 10 repetitions, since the data set had 10000 points.

Table 5.2: Generalization errors and AGE bounds with standard deviation (in percentage) for experiment 5.2.

Classifiers	*AGE*	Gen. error
Arch. A	23.14 (0.25)	13.16 (0.39)
Arch. B	21.81 (0.35)	11.94 (0.48)

From the results in table 5.2 we find that architecture B has smaller generaliza-

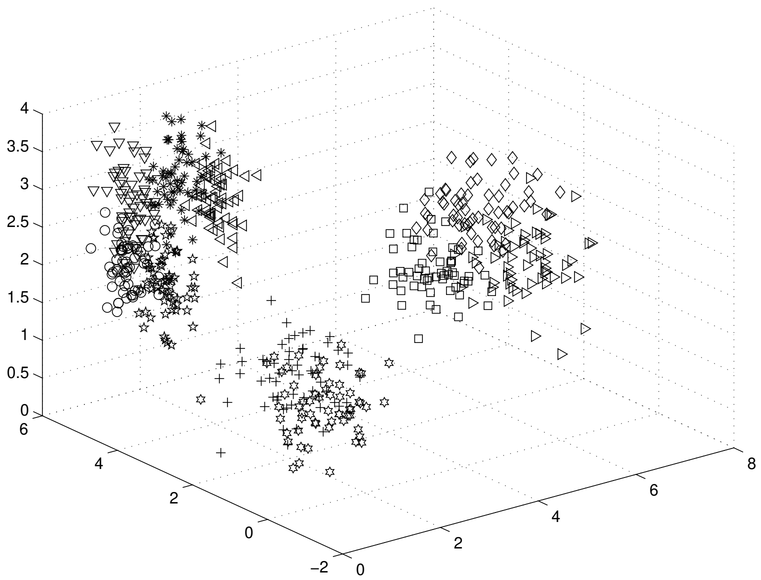

Figure 5.4: Data set for experiment 5.2.

tion error than architecture A (11.94% versus 13.16%). The prediction obtained by looking at the bounds is in agreement with this result since $AGE_B < AGE_A$.

Experiment 5.3

To test the prediction capabilities of our bounds on real data we used a data set for a vowel discrimination problem. The data consists of the first and second formants of the vowels 'i','I','a' and 'A' produced by 76 speakers (33 males, 28 females and 15 children). Each vowel was repeated twice by each speaker, giving a total number of 608 data points. It is a subset of the Peterson and Barney data set referred in [56] and is represented in figure 5.6. Both features were linearly transformed by dividing by 1000. This problem is clearly separable and the gate's decision boundary for one of the repetitions of the experiment is in figure 5.7.

Architecture A consists of an MLP with topology [2:10:4], which uses 74 weights. Architecture B consists of 2 MLPs with topologies [2:7:2] and [2:7:2]. The total number of weights is the same as in architecture A. All networks were trained using resilient back-propagation for 100 epochs.

The experiment was a leave-k-out cross-validation, with $k = 76$. This means that there were 8 repetitions.

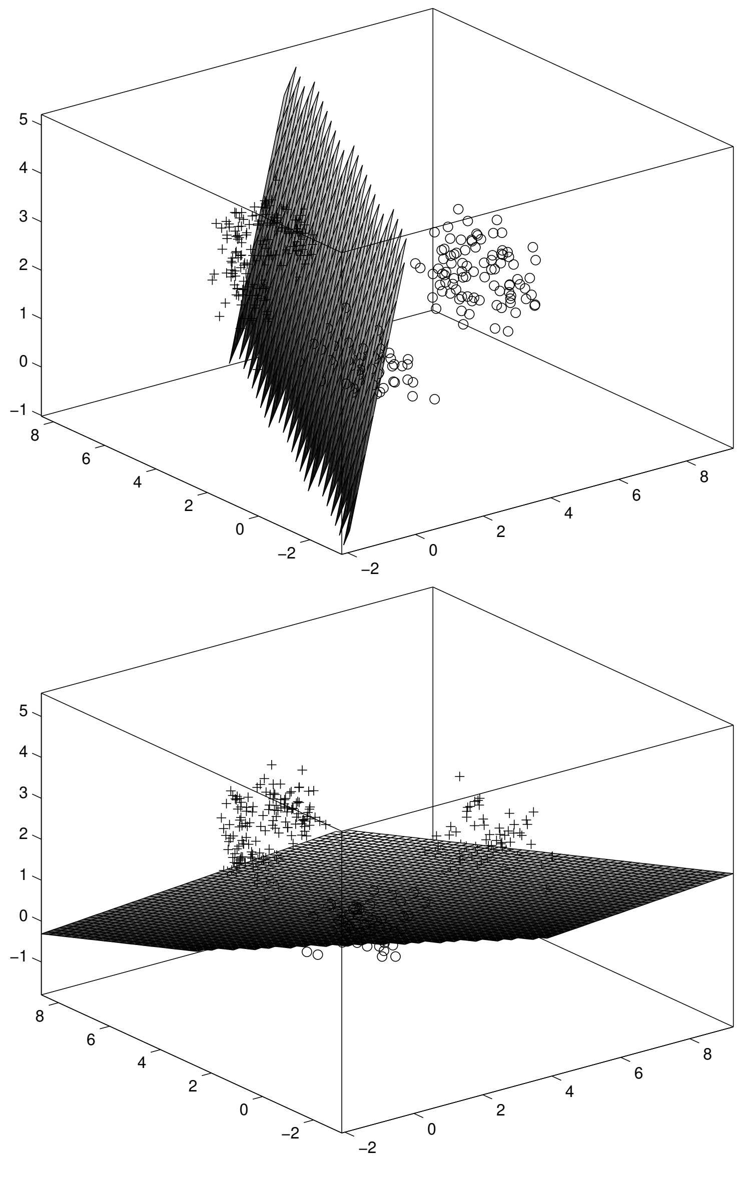

Figure 5.5: Separation of groups by two planes in experiment 5.2.

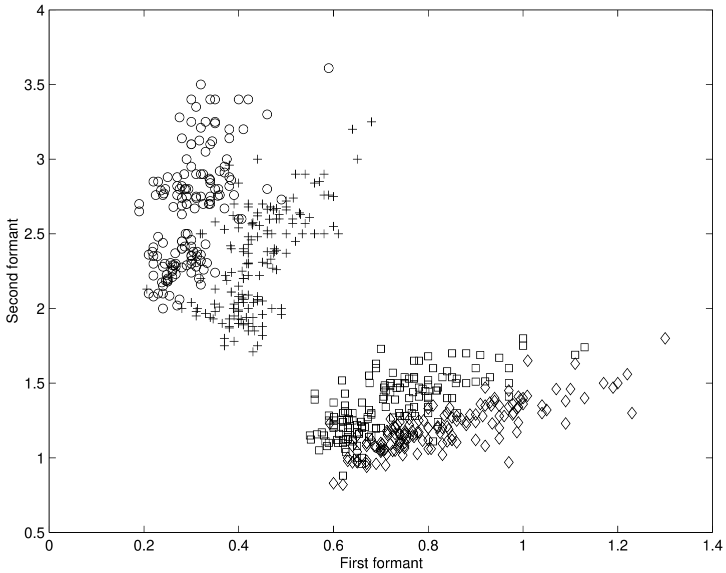

Figure 5.6: Data set for experiment 5.3.

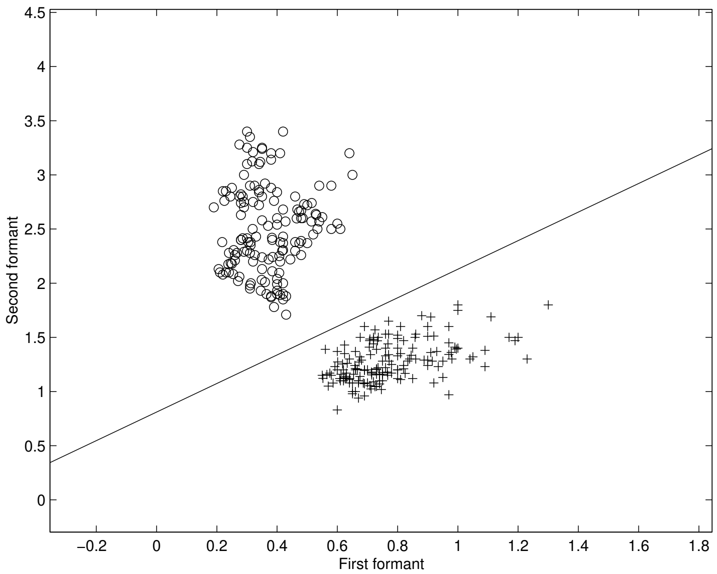

Figure 5.7: The gate's decision boundary for one of the repetitions of experiment 5.3.

Table 5.3: Generalization errors and AGE bounds with standard deviation (in percentage) for experiment 5.3

Classifiers	*AGE*	Gen. error
Arch. A	22.63 (1.03)	10.03 (1.94)
Arch. B	20.54 (0.53)	9.05 (0.88)

From the results in table 5.3 we find that architecture B has smaller generalization error than architecture A (average 9.05% versus 10.03%). The prediction obtained by looking at the AGE bounds is in agreement with this result since $AGE_B < AGE_A$. Once again this supports the idea that the best architecture for this problem could have been chosen after the training phase, by inspection of the AGE bounds.

5.2.3 Discussion

This section presented support for the use of modular neural network architectures for separable problems.

We gave a bound for the average generalization error of the 'mixture of experts' modular architecture that showed a good prediction capability, as illustrated in the experiments.

We showed that sometimes there is a number of groups of classes such that for problems with more groups it does not compensate to use the modular architecture. We think this is very interesting as it is somewhat unexpected. We suppose that this happens because if there are many groups of classes the number of weights available for each expert network may not be enough for learning its subtask.

We showed how a modular NN architecture can be build to outperform, on average, a single NN with the same number of weights and using the same number of training patterns, using expression 5.7.

By using the same number of weights in both architectures, the differences in performance are only due to the way the weights are connected in the architectures. Our results support the idea that the modular architecture, given its decoupled weights, performs better than the single network, which uses the same weights for learning the full problem.

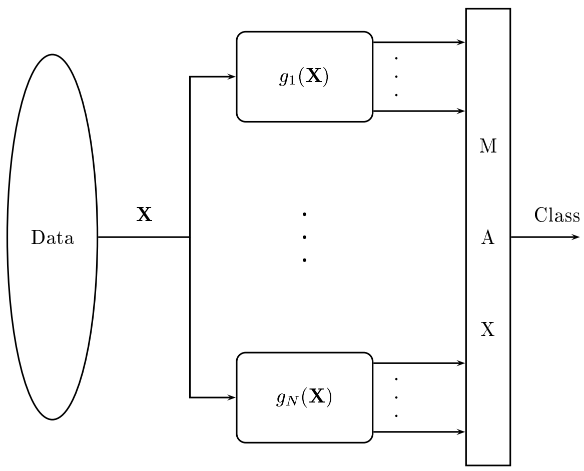

Figure 5.8: The decoupled MNN architecture.

5.3 Decoupled MNN

In this section we propose a theoretical model for the probability of correct classification (PCC) for the 'decoupled' MNN. This architecture is very simple and consists of N expert NNs trained to solve a particular subproblem of the total problem. They all try to classify the input pattern and the final output of the architecture is the class with highest posterior probability. The decoupled MNN is represented in figure 5.8.

The total number of classes in the problem is L. C_n represents class n. Each expert NN has a set of outputs I_i corresponding to the classes of its subproblem. We have thus,

$$\sum_{i=1}^{N} \#I_i = L \tag{5.16}$$

A classifier g_i outputs a vector of estimates of the posterior probabilities, $p_i(X \in C_n|X)$,

$$g_i(X) = (p_i(X \in C_n|X), \ldots, p_i(X \in C_{n-1+\#I_i}|X)), n, \ldots, n-1+\#I_i \in I_i \tag{5.17}$$

with I_i being the set of indexes that correspond to the classes that classifier g_i can deal with and $\#I_i$ the number of corresponding classes.

An input pattern is classified by all expert NN. The final decision consists of the class with the largest posterior probability from among all the expert NNs. This selection of the class with largest posterior probability is represented in figure 5.8 by the 'MAX' box.

5.3.1 General case

We extend the operator 'max' to work with vectors: it outputs the largest component of the vector.

The set of points in which the event 'classifier g_i has the largest posterior probability of all classifiers'occurs, will be called A_i:

$$A_i = \{X : \max g_i(X) = \max_{j=1,\ldots,N} \{\max g_j(X)\}\} \tag{5.18}$$

The set of points in which the event 'class n has the largest posterior probability for classifier g_i ' occurs will be represented as $B_{n,i}$:

$$B_{n,i} = \{X : p_i(X \in C_n|X) = \max g_i(X)\} \tag{5.19}$$

The set of points in which the event 'classifier g_i makes a correct classification' occurs will be represented by D_i:

$$D_i = \bigcup_{n \in I_i} (B_{n,i} \cap \{X : X \in C_n\}) \tag{5.20}$$

We will make a language abuse and call A_i an event and not the set of point where this event takes place. This will also be done for the similar defined sets $B_{n,i}$ and D_i and others to be defined below.

The probability of correct classification, PCC, for the decoupled architecture is given by

$$PCC_{dec} = P\left(\bigcup_{i=1}^{N} \bigcup_{n \in I_i} (A_i \cap B_{n,i} \cap \{X : X \in C_n\})\right) \tag{5.21}$$

or, using expression 5.20,

$$PCC_{dec} = P\left(\bigcup_{i=1}^{N}(A_i \cap D_i)\right) \tag{5.22}$$

where the index '*dec*' is used to identify the decoupled architecture. Since the events D_i are disjoint, so is the intersection $(A_i \cap D_i)$, and expression 5.22 can be written as

$$PCC_{dec} = \sum_{i=1}^{N} P(A_i \cap D_i) \tag{5.23}$$

In order to simplify this expression so that it can be evaluated experimentally, the assumption in the next paragraph is introduced.

5.3.2 Assumption

If we assume that the event of classifier g_i having the highest posterior probability among all classifiers is independent from the fact that it has correctly classified the input pattern, then the exact PCC_{dec} will be approximated by $\widetilde{PCC}_{dec}$

$$\widetilde{PCC}_{dec} = \sum_{i=1}^{N}(P(A_i)P(D_i)) \tag{5.24}$$

The exact probability of correct classification for this architecture is in expression 5.23, but expression 5.24 is the approximation we will use.

5.4 Other-output MNN

In this section a theoretical model for the probability of correct classification (PCC) for the 'other-output' MNN is proposed. This architecture is an evolution of the decoupled architecture. The individual experts now have an extra output that is used to report that the pattern is in a region of the input space in which they are not experts.

The other-output architecture consists of N expert NNs, $g_i(X),\ i = 1, \ldots, N$, trained individually to solve a particular subproblem of the total problem, and also to recognize when the input data does not belong to its own subproblem. This architecture is represented in figure 5.9.

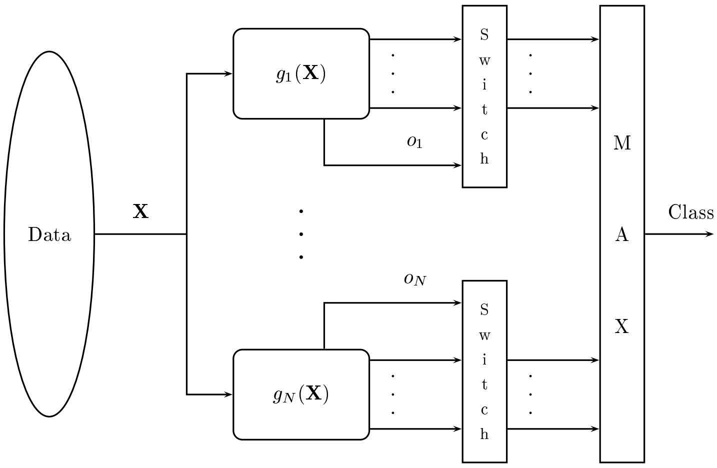

Figure 5.9: The other-output MNN architecture.

The total number of classes in the problem is L. Each expert NN has a set of outputs I_i corresponding to the classes of its subproblem and thus equation 5.16 holds.

Each expert g_i has also an output, o_i, that is set to zero if the expert 'believes' that the input belongs to its own set of classes, and is set to one, if it believes that the input should be classified by other expert.

Each expert g_i tries to classify the input pattern. If o_i is one, the switch associated with g_i will not allow the estimates of the posterior probabilities produced by classifier g_i to reach the maximum function. The final decision consists on the class with the largest posterior probability from among all the experts that have the other-output at a zero level.

5.4.1 General case

We define the set containing the indexes of all the experts as

$$H = \{1, \ldots, N\} \tag{5.25}$$

and also

$$H_j = H \backslash \{j\}, \; j \in H \tag{5.26}$$

The set containing the labels of all classifiers that had a maximum of their posterior probabilities greater than the maximum of classifier g_i, is defined by

$$J_i = \{j \in H_i : \max g_j(X) > \max g_i(X)\} \tag{5.27}$$

The set of points where the event 'the other-output of expert g_i is at level zero' occurs is represented by O_i.

Using the definitions in expressions 5.18 and 5.19, the probability of correct classification for the other-output architecture can be written as

$$PCC_{oo} = P\left(\bigcup_{i=1}^{N} \bigcup_{n \in I_i} ((A_i \cup \{X : o_j = 1, \forall j \in J_i\}) \cap B_{n,i} \cap \{X : X \in C_n\} \cap O_i)\right) \tag{5.28}$$

where the index 'oo' identifies the other-output architecture.

Notice how the event A_i in the previous architecture is replaced by the event $(A_i \cup \{X : o_j = 1, \forall j \in J_i\}) \cap O_i$. This means that even if classifier g_i does not have the maximum posterior probability from among all the classifiers, it can still be the one that classifies the input if all the other-outputs of the classifiers that have larger posterior probabilities, are at level 1.

Defining the event

$$E_i = \begin{cases} \{X : o_j = 1, \forall j \in J_i\} & \text{if } J_i \neq \emptyset \\ 0 & \text{if } J_i = \emptyset \end{cases} \tag{5.29}$$

and using expression 5.20, we can write

$$PCC_{oo} = P\left(\bigcup_{i=1}^{N} ((A_i \cup E_i) \cap O_i \cap D_i)\right) \tag{5.30}$$

Since the D_i are disjoint, we have

$$PCC_{oo} = \sum_{i=1}^{N} P((A_i \cup E_i) \cap O_i \cap D_i) \tag{5.31}$$

or

$$PCC_{oo} = \sum_{i=1}^{N} P((A_i \cap O_i \cap D_i) \cup (E_i \cap O_i \cap D_i)) \tag{5.32}$$

Since the events A_i and E_i are disjoint, PCC_{oo} can be written as

$$PCC_{oo} = \sum_{i=1}^{N} (P(A_i \cap O_i \cap D_i) + P(E_i \cap O_i \cap D_i)) \tag{5.33}$$

In the following paragraph we introduce some assumptions in order to simplify the expression of the PCC_{oo}.

5.4.2 Assumptions

We assume that the event A_i is independent of the event D_i, as we did when we derived expression 5.24. We can further assume that the event O_i is independent of D_i: the fact is that each classifier is learning two *separate* tasks simultaneously - it learns whether a pattern belongs to its group of classes and it learns to approximate the posterior probabilities of the classes in its group.

According to these assumptions, the PCC_{oo} can be approximated by

$$\widetilde{PCC}_{oo} = \sum_{i=1}^{N} (P(A_i)P(O_i)P(D_i) + P(E_i \cap O_i \cap D_i)) \tag{5.34}$$

We assume that the event O_i is also independent of E_i, which is plausible since there is no direct relation between these events. Finally, if we assume that E_i is independent of D_i which again is very likely, as the fact that classifier g_i has made a correct classification is not directly connected to the fact that all classifiers with posterior probabilities larger than the maximum posterior probability of classifier g_i have their other-outputs with a value of one. The approximation to the PCC for the other-output architecture comes as

$$\widetilde{PCC}_{oo} = \sum_{i=1}^{N} ((P(A_i) + P(E_i))P(O_i)P(D_i)) \tag{5.35}$$

5.5 CMNN Architecture

This section describes the CMNN architecture [13] and presents a model for its PCC. This architecture is represented in figure 5.10. The $CMNN$ consists on

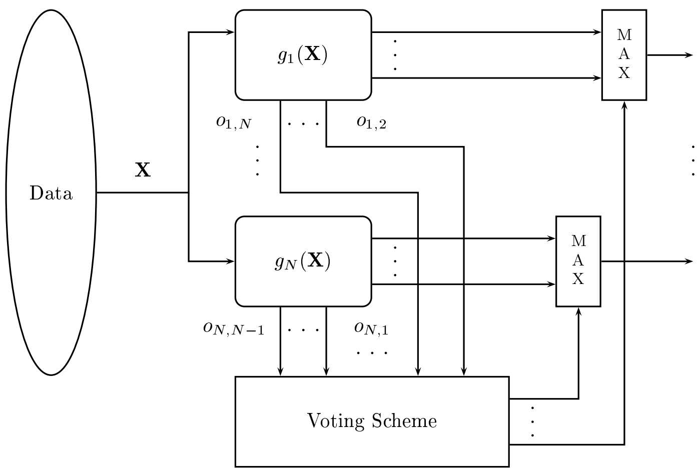

Figure 5.10: The CMNN architecture.

N expert NNs, $g_i(X),\ i = 1, \ldots, N$, that are trained to recognize a particular subproblem of the total problem, and also to recognize when the input data does not belong to its own subproblem, just like in the other-output architecture (see section 5.4). The difference is that, instead of the output indicating only whether the pattern belongs to another classifier, it outputs explicitly to which classifier it belongs.

The total number of classes in the problem is L. Each expert g_i has a set of outputs I_i corresponding to the classes of its subproblem. So, once again, equation 5.16 holds.

Each expert g_i has also a set of $N - 1$ outputs, $o_{i,j},\ j \in H_i$, corresponding to the other experts in the architecture. These outputs have values in $[0, 1]$. A higher value represents more confidence in the fact that the classifier g_j should be selected to produce the final decision.

For each input pattern, each expert NN produces a vector of posterior probabilities and tries to guess which classifier should be used to classify this pattern.

The final decision consists on the class with the largest posterior probability from the classifier that is selected by the voting of the $o_{i,j}$ outputs of all classifiers.

5.5.1 General case

The set of points where the event 'classifier g_i is elected as the one which will output the final decision' occurs will be represented by F_i.

This way the probability of correct classification for this architecture comes as

$$PCC_{cmnn} = P\left(\bigcup_{i=1}^{N} \bigcup_{n \in I_i} (B_{n,i} \cap \{X : X \in C_n\} \cap F_i)\right) \tag{5.36}$$

where the index 'cmnn' is used to identify the CMNN architecture.

Using expression 5.20,

$$PCC_{cmnn} = P\left(\bigcup_{i=1}^{N} (D_i \cap F_i)\right) \tag{5.37}$$

Since the events D_i are disjoint, so is the intersection $(D_i \cap F_i)$, and expression 5.37 can be written as

$$PCC_{cmnn} = \sum_{i=1}^{N} P(D_i \cap F_i) \tag{5.38}$$

Until now, the voting strategy used to select the expert that will give the final output, has been ignored due to F_i. Among the several different voting strategies that can be used (of which, several are referred in chapter 2), we will study the four strategies that were proposed originally with the CMMN architecture in [13].

5.5.2 Different voting strategies

Plurality vote

In this case, each expert g_i votes only for one (other) expert: the one with the highest value of $o_{i,j}$. The expert with more votes wins.

The number of votes that classifier g_i receives is T_i:

$$T_i = \sum_{j \in H_i} \mathcal{I}_{\{\max_{n \in H_j} o_{j,n} = o_{j,i}\}} \tag{5.39}$$

where $\mathcal{I}_{\{A\}}$ denotes the indicator function, which gives one if the event A is true and zero otherwise.

Using this definition we can write

$$F_i = \{X : T_i = \max_{j \in H} T_j\} \tag{5.40}$$

Borda count

The $o_{j,i}$ are ranked and a value of $N-2$ is assigned to the largest output of classifier g_j, $N-3$ to the second largest and so on, such that the smallest output receives a value of zero.

The values are summed for each classifier and the one with the largest sum is elected.

We define the function $r(o_{j,i}) : H \times H \mapsto \{1, \ldots, N-1\}$ that gives the rank of $o_{j,i}$.

The total value assigned to classifier g_i is

$$BC_i = \sum_{j \in H_i} (N - 1 - r(o_{j,n})) \tag{5.41}$$

The event F_i is thus

$$F_i = \{X : BC_i = \max_{j \in H} BC_j\} \tag{5.42}$$

Fuzzy vote

In this case, the elected classifier is the one with the largest summation of the values of the votes $o_{j,i}$.

We define

$$S_i = \sum_{j \in H_i} o_{j,i} \tag{5.43}$$

In this case, the event F_i comes as

$$F_i = \{X : S_i = \max_{j \in H} S_j\} \tag{5.44}$$

Nash vote

Nash vote is similar to fuzzy vote but instead of having a sum of the $o_{j,i}$ the product is used.

We define

$$PD_i = \prod_{j \in H_i} o_{j,i} \tag{5.45}$$

In this case, we have

$$F_i = \{X : PD_i = \max_{j \in H} PD_j\} \tag{5.46}$$

5.5.3 Assumptions

To simplify the expression 5.38 it will be assumed that the events D_i and F_i are independent. This leads to the following approximation to the PCC_{cmnn}

$$\widetilde{PCC}_{cmnn} = \sum_{i=1}^{N} P(D_i)P(F_i) \tag{5.47}$$

This assumption can be justified since the fact that classifier g_i is the chosen one for classifying the input (event F_i) is dependent of the majority of the classifiers, thus not particularly dependent of classifier g_i (the dependence that may exist, since classifier g_i also votes, is decreased as the total number of experts increases). Since the event D_i depends exclusively of classifier g_i, it is not a strong assumption to consider it independent of F_i.

5.6 A unifying view

The probability of correct classification for the decoupled, other-output and CMNN models can be viewed as variations of a single expression

$$PCC_j = \sum_{i=1}^{N} P(D_i \cap G_i), \; j \in \{dec, oo, cmnn\} \tag{5.48}$$

with G_i being given by the expressions in table 5.4. The event G_i for the CMNN is dependent on the particular voting scheme adopted, as was seen in section 5.5.2, and is represented in table 5.5.

Expression 5.48 evidentiates the basic nature of these MNNs: there has to be a part of the network that learns the subproblem, hence the event D_i. Another part of the network is concerned with deciding which of the experts should be used on

Table 5.4: The event G_i according to the architecture.

Architecture	G_i
Decoupled	A_i
Other-output	$(A_i \cup E_i) \cap O_i$
CMNN	F_i

Table 5.5: The event F_i according to the voting scheme.

Voting Scheme	F_i
Plurality	$\{X : T_i = \max_{j \in H} T_j\}$
Borda count	$\{X : BC_i = \max_{j \in H} BC_j\}$
Fuzzy	$\{X : S_i = \max_{j \in H} S_j\}$
Nash	$\{X : PD_i = \max_{j \in H} Pd_j\}$

the particular area of the input space from which the pattern to be classified is coming from, hence the event G_i.

This clearly synthesizes how these architectures work and shows that what distinguishes them is basically how they decide which expert should be used to produce the final decision.

5.7 Experiments

In this section we present experiment 5.4 (artificial data) and experiment 5.5 (real data) that are used to validate the models developed in the previous sections and to assess the performance of the decoupled, the other-output and CMNN architectures.

5.7.1 Weights used in the architectures

We intend to use the same number of weights for each architecture involved in the same experiment. This will evidenciate which of these architectures shows a better performance due to their intrinsic capabilities without the distortion that would be present if some had different learning capabilities (expressed by a different number of weights).

For the experiment 5.4, we found that an MLP with about 1000 weights is capable of properly learning the problem (increasing the number of weights only brought marginal improvements in performance, if any). For the experiment 5.5, this number was around 150 weights.

To insure that all the architectures have the same number of weights, the following constraints had to be respected:

- for experiment 5.4, the decoupled MNN has 2 MLPs with topology $[2:a:4]$ and 3 with topology $[2:b:3]$; the other-output has 2 MLPs with topology $[2:c:5]$ and 3 with topology $[2:d:4]$; the CMNN has 2 MLPs with topology $[2:e:8]$ and 3 with topology $[2:f:7]$;
- for experiment 5.5, the decoupled MNN has 2 MLPs with topology $[2:g:2]$; the other-output and the CMNN have 2 MLPs with topology $[2:h:3]$.

The variables $a, b \ldots, h$ are the unknown number of hidden layer neurons.

The values used, that approximately obey these constraints, are in tables 5.6 and 5.9 (the only exceptions are: the other-output in experiment 5.4 has 1132 instead of 1137 as the other architectures; the MLP in experiment 5.5 has 172 instead of the 174 weights used in the other architectures).

5.7.2 The models for the PCC of the MNNs

This section list the models derived in the previous sections that are going to be used in the estimation of the performance of the MNNs.

For the decoupled architecture

$$\widetilde{PCC}_{dec} = \sum_{i=1}^{N} (P(A_i \cap D_i)) \tag{5.49}$$

For the other-output architecture

$$\widetilde{PCC}_{oo} = \sum_{i=1}^{N} (P(D_i)(P(A_i \cap O_i) + P(E_i)P(O_i))) \tag{5.50}$$

For the CMNN architecture

$$\widetilde{PCC}_{cmnn} = \sum_{i=1}^{N} P(D_i)P(F_i) \tag{5.51}$$

5.7.3 A note on training the decoupled architecture

Originally [10], the training of the decoupled architecture consisted on making the individual expert's output approximate the posterior probabilities for the classes belonging to their respective groups. This implied that the experts were not exposed to any patterns that did not belong to their group, during the training phase.

It was the author's opinion, even before doing any tests using this architecture, that it did not make sense to ask for the experts' opinions when the input belonged to a group other than their own, since for those patterns the expert had no training and would output only noise. The problem was that, this noise could have a value higher than the true class posterior probability and, hence, be the chosen class.

After running several tests with this architecture, the author confirmed the expected problem. This architecture would perform consistently bad: usual error rates higher than 90%. The solution found for this problem consists on training the experts, not only to approximate the posterior probabilities of the patterns belonging to their own group, but also to train them to output zero when confronted with patterns from groups other than their own. With this change, the decoupled architecture considerably improved its performance (to values typically under 50% of error).

It would not be correct to train them to output the true classes for patterns belonging to groups other than their own, since this would not make them experts, in the sense that they are specialized on a particular region of the problem, but they would be learning the complete problem, and the architecture would be an ensemble (a set of equal NNs) competing for the output - clearly not what the original idea was.

Note that this change in the way the experts are trained, from the original specification, does not imply any change in the model of the PCC developed earlier.

5.7.4 Experiment 5.4

A 17 class artificial problem was generated with 2 features and 17 classes that are roughly clustered in 5 groups. The classes were generated using Gaussian distributions. 150 points were generate for each class. This gives for the total data

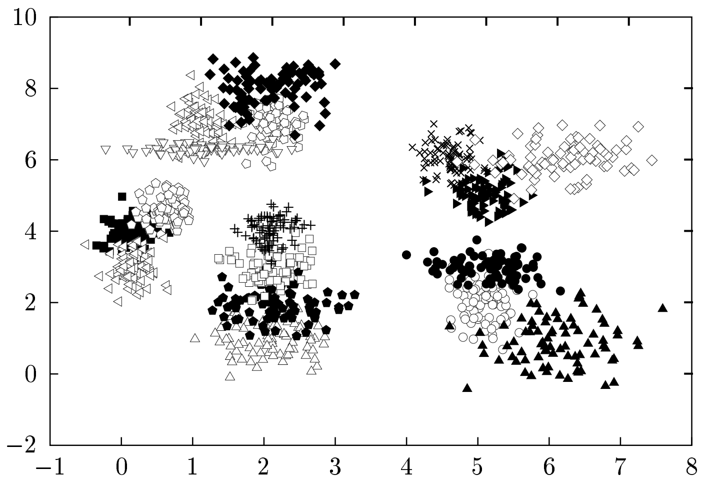

Figure 5.11: The data set for the experiment 5.4.

Table 5.6: Topologies and total number of weights for the experiment 5.4.

Architecture	Topology	# Weights
MLP	[2:56:17]	1137
Decoupled	$2 \times [2:35:4] + 3 \times [2:35:3]$	1137
Other-output	$2 \times [2:30:5] + 3 \times [2:30:4]$	1132
CMNN	$2 \times [2:20:8] + 3 \times [2:22:7]$	1137
CMNN-iv	$2 \times [2:20:8] + 3 \times [2:22:7]$	1137

set, 2550 points. The data is plotted in figure 5.11 (only 80 point per class are included in to avoid cluttering the figure).

This data set is used because it has 5 groups of classes, and is thus a good problem to illustrate the MNN architectures.

The values of the topologies of the architectures used in this experiment are in table 5.6.

Table 5.7 presents the estimated probabilities of several events used in the models. Each line in the table corresponds to one of the experts in each of the architectures. These probabilities were estimated as follows:

Table 5.7: Parameters estimated on experiment 5.4.

Decoupled		Other-output				CMNN	
$P(A_i)$	$P(D_i)$	$P(D_i)$	$P(E_i)$	$P(O_i)$	$P(A_i \cap O_i)$	$P(D_i)$	$P(F_i)$
0.2298	0.8342	0.8212	0.0117	0.2376	0.2267	0.8063	0.2353
0.1671	0.8945	0.8862	0.0014	0.1812	0.1769	0.8458	0.1757
0.2051	0.8990	0.9004	0.0015	0.1776	0.1769	0.8938	0.1776
0.2239	0.9201	0.9230	0.0029	0.1788	0.1741	0.9226	0.1773
0.1741	0.8671	0.8492	0.0030	0.2349	0.2329	0.8246	0.2341

- $P(A_i)$: the number of times classifier i had the largest posterior probability is counted and divided by the number of test points;
- $P(D_i)$: the number of times the expert i was chosen correctly divided by the number of times that it should have been chosen;
- $P(E_i)$: the number of times that all classifiers with maximum of the posterior probabilities larger than classifier i have their other-output at level 1 divided by the number of times that at least one of the other expert had the maximum of its posterior probabilities larger than the maximum of the posterior probabilities of classifier i;
- $P(F_i)$: the number of times the expert i is chosen divided by the number of test points;
- $P(O_i)$: the number of times that expert i had its other-output at level 0 divided by the number of test points;
- $P(A_i \cap O_i)$: the number of times classifier i had the largest posterior probability and simultaneously had its other-output at level 0 is counted and divided by the number of test points.

The CMNN used was implemented with plurality voting. An idea of the influence that the voting strategy has in the final error can be obtained by comparing these results with the CMNN-iv, which was implemented with ideal voting: no errors in the expert selection are made.

The first and last groups of classes in the problem were the ones with 4 classes. The other three groups had 3 classes. Hence, the experts that work with groups

1 and 4, should be chosen more often to give the final decision, since the classes have all the same number of patterns. This is reflected in the fact that, for the CMNN, the values of $P(F_1)$ and $P(F_5)$, are higher than the values of $P(F_i)$ for $i \in \{2, 3, 4\}$.

This is also evident for the other-output architecture. In this case, the probabilities $P(A_i \cap O_i)$ and $P(O_i)$, for $i \in \{1, 5\}$, are again higher than the same probabilities for the cases in which $i \in \{2, 3, 4\}$.

It is interesting to notice that the decoupled architecture does not follow this pattern. The values of $P(A_i)$ in this case, are not higher for the first and fifth group as with the previous architectures. This shows clearly that the expert in the decoupled architecture that made the final decision was not always the correct one.

Since the values corresponding to the portion of the model dedicated to choosing which expert should produce the final decision are similar for the other-output and the CMNN, we can conclude that the other-output should be used since its simpler method for choosing the final decision maker produces the same results as the more elaborate (which implies the need of more weights) process used in the CMNN.

The values of $P(D_i)$ in table 5.7, for the different architectures, also convey some interesting information. The values of $P(D_i)$ for the decoupled architecture are higher than for the other-output architecture, which in turn are higher than the values of $P(D_i)$ for CMNN architecture (with only a couple of exceptions). This can be understood since the architectures all have the same number of weights. This means that since in the decoupled architecture there is no mechanism for learning which of the experts should produce the final decision, all the weights are used to learn the problem. In the CMNN the mechanism is more elaborated than in the other-output architecture. This means that the other-output will have more weights available to learn the problem than the CMNN architecture.

Table 5.8 presents the average classification error and standard deviation, both in percentage, for the 10 repetitions of the leave-k-out cross-validation, with $k = 255$.

The decoupled MNN performed considerably worst than the other classifiers. This is due to the fact that an expert can make the final decision on a point that does not belong to its region of expertise, by simply outputting a value of

Table 5.8: Measured and estimated errors, with standard deviation, for the experiment 5.4.

Architecture	Measured Error [%]	Estimated Error [%]
MLP	16.59 (0.92)	-
Decoupled	54.47 (3.67)	11.87 (0.92)
Other-output	13.18 (0.70)	13.50 (0.80)
CMNN	15.26 (0.72)	14.70 (0.75)
CMNN-iv	14.71 (0.75)	-

posterior probability greater than all the others. This means that an expert making a mistake, and outputting a large value for one posterior probability can impose its decision on an expert with the correct answer.

The other-output architecture outperformed the others. This fact suggest that, since all architectures have the same number of weights, the organization of the weights in the other-output is better than in the single MLP, where all weights are used to learn the problem simultaneously. But, the amount of weights used in the CMNN architecture for deciding on which classifier to use for the final decision, made the number of weights available for each expert to learn its own subproblem, not enough to outperform the other-output.

According to what was previously mentioned, there is an entry in this table for an CMNN-iv architecture. This represents the error if the CMNN had an ideal voting (iv) system that always knew which was the correct expert. This is used to assess the error introduced by the voting system considered. It can be observed that the voting in this experiment introduced about 0.5% error. It is also interesting to note that, even with this ideal voting system, the CMNN would not outperform the other-output architecture in this experiment.

A final comment on the performance of the single MLP: the difference in performance for the CMNN architecture is not large, only 1.33% in 16.59% (8% difference). This may suggest that the problem is not complex enough to take advantage of the elaborate CMNN architecture. But it pays to go as far as the complexity of the other-output architecture since in this case the difference in error is 3.41% in a total of 16.59% (21% difference).

We will now comment the performance of the models as far as the estimates of the error they produced.

The estimate for the decoupled architecture is quite different from the actual result. This means that the assumption we made to obtain expression 5.24 is not valid. It was, in fact, quite a strong assumption. It meant that the choice of which expert to use was independent of whether the expert had made the correct decision. It turns out that it is not independent and it is understandable: if an expert makes a correct decision it will most probably output a large value for the respective posterior probability, thus making it a strong candidate to be the one that makes the final decision for the architecture.

The estimates for the other-output and the CMNN architectures are both quite accurate: 0.32% on 13.18% (difference of 2.4%) for the other-output and 0.56% on 15.26% (3.7% difference) for the CMNN. These results validate the approximate models and the assumptions made to obtain them.

5.7.5 Experiment 5.5

The second data set to be used in the experiments of this chapter is the vowel discrimination problem presented in section 5.2.2. The data is plotted in figure 5.6. It is used because it presents data from a real problem and it has clearly two groups. The fact that it has only 2 input features allows a better visualization of the complexity of the problem. To this respect, it can be observed that this problem is simpler than the previous one, not only because it has fewer classes (and groups) but also because these classes exhibit a smaller level of overlapping. This is confirmed by the error rates measured, as we will see.

The values of the topologies of the architectures used in this experiment are in table 5.9. The errors were measured using leave-k-out cross validation with $k = 76$, hence, the error was measured 8 times.

The estimated probabilities of the events used in the models are presented in table 5.10. Each line in the table corresponds to one of the experts in the architecture.

For this experiment, the modular architectures only had 2 experts. The probability of each of these 2 experts to be chosen for making the final decision should be 50% for both since the problem had the same number of patterns in each group. It can be observed in table 5.10 that the events relative to this choice of the expert

Table 5.9: Topologies and total number of weights for the experiment 5.5.

Architecture	Topology	# Weights
MLP	[2:24:4]	172
Decoupled	$2 \times [2:17:2]$	174
Other-output	$2 \times [2:14:3]$	174
CMNN	$2 \times [2:14:3]$	174
CMNN-iv	$2 \times [2:14:3]$	174

Table 5.10: Parameters estimated on experiment 5.5.

Decoupled		Other-output				CMNN	
$P(A_i)$	$P(D_i)$	$P(D_i)$	$P(E_i)$	$P(O_i)$	$P(A_i \cap O_i)$	$P(D_i)$	$P(F_i)$
0.4490	0.9391	0.9415	0	0.5000	0.5000	0.9215	0.5000
0.5510	0.9226	0.9058	0	0.5000	0.5000	0.9208	0.5000

that produces the final decision, i.e., $P(A_i \cap O_i)$ and $P(O_i)$ for the other-output and $P(F_i)$ for the CMNN, reflect this equal probability. As it happened in experiment 5.4, the decoupled architecture does not exhibit this equal probability in the event $P(A_i)$.

The values of the $P(D_i)$ for all 3 architectures are quite similar. This makes the difference in performance between the other-output and the CMNN very subtle, since the choice of the expert for the final decision also had very similar results.

The errors and its estimates for all the classifiers are in table 5.11. The other-output and CMNN had approximately the same performance. The surprise came from the MLP: the best performance, by a difference of more than 1.5% in about 7% (which gives about 20% relative difference). We believe that, since this problem is simple, it does not benefit from the added complexity of the modular architectures. The MLP had all its weights learning the full problem and benefited against the MNN which had part of their weights used to learn how to separate the groups.

The estimates for the other-output and the CMNN gave exact results, once again validating the approximate models proposed.

The estimate for the decoupled is not valid since it is based on a assumption that can not be made, as seen previously.

Table 5.11: Measured and estimated errors, with standard deviation, for the experiment 5.5.

Architecture	Measured Error [%]	Estimated Error [%]
MLP	6.09 (1.43)	-
Decoupled	12.83 (6.16)	6.98 (1.30)
Other-output	7.57 (1.47)	7.57 (1.47)
CMNN	7.73 (1.30)	7.73 (1.30)
CMNN-iv	7.73 (1.30)	-

5.7.6 Conclusions

The decoupled architecture has a poor performance due to the way it is designed. We expected that the CMNN architecture would outperform the other-output architecture. The results in these experiments make us think differently. The better performance of the CMNN (reported in [11, 12]) must happen with more complex problems. What we observed was that, for architectures with equal weights, the other-output performed equally or better than the CMNN. Also, for simple problems, it might not be worth the use of a modular architecture at all, since the MLP may give better results.

Given these results, we intend to use in the next chapter, the applications to medical data, the other-output and the CMNN. We will still use an MLP as a standard method of comparison.

5.8 Summary

This chapter focused on the study of MNNs architectures.

First it showed that the simple mixture of experts MNN outperforms an MLP with the same number of weights, on a separable problem. The derivation also showed that, in some cases, there is an upper bound on the number of groups of classes of a problem such that the mixture of experts outperform the MLP. We presented a justification for this fact.

Models for the probability of correct classification for the decoupled, other-output and CMNN architectures were proposed. These models were used to do an analysis of the performance of these MNNs. This enabled a greater insight to how these MNNs work. Experiments were presented validating the models for the MNNs. A unifying view of the decoupled, other-output and CMNN architectures was also presented.

The next chapter contains experiments involving medical data. Results from the rest of the thesis are applied in these experiments. An example of the development of a PR system, as described in the first chapter, is also included.

Chapter 6

Application to medical data

This chapter presents applications of the ideas and methods presented in the previous chapters, to medical data.

In sections 6.1 and 6.3, the problem of detecting the presence of fluid in the lungs from the analysis of chest radiographs is studied. This is a complete example of a pattern recognition problem since all the steps represented in figure 1.1 are present.

Section 6.2 describes the pre-processing applied to the data sets in this chapter.

Section 6.4 introduces the algorithm used to cluster the data sets for training the modular architectures that are used on the subsequent problems.

Section 6.5 presents a problem involving the analysis of a cardiotocographic (CTG) signal to determine the state of a fetus.

In section 6.6 the data set represents the problem of detecting 4 types of heart disease.

The final data set used in this chapter concerns a breast cancer detection problem. It is in section 6.7.

Section 6.8 presents the conclusions drawn from the experiments and the final section contains a summary of the chapter.

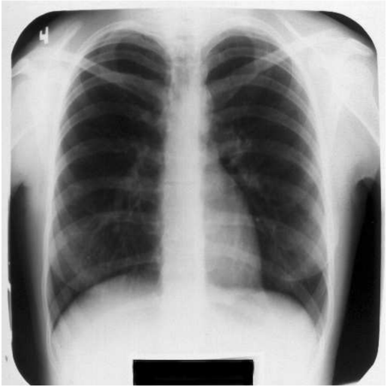

Figure 6.1: Normal radiograph showing both CDAs clear.

6.1 Chest radiograph image analysis

6.1.1 Introduction

To facilitate the task of analyzing chest radiographs, a Computer-Aided Diagnosis (CAD) system is under development at INEB. Since there is a variety of different pathologies that can be detected in a chest radiograph [26, 68, 73, 74, 102, 103], a divide-and-conquer approach is being used to tackle the problem. This means that an appropriate method is being developed to detect the presence of each pathology in the image.

This section presents one of the methods developed [5, 6]. The goal is to detect whether the costal diaphragmatic angles (CDAs) are occupied.

This occupation can indicate an active or residual pathology. The most frequent in an active pathology is the pleural spill (presence of fluid between the pleura). This situation happens most frequently in the inflammatory and infectious processes, tuberculosis, cancer and cardiac insufficiency. As a residual process following these pathologies there is usually a tack between the pleura originating an occupancy of the costal diaphragmatic angles [72, 73].

This can be detected since in a normal lung the costal diaphragmatic angle ends in a sharp corner, otherwise it does not present such a corner. Figure 6.1 presents a normal radiograph where both CDAs present sharp corners, while figure 6.2 shows an image where the right CDA is occupied.

There are several difficulties in the analysis of these images. If the patient

Figure 6.2: Occupied right lung CDA.

does not inhale properly, or the patient presents a large rib cage, the CDAs may not exhibit a sharp corner. Another problem occurs when the patient presents a large amount of fat tissue. An example of such a case is presented in figure 6.3. Breasts in women and large heart silhouettes, together with the presence of air in the stomach and in the colon, also interfere with the proper detection of the diaphragm (see figures 6.4 and 6.5).

6.1.2 Description

The images were digitized from a 10×10 cm chest radiograph film to $2048 \times 2048 \times 12$ bit with a UMAX 1220S scanner.

The image analysis task is divided into several steps:

- lateral lung edge delineation;
- diaphragm line edge representation;
- computing CDAs co-ordinates;
- diaphragm slope characterization, near the CDAs.

The features obtained from these steps were used as input to a MLP to detect whether the CDAs were occupied.

Figure 6.3: Both CDAs are clear but difficult to detect due to a large amount of fat tissue.

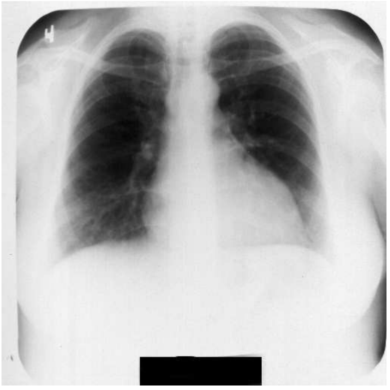

Figure 6.4: Left CDA difficult to detect due to large heart silhouette and the superposition of the breast.

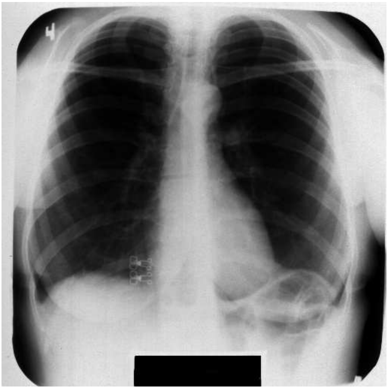

Figure 6.5: Left CDA difficult to detect due to the presence of air in the colon and stomach and the superposition of the breast.

Pre-processing

In order to reduce noise in the images and also to reduce the amount of information to process, without jeopardizing the final goal, a reduction of the number of bits per pixel, from 12 to 8, and a resize of the images, to 512×512, was done. The histogram was stretched so that the full 8 bit range was used by the images. The function used was

$$IS = 255\frac{(I - min(I))}{(max(I) - min(I))} \tag{6.1}$$

where I is the original image and IS is the stretched image. The max and min functions return the maximum and minimum gray value of the image, respectively. Figures 6.6 and 6.7 show an image before and after the histogram stretch. A different type of normalization, based on the average levels of the lungs and mediastinum [90] was tested, but was not considered adequate to this problem since the segmentation results obtained using the images with this normalization were not satisfactory.

Lung edge detection

We use a gradient plus thresholding approach to detect the external edge of the lungs [58]. Although it can produce well-defined objects, this technique can be difficult to automate. This happens because the result is critically dependent on filter size, and thresholding levels.

The detection of the exterior lung edges was done separately for the right and left lungs. We will describe the process for the right lung. The left lung exterior edge was obtained by the same process - differences will be referred when appropriate.

The first step was the definition of the region of interest (ROI). A ROI is defined so that the attention is focused on a particular region of the image. The vertical signature is calculated by

$$VS(x) = \frac{1}{512}\sum_{k=1}^{512} IS(k,x) \tag{6.2}$$

Figure 6.8 shows a typical vertical signature. MX is the maximum value of VS within the 1D ROI defined by $x \in 512 \times [0.4, 0.6]$. This point approxi-

Figure 6.6: Normal radiograph before the histogram stretch.

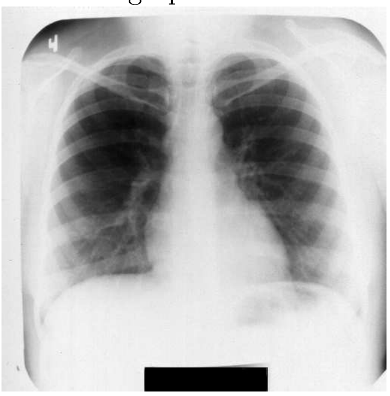

Figure 6.7: Radiograph of figure 6.6 after the histogram stretch.

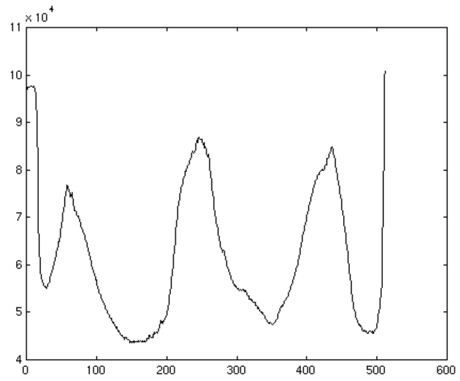

Figure 6.8: Vertical signature for a typical image.

mately coincides with the mediastinum. The right ROI is defined as the sub-image $IR(x, y) = IS(x, y)$, $x \in [20, MX - 20]$, $y \in [1, 512]$. The left ROI is defined as $IL(x, y) = IS(x, y)$, $x \in [MX + 20, 492]$, $y \in [1, 512]$. Note that a margin of 20 pixels is given at both ends of the ROIs. This avoids some noise present in the right and left portions of the image. It also shrinks the area to process when we try to find the CDAs, since they are not found within a ± 20 pixels zone from MX.

Since the external edge of the lungs is approximately a vertical line, we applied a horizontal gradient filter to enhance vertical transitions. The result for one of the images is in figure 6.9a.

The resulting image was thresholded at 127 (figure 6.9). This binarized image was again differentiated only in the horizontal direction, with a small window filter (figure 6.10). This was again thresholded now at a level of 100.

Among the various lines we find the external edge of the lung. To isolate it we label all the objects in the image and build a new image containing only the largest object - this is the desired external lung edge. Finally, the edge was thinned. The result is presented in figure 6.10b. We are interested in finding the CDAs, consequently the only portion of the lung edges that we seek is the one that intercepts the diaphragm.

Diaphragm detection

We start by defining a ROI. It is defined as the sub-image $Ir(x, y) = IS(x, y)$, $x \in [1, 512]$, $y \in [205, 512]$. These limits were obtained empirically. With this ROI

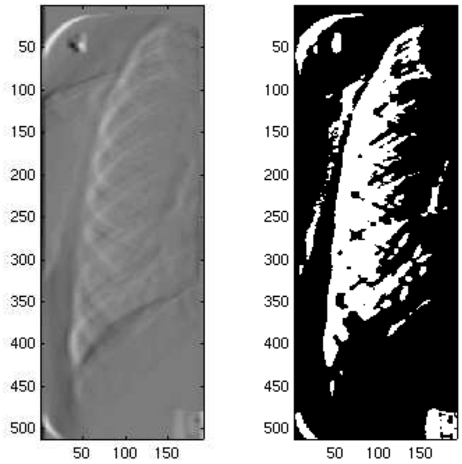

Figure 6.9: First part of the process of finding the lateral lung edges. Left: gradient of the ROI. Right: thresholding at 127.

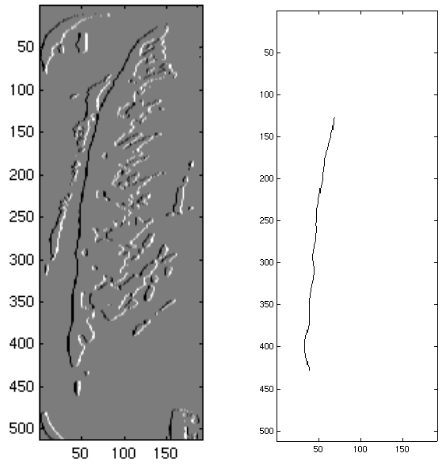

Figure 6.10: Second part of the process of finding the lateral lung edges. Left: horizontal gradient. Right: edge after thresholding, selection and skeletonizing.

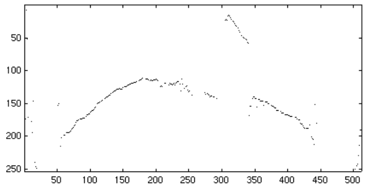

Figure 6.11: Diaphragm detected.

the top portion (40%) of the image is excluded from the processing, since we know the diaphragm is in the lower portion of the image.

Since the diaphragm is mainly a horizontal structure, the first operation is an approximation to a derivative in the vertical direction. Actually, we did a convolution with a filter that detects vertical transitions from dark to bright regions. The resulting image was analyzed leaving only the maximum value of the derivative for each column [45]. This can be justified, since the diaphragm appears as the main transition from a dark to bright region that is oriented vertically. Problems could occur since the ribs can also fit in this description. To avoid them, the filter was constructed so that it only detects objects larger than the average rib. Figure 6.11 shows an example of a segmented diaphragm.

Note that this procedure does not obtain the diaphragm correctly in the zone below the mediastinum. The reason is that, in this zone, the diaphragm is not a transition between a dark and a bright region. This is not important to us since we are only interested in the diaphragm near the CDAs. So we simply ignore the central part of the detected diaphragm.

Determining the occupancy of the CDAs

The intersection of the diaphragm with the lateral lung edges define the CDAs. Figure 6.12 shows this intersection for one image.

To decide if an CDA is occupied, we chose to analyze the region around the vertex of the CDAs and to compare the shape of the left and right portions of the diaphragm. In a normal image the portion of the diaphragm near the CDAs is almost symmetric. If one of the CDAs is occupied, that symmetry is lost. Also, the coordinates of the CDAs are important features since in a normal case, the

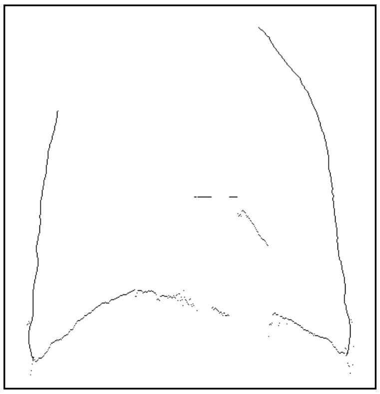

Figure 6.12: Result of the segmentation of the CDAs.

vertex of the CDAs are approximately at the same height (eventually, in a normal patient, the right CDA may be in a lower position than the left).

Thus, one of the features is obtained from the coordinates of the vertexes of the CDAs. Since the important coordinate is the height, the X coordinate is not used. Since the important characteristic is the relative position of the vertices's, only the difference between the heights of the vertexes is used.

To retain information about the shape of the diaphragm near the CDAs, the slope of the diaphragm in 5 equally distant points for each CDA is calculated. The values of the slopes for the right and left CDAs, in a normal radiograph, are almost symmetric. In the case of an occupied CDA, this symmetry is lost. To calculate these slopes, the difference in height between each two consecutive points was divided by their spacing. Note that the points are all equally spaced.

It would be possible to use as features only the differences between each pair of corresponding right/left slopes. Since a symmetry would yield values close to zero, an asymmetry would yield high values. But this would fail to detect the cases where both CDA are occupied: in this case there would also be a symmetry between the CDAs. What distinguishes these cases from a normal one is the shape of the CDAs. So, the features used are the two sets of 5 values of the slopes of the diaphragm near the CDAs and the difference between the heights of the vertexes of the CDAs. This means that the total number of features is $2 \times 5 + 1 = 11$.

6.1.3 Experiments

A database of 174 images was built. From this initial set, 44 were eliminated because they were not technically correct: women did not remove their bras or the lungs were not completely in the image. Even so we kept some images that are not technically correct since they are over or underexposed. The database used for the experiments was then constituted by 130 images. Of these, 16 images present one occupied CDA (11 left CDA and 5 right CDA). We consider two parts in the experiment: the first consists in the correct detection of the diaphragm and external lung edges, allowing the correct identification of the CDAs' coordinates (segmentation). The second part is the detection of the occupancy of the CDA given already correctly segmented images (classification, which is presented in section 6.2). Note that the case of both occupied CDAs was not used since it is not represented in the database.

Segmentation

The algorithms were applied to the 130 images in the database. The segmentation results are presented in table 6.1.

Table 6.1: Percentage of error in the segmentation.

Right CDA	*Left CDA*
15.0%	26.7%

The evaluation of the segmentation was done considering that a correct segmentation of the CDA occurred if the external lung edge intercepted the diaphragm in the correct location and the diaphragm was correctly segmented in the zone of the CDA. The experiments show that the diaphragm is almost always correctly detected - even when there are problems, the portion close to the vertex of the CDA is detected. The main problems are large heart silhouette and air in the colon and stomach, and these affect only the left CDA (an example in figure 6.13). Sometimes the lateral lung edges are not completely detected preventing the CDAs from being defined. Figure 6.14 presents one example. When these cases occurred we completed manually the lung edge in order to be possible to define the CDA, so that all 130 images were available to the classification stage, described in the next section as problem 6.1.

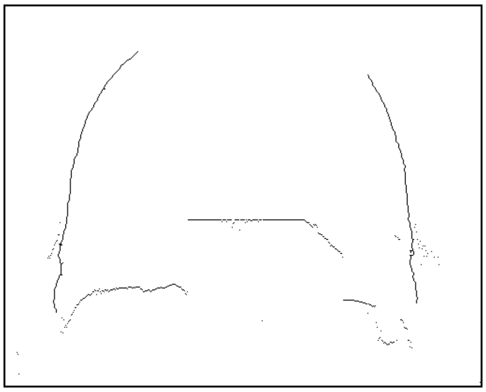

Figure 6.13: Left CDA misdetection due to air in the stomach and colon.

Figure 6.14: The CDAs are not defined due to incomplete detection of lateral lung edges.

6.2 Pre-processing of the data sets

The data sets used with NNs are usually pre-processed by making the features have zero mean (centered) and unit standard deviation (reduced) using a simple linear transformation [18, 36, 80, 82]. This is done to avoid the possibility that some of the features dominate the weight changes during training because they have values greater than others. This may happen when the features represent different quantities and are thus expressed in different units. Notice that some features may differ from others in orders of magnitude and this transformation avoids this disparity. This transformation usually speeds up convergence during training.

We tested the use of principal component analysis (PCA) [18, 80, 82] to make feature selection for the four classification problems in this chapter. We did not find a criterium to select the features after the PCA that, when applied to all the four problems, would improve the classification results of all the problems simultaneously (some would give less error while others presented more). We tested the following criteria: the Kaiser criterium (discard components with eigenvalues below 1) and the scree test (discard components starting on those where a plot of the eigenvalues in decreasing order, levels off) [82].

So, we decided to use all the features available and make no feature selection.

6.3 Classification: problem 6.1

Since this problem only has 3 classes (normal, right CDA occupied, left CDA occupied) it is more suitable for the application of the ensembles of classifiers discussed in chapter 4 than for the MNN architectures discussed in chapter 5.

6.3.1 Selection of the classifier

A MLP with increasing number of hidden layer neurons was used to solve the problem. The error was measured with a 10-fold cross-validation. The result is in figure 6.15. Since the error is quite small (about 3%), we decided that it would not be necessary to use combinations of classifiers or MNNs since a single MLP can solve the problem adequately.

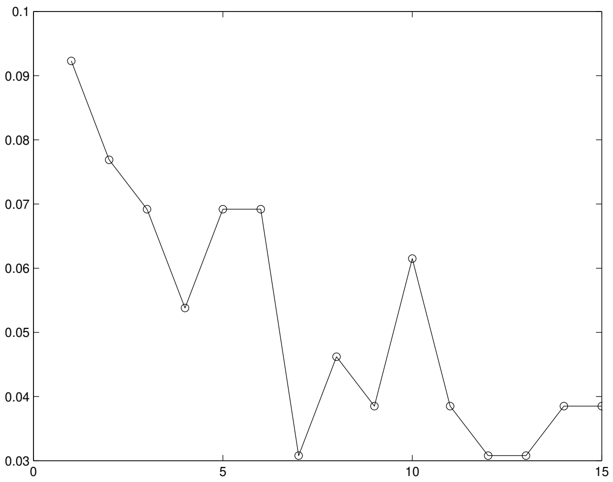

Figure 6.15: Error for data set 6.1 using an MLP with increasing number of hidden layer neurons. Values for a 10-fold cross validation.

We will use this problem to illustrate the variation in different types of error due to changes in the loss function.

We will use an MLP with 7 neurons on the hidden layer since it is the smallest number of hidden layer neurons that yielded the smallest error.

6.3.2 Variation of the loss function

In medical problems, not all errors are of equal importance. The most important error (the one that must be avoided at all cost) is the false negative error, which consists in not detecting an existing pathology.

It is possible to bias a classification system in order to decrease the false negative errors, at the expense of an increase of the other types of error, such as the false positive. This is done by manipulating the loss function. Instead of using the function presented in equation 2.3, it is possible to build a loss function which reflects the relative importance of the different types of error.

Two parameters that can be defined to allow the characterization of the classification system from a medical viewpoint are the sensitivity and the specificity.

The sensitivity is defined as the ratio between the number of pathological cases that were classified as pathological and the total number of pathological cases.

The specificity is defined as the ratio between the number of normal cases that

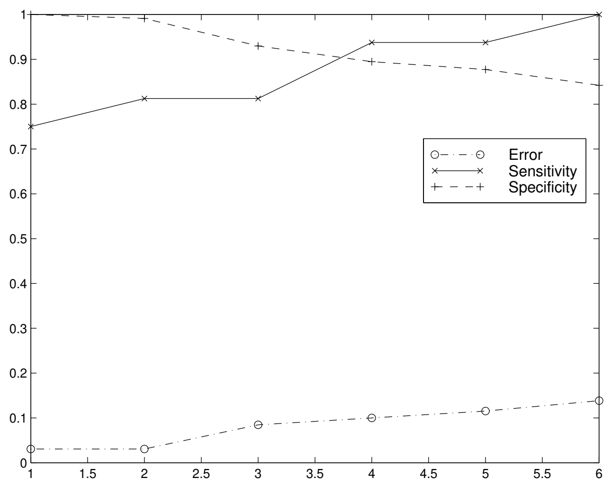

Figure 6.16: Evolution of the error, sensitivity and specificity for problem 6.1, as a function of an increasing penalty of the false negative error.

were classified as normal and the total number of normal cases.

From these definitions, we find that the false negative ratio equals 1-sensitivity, while the false positive ratio equals 1-specificity.

As discussed above, it is important that a medical classification system has a very small false negative ratio, which means a high sensitivity, such that the pathologies do not go undetected. If the classification system exhibits high specificity it means that it will have a small rate of false alarms [82].

To illustrate these concepts, we define the following loss function for this problem

$$\begin{pmatrix} 0 & 1 & 1 \\ i & 0 & 1 \\ i & 1 & 0 \end{pmatrix} \tag{6.3}$$

The value of i in the above expression is varied from 2 to 6. The matrix obtained with $i = 1$ is the same as the matrix mentioned before (in section 2.1.2), which gives all errors the same importance. Figure 6.16 presents the results. It shows how the sensitivity, specificity and the total error varied as the loss function was changed.

Notice how the sensitivity of the classifier increased as the loss function penalized the false negatives. This was possible at the expense of the specificity: the

Table 6.2: Confusion matrix for the initial loss function.

CDAs		Assigned Class		
		Both clear	Right occupied	Left occupied
Actual Class	Both clear	114	0	0
	Right occupied	3	2	0
	Left occupied	1	0	10

Table 6.3: Confusion matrix for the loss function with $i = 6$.

CDAs		Assigned Class		
		Both clear	Right occupied	Left occupied
Actual Class	Both clear	93	10	11
	Right occupied	0	5	0
	Left occupied	0	0	11

number of false alarms increases and so does the total error.

Table 6.2 shows the confusion matrix for the loss function with $i = 1$. The number of pathological cases missed is 4 and the total error is 3.08%.

Table 6.3 shows the confusion matrix for the loss function obtained from expression 6.3 using $i = 6$. No pathological cases are missed but the total error is 13.85%.

To conclude, we can say that this problem illustrated several aspects mentioned in the thesis. First, it showed a complete PR system as referred in chapter 1. All steps from the acquisition of the data, to the image processing necessary to obtain the features, to the final classification of the data were present. Then it showed how can the manipulation of the loss function be used to eliminate certain types of error that can be considered more dangerous than others. This is particularly relevant in medical applications such as those in this chapter. It also showed that some problems can be adequately dealt with by using simple classifiers, if the features have enough discriminative power. An ensemble was tried but no improvements from the 3% error could be obtained showing that sometimes it is not possible to improve classification even using more powerful classifiers, since the samples in error can be outliers or simple measurement/processing errors. Finally the problem was solved using a very simple classifier (a [11:7:3] MLP).

6.4 Finding the number of clusters using ISODATA

To find the number of clusters for the next data sets, we will use an automatic algorithm: the ISODATA (Iterative Self-Organizing Data Analysis Technique)[14].

Along with k-means [18, 80], this is one of the most used unsupervised classification algorithm. But unlike the k-means, the user does not define the number of clusters desired. The algorithm finds the number of clusters automatically, based on 3 parameters provided by the user. These parameters are: `mindist`, the minimum distance between any two clusters. If the distance is smaller than the algorithm merges them into one; `maxstdev`, the maximum standard deviation for any cluster. If a cluster has larger standard deviation it is split into two; `minpoints`, the minimum number of points any cluster is allowed. If a cluster has less than `minpoints` data points it is erased. The algorithm also receives an initial number of clusters, `k`. We note that this number cannot be greater than the actual number of classes in the problem, so this is a good value to use for the initialization.

Our goal is to find clusters that will be assigned to each expert in the modular architectures. This is necessary since these problems have more than 3 features and it is not possible to do the clustering by visual inspection as was done for the experiments in chapter 5.

Some post-processing of the output of the ISODATA algorithm was necessary since we use it to separate groups of classes and not classes. Details can be found in appendix B.

6.5 Problem 6.2: CTG analysis

This problem consists on the analysis of cardiotocographic (CTG) data (more precisely of fetal heart rate (FHR) signals, since CTG exams include other measurements not used here) to determine if there are problems with the fetus.

The data was obtained from [82]. It consists of 2126 data points, with 23 features. It is a problem with 10 classes.

The original data set contained 23 features but one of the features (DR) was discarded since it had the same value for all data points, and therefore contained no valuable information for the matter of discriminating among the classes. Also, the first and second features have identical values on all points in the data set.

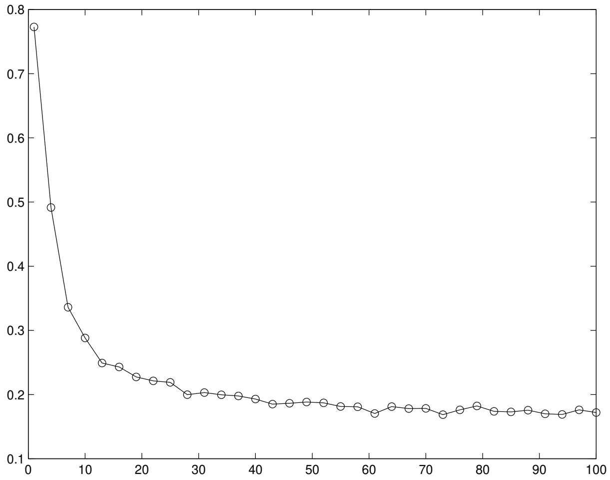

Figure 6.17: Error for data set 6.2 using an MLP with increasing number of hidden layer neurons. Values for a 10-fold cross validation.

So the first was discarded. Four other features (FM,DS,MODE,NZEROS) were discarded since, according to [82], are of little use in discriminating the classes. Finally, the remaining features were centered and reduced. A full description of the features can be found in the appendix A.

6.5.1 Size of the architectures

To decide on the size of the architectures to use in this problem, the size of the single MLP that is able to solve this problem was searched. The results of the error for a 10-fold cross validation are in figure 6.17.

Based on the data in the figure, we choose the number of neurons on the hidden layer to be 28. This was the chosen value since larger values only reduce the error by small amounts: the error obtained at this value (21.45%) is only 1.5% higher than the smallest error that occurred for the number of hidden layer neurons equal to 69 (19.94% error). The total number of weights in the first case is 820 while in the second case is 1873.

6.5.2 Number of experts

The ISODATA algorithm was used to find the number of clusters in the data. This is the number of experts that we will consider in the modular architectures. The parameters used in the algorithm were: `k=10`, `mindist=0.4`, `maxstdev=1500`, `minpoints=50`. The value of k used is the number of classes in the problem, as mentioned before. The values of the other 3 parameters were obtained empirically, after testing several combinations until the algorithm converges. This was also the case with the remaining two problems in this chapter. The algorithm found 4 clusters. Two of the clusters with 2 classes and the other two with 3 classes.

The other-output architecture was composed of 4 experts. The experts used in the subproblem with 2 classes had [17:9:3] topologies. The other two experts had [17:9:4] topologies. The total number of weights in the other-output MNN was 788.

The CMNN architecture was also composed of 4 experts. The experts used in the subproblem with 2 classes had [17:8:5] topologies. The other two experts had [17:8:6] topologies. The total number of weights in the other-output MNN was 792.

6.5.3 Error bound

With the error value obtained in section 6.5.1 with the MLP, 21.45%, it is possible to set a value from which it is not probable that the error will go beyond. Using the bound developed in chapter 3, and setting N=2126 testing points and $\hat{P}_s = 0.2145$ one finds the following value for $\alpha = 0.2845$ for a value of the upper bound of 2.4%. This means that, the probability that the error measured while testing on the complete set of points is larger than 28.45%, is smaller than 2.4%. As we will see, the result for the MLP is smaller than 28.45%. (Notice that the bound depends on the classifier used to estimate the error, and that is why the MLP is mentioned and not the MNNs).

Comparing MLP and MNNs with equal number of weights

The average results of 5 repetitions of 10-fold cross-validations are in table 6.4.

The different voting schemes in the CMNN architecture were represented by using '-vi' after 'CMNN', with $i = 1, \ldots, 4$. It represents the plurality vote, fuzzy vote, Nash vote and Borda count vote, respectively. The result obtained if there were no errors in the voting process is represented by 'CMNN-iv', where 'iv' stands for 'ideal vote'.

The best results are obtained with the other-output MNN (18.99% error). The second best architecture is the MLP with 20.67% error. The CMNN has its best result with the CMNN-v2 with 22.64% error. In this problem, the error introduced in the voting process is more than 10% for all variants of the CMNN. This indicates that the clusters were not very well defined, that is, this problem has a high overlap between the clusters.

Table 6.4: Errors for the data set of problem 6.2. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Architecture	Weights	Error [%]	Std. [%]
MLP	794	20.67	0.32
Other-output	788	18.99	0.21
CMNN-v1	792	22.82	0.30
CMNN-v2	792	22.64	0.33
CMNN-v3	792	26.19	0.83
CMNN-v4	792	22.76	0.28
CMNN-iv	792	12.11	0.25

Using an ensemble of MLPs

To achieve a better performance it is possible to use ensembles of classifiers combined using simple rules. This approach was discussed in chapters 2 and 4.

To illustrate this possibility and the kind of error reduction that can be achieved, we used 4 MLPs with the topology [17:28:10], the same used in the previous section by the isolated MLP.

The estimates of the posterior probabilities the MLPs produced were combined with 5 rules: maximum, mean, median, minimum and product. The results are in table 6.5

The best result was obtained with the minimum rule (19.06%). All the rules obtained very similar results, all between 19% and 20%, with the exception of the maximum which presented the worst result for the ensembles with 20.21% error. It

Table 6.5: Errors for the data set of problem 6.2 using an ensemble of 4 MLPs. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Combination rule	Error [%]	Std. [%]
Maximum	20.21	0.09
Mean	19.49	0.21
Median	19.76	0.21
Minimum	19.06	0.25
Product	19.41	0.24

should be noticed that, except for the result obtained with the other-output MNN, the results obtained with the ensembles are all better than the ones obtained by the architectures in table 6.4.

This is expected since the ensembles are using a larger number of weights overall than the other of the architectures and have thus, a higher learning capacity.

It is interesting to notice that, the other-output MNN, even confronted with the larger overall learning capacity of the ensembles, produced better results (18.99% error). This value is however, very close to the one obtained by the ensemble using the minimum rule. We have no clear justification for this fact. It may have resulted from some possible statistical fluctuation inherent to this type of experiments. Or it may be that this architecture is more adapted to this problem by some (unknown) reason.

6.6 Problem 6.3: Cleveland heart disease

This problem consists on the identification of heart infections. This is a 13 feature 5 class problem. The classes represent 4 types of infection and no infection (healthy). The data set has 303 points, but 7 had missing values and were removed. Thus, our version of the data set has 296 data points. Some of the features had nominal values. These were transformed into integer values. The data set was also centered and reduced. The data was obtained from [19].

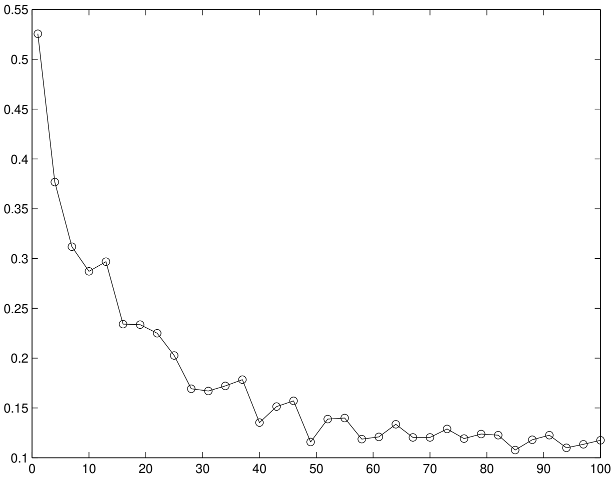

Figure 6.18: Error for data set 6.3 using an MLP with increasing number of hidden layer neurons. Values for a 10-fold cross validation.

6.6.1 Size of the architectures

To decide on the size of the architectures to use in this problem, the size of the single MLP that is able to solve this problem was searched. The error for a 10-fold cross validation as a function of the number of neurons in the hidden layer, is in figure 6.18.

Based on the data in the figure, the number of neurons on the hidden layer was chosen to be 40, since this value is clearly in the region where the decrease of the error with the increase in the number of neurons in the hidden layer has stopped.

This choice yields a total number of weights for the architectures used to solve this problem of 765 (the number of weights of the [13:40:5] MLP).

6.6.2 Number of experts

The ISODATA algorithm partitioned the data in two clusters with 2 and 3 classes each. The first cluster contains classes 0 (healthy) and 1, and the other cluster contains classes 2, 3 and 4.

The parameters used in the algorithm were: `k=5`, `mindist=0.5`, `maxstdev=100`, `minpoints=50`.

Both the other-output and the CMNN architectures are composed of two experts with the following topologies: [13:20:3] (343 weights) and [13:23:4] (418 weights), yielding a total of 761 weights, very close to the 765 weights of the MLP.

6.6.3 Classification results

Using the error value obtained with the MLP, 13.54%, it is possible to set a value from which it is not probable that the error will go beyond. Using the bound developed in chapter 3, and setting N=296 testing points and $\hat{P}_s = 0.1354$ one finds the following value for $\alpha = 0.2754$ for a value of the upper bound of 2.4%. This means that, the probability that the error measured while testing on the complete set of points is larger than 27.54%, is smaller than 2.4%. As we will see, the result for the MLP is smaller than this value.

Comparing MLP and MNNs with equal number of weights

Table 6.6: Errors for the data set of problem 6.3. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Architecture	Weights	Error [%]	Std.[%]
MLP	765	15.49	1.19
Other-output	761	17.78	0.84
CMNN-v1	761	17.76	0.57
CMNN-v2	761	18.09	0.55
CMNN-v3	761	18.78	0.52
CMNN-v4	761	17.76	0.57
CMNN-iv	761	12.90	0.61

The average errors for 5 repetitions of a 10-fold cross-validation are presented in table 6.6. The MLP shows the best performance (15.49% error) followed by the CMNN-v1 and CMNN-v4 both with 17.76% error. The other-output MNN outperformed the remaining versions of the CMNN. Notice that, given the standard deviations observed, it is not clear which is the best voting scheme for the CMNN or even if the other-output performs better than the CMNN. We can say that they have similar results.

It can be seen from the CMNN-iv result that the voting error for the CMNN is about 5%. The remaining error is due to the errors that the individual experts make in their own subproblems.

Using an ensemble of MLPs

To achieve a better performance it is possible to use ensembles of classifiers combined using some simple rules. These were discussed in chapter 4.

To illustrate this possibility and the kind of error reduction that can be achieved, we used 4 MLPs with the topology [13:40:5] as defined above.

The estimates of the posterior probabilities produced by the MLPs were combined with 5 rules: maximum, mean, median, minimum and product. The results are in table 6.7

Table 6.7: Errors for the data set of problem 6.3 using an ensemble of 4 MLPs. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Combination rule	Error [%]	Std. [%]
Maximum	15.65	0.36
Mean	13.35	0.31
Median	13.77	0.51
Minimum	13.44	0.33
Product	13.16	0.46

The best result was obtained using the product rule (13.16% error) and the worst result by the maximum rule (15.65% error). The result obtained with the maximum was the only one that was not better than all the results presented in table 6.6 for the isolated MLP and the MNNs. It was, never the less, comparable with the result of the isolated MLP, which achieved 15.49% error.

The remaining 4 rules presented results that are, at least, 4% better than the ones obtained using the MNNs.

As we mentioned before, this is not a fair comparison since the ensemble has a much larger number of weights than the architectures in table 6.6, but it gives an idea of the improvement that can be obtained, for this problem, using an ensemble.

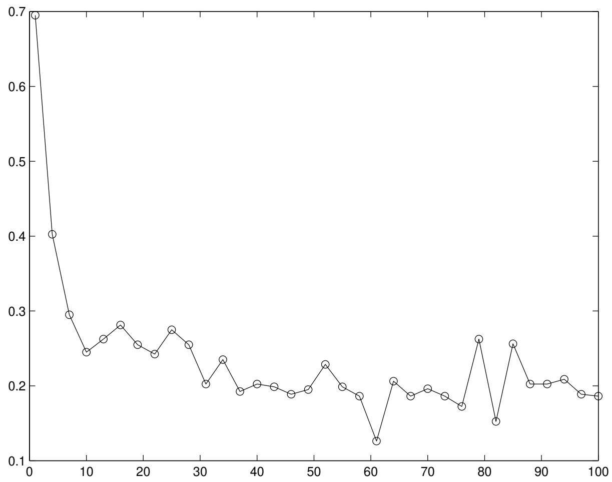

Figure 6.19: Error for data set 6.4 using an MLP with increasing number of hidden layer neurons. Values for a 10-fold cross validation.

6.7 Problem 6.4: Breast cancer detection

This problem consists on the identification of 6 types of breast tissue. The data set contains 106 points, with 9 features. A detailed description can be found in appendix A. The data was obtained from [82]. The features were centered and reduced.

6.7.1 Size of the architectures

To decide on the size of the architectures to use in this problem, the size of the single MLP that is able to solve this problem was searched. The results of the error in a 10-fold cross validation are in figure 6.19.

Based on the data in the figure, we choose the number of neurons on the hidden layer to be 31 (20.25% error) since for this value the MLP was in the region where the error decrease had stopped.

This choice yields a total number of weights for the architectures used to solve this problem of 502 (the number of weights on the [9:31:6] MLP).

6.7.2 Number of experts

The ISODATA algorithm partitioned the data set into 2 clusters, one with 4 classes and the other with 2 classes. The parameters used in the algorithm were: `k=6`, `mindist=1`, `maxstdev=15`, `minpoints=7`.

This yielded for the other-output and for the CMNN equal topologies for the experts (since there are only 2 clusters). These are [6:20:5] (305 weights) and [6:15:3] (198 weights) MLPs. The sum of the weights in the modular architectures gives 503 versus the 502 in the MLP.

6.7.3 Classification results

Once again we can use the bound developed in chapter 3, and the error value obtained with the MLP (20.25%), to set a value from which it is not probable that the error will go beyond. Setting N=106 testing points and $\hat{P}_s = 0.2025$ one finds the following value for $\alpha = 0.4525$ for a value of the upper bound of 2.1%. This mean that, the probability that the error measured while testing on the complete set of points is larger than 45.25%, is smaller than 2.1%. As we will see, the result for the MLP is smaller than this value. Notice how the value for α is considerably larger than the values for the other data sets, for similar upper bound (around 2%). This is due to the fact that this problem deals with a smaller data set which implies a higher variability in the outcome of the experiments.

Comparing MLP and MNNs with equal number of weights

The results obtained can be compared with published ones: in [85], the error rate obtained was 33.63%. It is interesting to note that, in that paper, a hierarchical approach was used. The first stage separated the classes into 2 groups (fatty tissue and the other). One with 2 and the other with 4 classes. Note that this was also the clustering result obtained here using the ISODATA algorithm. The cluster with 4 classes included the carcinoma, fibro-adenoma, mastopathy and glandular tissue and the connective and adipose tissue were in the other cluster.

These results are better than the cited ones, even with the simple MLP. The reason for this must be the pre-processing of the data and possibly the fact that, in the cited paper, the authors used linear discriminants instead of NNs.

Table 6.8: Errors for the data set of problem 6.4. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Architecture	Weights	Error [%]	Std.[%]
MLP	502	24.02	0.67
Other-output	503	24.87	1.05
CMNN-v1	503	23.67	1.24
CMNN-v2	503	23.55	1.27
CMNN-v3	503	23.55	1.27
CMNN-v4	503	23.67	1.24
CMNN-iv	503	23.55	1.27

Using an ensemble of MLPs

In this section, 4 MLPs with the topology [9:31:6] were used to evaluate the improvement in the classification error that this approach can produce. The estimates of the posterior probabilities produced by the MLPs were combined with the same 5 rules used in experiments 6.2 and 6.3. The results are in table 6.9

Table 6.9: Errors for the data set of problem 6.4 using an ensemble of 4 MLPs. Results obtained by averaging 5 repetitions of a 10-fold cross validation.

Combination rule	Error [%]	Std. [%]
Maximum	21.22	0.48
Mean	19.40	0.51
Median	19.80	0.49
Minimum	20.28	0.87
Product	19.20	0.49

The best result was obtained by the product rule with 19.20% error. The worst rule was the maximum with 21.22% error. All the results of the ensembles were better than the results obtained using the MNNs and the isolated MLP, as expected. For this data set the improvement is more than 4%. The results obtained by the different rules vary about 2%.

6.8 Conclusions

Table 6.10 presents a summary of the main aspects of the classification experiments in this chapter.

Table 6.10: Summary table for the problems in this chapter.

Data Set	6.1	6.2	6.3	6.4
Error (MPL and MNNs)	3.08%	18.99%	15.49%	23.55%
Best ensemble	-	19.06%	13.16%	19.20%
Best architecture	-	Other-output	MLP	CMNN
N. clusters	-	4	2	2
N. points	130	2126	296	106
N. classes	3	10	5	6
N. features (original/used)	11/11	23/17	13/13	9/9
N. weights	108	788	765	503
Cross-validation folds	10	10	10	10
Bound (2%)	-	28.45%	27.54%	45.25%

From the experiments that compared the single MLP with the MNNs with equal weights, we found that none of these architectures clearly outperformed the others, on all problems.

The reason why the CMNN had better results in previous comparisons, most notably in [10], is that the comparisons were not made using the same number of weights in both architectures. The number of weights available for the MLP was smaller than the number of weights available for the CMNN.

This can be illustrated with the data set 6.3. In [10], the CMNN model with 1240 weights is compared with an MLP with 271 weights. The error for the CMNN is 10.06% and for the MLP is 20.05%. We tested an MLP with 1240 weights for this problem, having the topology [13:65:5], using 10-fold cross-validation and found an error of 10.04%. This confirms the idea that if an equal number of weights is available to these architectures their performance is similar. With an advantage for the MLP since the training requires less effort: there is no need to use a clustering stage.

The conclusions drawn from the use of ensembles of MLPs are that they have a better performance than the other architectures, including the more elaborate MNNs (with one exception: in problem 6.2 the other-output MNN outperformed even the ensembles). This is expected since the ensembles involve the use of a larger number of weights than the other individual architectures.

It was clear with problem 6.1 that it is possible, by manipulating the loss function, to reduce significantly the number of false negatives at the expense of an increase in the false positives. Since this problem was easy from a classification point of view, it was not a good example for the use combinations of classifiers or MNNs to reduce the error yielded by the single MLP.

The error bound developed in chapter 3 was applied to the data sets and predicted, from the estimated errors using the isolated MLP when searching for the number of hidden layer neurons, a value for the error in the experiments that would probably (probability of about 98%) not be reached. These predictions were confirmed by the experiments.

6.9 Summary

This chapter contains applications of some of the results obtained in the previous chapters to problems involving medical data.

The problem of fluid detection in the lungs is an example of a 'complete' pattern recognition problem since all the steps from the raw data to the classification stages were present. It allowed the illustration of how to reduce the false negatives by manipulation of the loss function.

The other three addressed problems enabled the study of how ensembles and MNNs behave with real data from medical problems. The ensembles showed the best performance. The MNNs did not show much improvement (if any) over the isolated MLP.

The error bound developed in chapter 3 was applied and its predictions were validated.

The next chapter presents the general conclusions of the thesis and discusses possible future developments.

Chapter 7

Conclusion

7.1 Contributions

The author's contributions to the field of pattern recognition in the area of classification which are presented in this thesis are now reviewed.

An extensive review of the state of the art in the field of classifier combination was done. The methods were grouped into four categories according to their main features: static, adaptive, methods that generate new classifiers and architectural methods. A summary table synthesizing the most relevant features of these methods along with related paper references was also presented in chapter 2.

While studying the bounds on classification error, we developed distribution independent probabilistic bounds for classification error, based on Sanov's theorem. This is a theorem on the probability of a given distribution belonging to a set of distributions. It was adapted to the classification error context. This result enables the calculation of absolute error bounds (not dependent of the true error of the problem) for a classification problem.

Following the work of Tumer [93], we extended his results for the combination of classifiers using simple average to the case of a combination of classifiers using a weighted average, showing how the expected added error would vary and showing that, if all classifiers perform equally well, the simple average should be used for their combination (that is, an equal weight should be given to all classifiers) [2].

There is an ongoing debate on when to apply a sum rule or a product rule to a given classification task. We proved the equivalence of these two rules for any two class classification problem when combining only two classifiers (and their estimates of the posterior probabilities sum to one) and proved that this is the only case when these two rules are equivalent [4]. This was achieved in two steps: first we showed directly the equivalence of the rules in the case mentioned. Then we showed how to build examples in which the rules would yield different results for any other number of classes or number of classifiers involved in the problem.

The choice between a single NN and an MNN was also studied. In the case of a mixture of experts MNN architecture we developed an average error bound to predict its generalization error from the training set error and showed in what conditions it is expected that the modular architecture outperforms the single NN [1].

The decoupled, other-output and cooperative MNN were studied. We formalized their intuitive original descriptions, built models for the probability of correct classification and presented a unifying view of these three MNNs, based on these models. The models were built by considering the probability of the events involved in the decision process of each architecture. The unifying view of these architectures enables a clear understanding of how they work: part of the network learns the subproblem and the other part choses which of the experts should produce the final decision. The studied architectures only differ in the way to make this choice.

We applied the ensembles of MLPs with different combining rules and two of the MNNs studied in chapter 5 to tasks dealing with medical data. The first task involved medical image analysis. In particular, we worked with chest radiographs for the detection of the occupation of the costal diaphragmatic angles. Our results show a very good performance in terms of classification error (varying the loss function yields zero false negatives), but not so good results for the task of segmenting the lung edges (note that this is not a classification task but a image processing one).

The other three medical problems involved fetal heart signal analysis, heart disease detection and breast cancer detection. From the application of the MNNs and the isolated MLP to these problems we concluded that it is not clear that the more elaborated MNNs architectures outperform the single MLP. The evidence we gathered, points to the conclusion that given the same number of weights to either architecture they present equivalent results. On the other hand, the ensembles of

MLPs clearly outperform the isolated MLP and the MNNs. This was expected since they have a greater learning capacity due to the larger number of weights available.

7.2 Future work

The results on chapter 3 (the new bounds) and the results on the first part of chapter 4 (equivalence of sum and product under some conditions) are closed in the sense that it is not possible to extend them directly.

The results in the second part of chapter 4, can be extended if one makes no assumptions on the biases or on the error distributions of the classifiers.

It would be interesting to be able to find if a n-dimensional problem is separable, in the sense defined in chapter 5. This question arises from the results obtained therein that are only valid for separable problems.

It would also be of great importance to be able to use the models developed in chapter 5 for the MNNs to understand in what type of data sets a given architecture is probable to outperform the others. This is, we suspect, a quite difficult task.

It would also be interesting to make probabilistic models like the ones developed in chapter 5, for other architectures. This would help to achieve a greater insight on the way these methods work.

More generally, the study of what are the adequate combination methods for a given type of problem seems to be a good research direction. It is part of the model selection problem mentioned in chapter 5.

In terms of applications, the author is currently engaged in a CAD project at INEB, with the goal of detecting pathologies in chest radiographs. The application in section 6.1 is part of this project. Other abnormalities that are to be detected in these images include nodules and non-nodular opacities. We expect to apply the knowledge acquired during this thesis, namely the techniques of classifier combination (rules and architectures) and also the classification error bounds, to the classification phase of these important medical applications.

As a final comment, we would like to state that we feel that we achieved the original objective of this thesis, that consisted of contributing to a deeper theoretical understanding of how methods that combine classifiers work.

Appendix A

Data sets

This appendix presents more information for two of the data sets used in chapter 6. Both data sets were obtained from [82].

A.1 Data set used in experiment 6.2

This data set consists in features obtained from cardiotocographic examinations of 2126 foetuses. These examinations were made at S. João Hospital, Porto. Cardiotocography is a diagnostic method used in obstetrics which analyses foetal heart rate signals, uterine contractions and foetal movements. This data set only involves features based on the foetal heart rate signal. The classification was provided by expert obstectricians. The features are listed in table A.1. The measurements were made using SisPorto® which is a program for automated analysis of cardiotocograms, resulting from the collaboration between the Department of Obstetrics and Gynaecology of the Porto Faculty of Medicine and the Institute of Biomedical Engineering (INEB).

A.2 Data set used in experiment 6.4

This data set contains electrical impedance measurements in samples of fresh excised tissue from the breast.

Six classes of freshly excised tissue were studied using electrical impedance measurements: carcinoma, fibro-adenoma, mastopathy, glandular, connective and adipose.

Table A.1: Features of the data set used in experiment 6.2

LBE	baseline value (medical expert)
LB	baseline value (SisPorto)
AC	accelerations (SisPorto)
FM	foetal movement (SisPorto)
UC	uterine contractions (SisPorto)
aSTV%	percentage of time with abnormal short term variability (SisPorto)
mSTV	mean value of short term variability (SisPorto)
aLTV%	percentage of time with abnormal long term variability (SisPorto)
mLTV	mean value of long term variability (SisPorto)
Dl	light decelerations
Ds	severe decelerations
Dp	prolongued decelerations
Dr	repetitive decelerations
Width	histogram width
Min	low freq. of the histogram
Max	high freq. of the histogram
Nmax	number of histogram peaks
Nzeros	number of histogram zeros
Mode	histogram mode
Mean	histogram mean
Median	histogram median
Variance	histogram variance
Tendency histogram	tendency: -1=left assymetric; 0=symmetric; 1=right assymetric

Table A.2: Features of the data set used in experiment 6.4

I0	impedivity (ohm) at zero frequency
PA500	phase angle at 500 KHz
HFS	high-frequency slope of phase angle
DA	impedance distance between spectral ends
AREA	area under spectrum
A/DA	area normalized by DA
MAX IP	maximum of the spectrum
DR	distance between IO and the real part of the max. frequency point
P	length of the spectral curve

Impedance measurements were made at the frequencies: 15.625, 31.25, 62.5, 125, 250, 500, 1000 kHz. These measurements plotted in the (real, -imaginary) plane constitute the impedance spectrum from where the features were computed. The features are listed in table A.2. The source for this data is Dr. Jacques Jossinet, at INSERM U.281, Lyon, France.

Appendix B

The ISODATA algorithm

This section describes the original ISODATA algorithm and the post-processing implemented to make it applicable to the MNNs used in the problems of chapter 6.

B.1 Description of the algorithm

As described in section 6.4, this algorithm is used for unsupervised classification. The algorithm finds the number of clusters automatically, based on 3 parameters provided by the user. These parameters are: `mindist`, the minimum distance between any two clusters. If the distance between any two clusters is smaller than this, the algorithm merges them in one. The other parameter is `maxstdev`, the maximum standard deviation for any cluster: a cluster with larger standard deviation is split in two. The third parameter is `minpoints`, the minimum number of points any cluster is allowed. If a cluster has less than `minpoints` data points it is erased. The algorithm also receives an initial number of clusters. The ISODATA algorithm follows.

```
REPEAT
   Classify input to nearest mean
   Recalculate class means
   Report classification result
   IF (two classes are closer than <mindist>)
```

```
        THEN merge them
      ELSEIF (a class has stdev larger than <maxstdev>)
        THEN split it
      ELSEIF (a class has fewer than <minpoints> data points)
        THEN remove it
      ENDIF
      IF (any such changes)
        THEN Recalculate class means
    UNTIL no changes
```

B.2 Post-processing of the ISODATA results

The output of this algorithm is not directly applicable to the MNN architectures. This section describes the post-processing applied to it's output.

Each of the expert modules of the MNN architectures referred in this thesis, deals with a group of classes. This means that when the ISODATA algorithm groups the classes there should be no points of the same class in two or more groups. Since this may happen with the ISODATA algorithm, which clusters the points by their proximity regardless of their original classes, the post-processing of the output of the ISODATA algorithm changes points that are in two or more groups to the group with more points of that original class.

Another step of the post-processing assures that the clusters have groups of classes with sequential names. This is needed because the code developed for the classifiers expects the classes to be sequentially numbered. According to the clustering made by this algorithm, a vector is produced which translates the original names of the classes to the new names, insuring that all classes in a given group are named sequentially.

Bibliography

[1] L.A. Alexandre, A.C. Campilho, and M. Kamel. Average error bound for the mixture of experts MNN architecture. In *to appear in the Proceedings of the 12th Portuguese Conference on Pattern Recognition*.

[2] L.A. Alexandre, A.C. Campilho, and M. Kamel. Combining independent and unbiased classifiers using weighted average. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 495–498, Barcelona, Spain, September 2000. IEEE Press.

[3] L.A. Alexandre, A.C. Campilho, and M. Kamel. Combining unbiased and independent classifiers using geometric mean. In *Proceedings of the 11th Portuguese Conference on Pattern Recognition*, pages 399–401, Porto, Portugal, May 2000.

[4] L.A. Alexandre, A.C. Campilho, and M. Kamel. On combining classifiers using sum and product rules. *Pattern Recognition Letters*, 22:1283–1289, 2001.

[5] L.A. Alexandre, M.J. Silva, and A.C. Campilho. Automatic detection of the occupation of the costal diaphragmatic angle. In *Workshop on Medical Imaging*, pages 15–18, Aveiro, Portugal, 1999.

[6] L.A. Alexandre, M.J. Silva, and A.C. Campilho. Detection of the occupation of the costal diaphragmatic angle in chest radiographs. *Electrónica e Telecomunicações*, 3(2):187–192, 2000.

[7] F.M. Alkoot and J. Kittler. Improving the performance of the product fusion strategy. In *Proceedings of the 15th International Conference on Pattern*

Recognition, volume 2, pages 164–167, Barcelona, Spain, September 2000. IEEE Press.

[8] F.M. Alkoot and J. Kittler. Improving the performance of the product fusion strategy. *Pattern Recognition Letters*, 23:957–965, 2002.

[9] E. Alpaydin. Techniques for combining multiple learners. In E. Alpaydin, editor, *Proceedings of Engineering of Intelligent Systems98*, volume 2, pages 6–12. ICSC Press, 1998.

[10] G. Auda. *Cooperative Modular Neural Network Classifiers.* PhD thesis, University of Waterloo, Ontario, Canada, 1996.

[11] G. Auda and M. Kamel. CMNN: Cooperative modular neural networks for pattern recognition. *Pattern Recognition Letters*, 18:1391–1398, 1997.

[12] G. Auda and M. Kamel. Modular neural network classifiers: A comparative study. *J. Intel. Robotic Systems*, (21):117–129, 1998.

[13] G. Auda, M. Kamel, and H. Raafat. Voting schemes for cooperative neural network classifiers. In *IEEE Int. Conference on Neural Networks*, volume 3, pages 1240–1243, Australia, November 1995.

[14] G.H. Ball and D.J. Hall. Isodata, an iterative method of multivariate analysis and pattern classification. In *Proceedings of the IFIPS Congress*, 1965.

[15] R. Battiti and A.M. Colla. Democracy in neural networks: Voting schemes for classification. *Neural Networks*, 7(4):691–707, 1994.

[16] S. Behnke, M. Pfister, and R. Rojas. A study on the combination of classifiers for handwritten digit recognition. In *Proceedings NN'98*, pages 39–46, Magdeburg, 1998.

[17] J.A. Benediktsson, J.R. Sveinsson, O.K. Ersoy, and P.H. Swain. Parallel consensual neural networks. *IEEE Trans. Neural Networks*, 8(1):5–17, January 1997.

[18] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[19] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[20] F. Blayo, Y. Cheneval, A. Guérin-Dugué, C. Chentouf, C. Aviles-Cruz, J. Madrenas, M. Moreno, and J.L. Voz. Deliverable R3-B4-P - Task B4:benchmarks. Technical Report 6891, ESPRIT-Basic Research Project, June 1995.

[21] L. Breiman. Arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996.

[22] L. Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California, Berkeley, 1997.

[23] L. Breiman. Using convex pseudo-data o increase prediction accuracy. Technical Report 513, Statistics Department, University of California, Berkeley, 1997.

[24] I. Bronstein and K. Semendiaev. *Manual de matemática para engenheiros e estudantes*. Editora MIR, Moscovo, 1984.

[25] W.L. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, School of Computer Science, University of Technology, Sidney, 1992.

[26] M.J. Carreira, D. Cabello, M.G. Penedo, and A. Mosquera. Computer-aided diagnosis: Automatic detection of lung nodules. *Med. Phys.*, 25(10):1998–2006, October 1998.

[27] K. Chen, L. Wang, and H. Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 11(3):417–445, 1997.

[28] S.B. Cho. Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Trans. Neural Networks*, 8(1):43–53, January 1997.

[29] S.B. Cho and J.H. Kim. Multiple network fusion using fuzzy logic. *IEEE Trans. Neural Networks*, 6(2):497–501, March 1995.

[30] W.G. Cochran. *Sampling techniques.* John Wiley & Sons, 1977.

[31] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[32] M. De Bollivier, P. Gallinari, and S. Thiria. Cooperation of neural nets and task decomposition. In *Int. Joint Conf. on Neural Networks*, volume 2, pages 573–576, Seattle,USA, 1991.

[33] H. Demuth and M. Beale. *Neural Network Toolbox User's Guide.* The MathWorks, Inc., 1998.

[34] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.

[35] H. Drucker, C. Cortes, L.D. Jackel, Y. Lecun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, (6):1289–1301, 1994.

[36] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, 2001.

[37] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[38] C.M. Friedrich. Ensembles of evolutionary created artificial neural networks. In W. Grauel, editor, *Proceedings of the 5th International Workshop on Fuzzy-Neuro-Systems (FNS'98)*, pages 250–257, Munich, Germany, 1998.

[39] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[40] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33:2099–2101, 2000.

[41] G. Fumera, F. Roli, and G. Vernazza. A method for error rejection in multiple classifier systems. In *Proceedings of 11th International Conference on Image Analysis and Processing, ICIAP2001*, pages 454–458, Palermo, Italy, September 2001.

[42] J. Gama. Combining classifiers by constructive induction. In *Proceedings ECML-98. Lecture Notes in Artificial Intelligence*. Springer Verlag, 1998.

[43] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, (4):1–58, 1992.

[44] G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22:25–33, 2001.

[45] B. van Ginneken and B.M.H. Romeny. Automatic segmentation of lung fields in chest radiographs. In *MICCAI'99, Lecture Notes in Computer Science*, number 1679, pages 184–191, 1999.

[46] C.W.J. Granger and R. Ramanathan. Improved methods for combining forecasts. *Journal of Forecasting*, 3:197–204, 1967.

[47] H. Gu and H. Takahashi. Towards more practical average bounds on supervised learning. *IEEE Trans. Neural Networks*, 7(4):953–968, July 1996.

[48] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. PAMI*, 12(10):993–1001, October 1990.

[49] S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997.

[50] S. Haykin. *Neural Networks: A Comprehensive Foundation*. MacMillan College Publishing Company, Inc., 1994.

[51] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifier systems. *IEEE Trans. PAMI*, 16(1):66–75, January 1994.

[52] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[53] T. Hrycej. *Modular Learning in Neural Networks*. John Wiley & Sons, 1992.

[54] Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. PAMI*, 17(1):90–94, January 1995.

[55] R. Jacobs. Method for combining experts' probability assessments. *Neural Computation*, (7):867–888, 1995.

[56] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, (3):79–87, 1991.

[57] R.A. Jacobs, F. Peng, and M.A. Tanner. A bayesian approach to model selection in hierarchical mixtures-of-experts architectures. *Neural Networks*, 10(2):231–241, 1997.

[58] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1987.

[59] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Trans. PAMI*, 22(1):4–37, January 2000.

[60] A.K. Jain and J. Mao. Guest editorial: Special issue on artificial neural networks and statistical pattern recognition. *IEEE Trans. Neural Networks*, 8(1):1–4, January 1997.

[61] A.K. Jain, S. Prabhakar, and S. Chen. Combining multiple matchers for a high security fingerprint verification system. *Pattern Recognition Letters*, 20:1371–1379, 1999.

[62] C. Ji and S. Ma. Combination of weak classifiers. *IEEE Trans. Neural Networks*, 8(1):32–42, January 1997.

[63] M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27(1), 1997.

[64] M.J. Kearns and U.M. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, 1994.

[65] F. Kimura and M. Shridhar. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, 24(10):969–983, 1991.

[66] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Trans. PAMI*, 20(3):226–239, March 1998.

[67] J. Kittler, M. Hatef, and R.P.W. Duin. Combining classifiers. In *Proceedings of the International Conference on Pattern Recognition'96*, pages 897–901, 1996.

[68] S.M. Lai, X. Li, and W.F. Bischof. Automated detection of breast tumors. In A. Krzyzak, T. Kasvand, and C.Y: Suen, editors, *Computer Vision and Shape Recognition*, pages 115–132. World Scientific, 1989.

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

[70] C.J. Merz. *Classification and regression by combining models.* PhD thesis, University of California, Irvine, USA, 1998.

[71] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, (11):169–198, 1999.

[72] C.S. Pedrosa and R. Casanova. *Diagnostico por Imagen.* Interamericana - McGraw-Hill, 1987.

[73] I.B. Pena. *Radiolagia Clinica del Torax.* Ediciones Toray, S.A., Barcelona, 1970.

[74] M.G. Penedo, M.J. Carreira, A. Mosquera, and D. Cabello. Computer-aided diagnosis: A neural-network based approach to lung nodule detection. *IEEE Trans. Medical Imaging*, 17(6):872–880, December 1998.

[75] W.D. Penny, D. Husmeier, and S.J. Roberts. Covariance-based weighting for optimal combination of model predictions. In Springer-Verlag, editor, *ICANN-99, Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 826–831, 1999.

[76] J.R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings AAAI-96 Fourteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

[77] J.R. Quinlan. Boosting first-order learning. In *Algorithmic Learning Theory, 7th International Workshop, ALT'96*, Sydney, Australia, 1996. Lecture Notes in Artificial Intelligence 1160, Springer-Verlag.

[78] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., 1993.

[79] M. Richard and R. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, (3):461–483, 1991.

[80] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[81] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[82] J.P.M. Sá. *Pattern Recognition: Concepts, Methods and Applications.* Springer, 2001.

[83] R. Schapire. The strength of weak learnability. *Machine learning*, 5(2), 1990.

[84] T. Scheffer and T. Joachims. Expected error analysis for model selection. In *Proceedings of the 16th International Conference on Machine Learning*, pages 361–370. Morgan Kaufmann, San Francisco, CA, 1999.

[85] J.E. Silva, J.P.M. Sá, and J. Jossinet. Classification of breast tissue by electrical impedance spectroscopy. *Medical & Biological Engineering & Computing*, 38(1):26–30, January 2000.

[86] P. Smagt. The bootstrap. http://www.robotic.dlt.de/Smagt/research/feedforward/bootstrap.

[87] D. Tax, R. Duin, and M. Breukelen. Comparison between product and mean classifier combination rules. In *Workshop on Statistical Techniques in Pattern Recognition*, Prague, Czech Republic, 1997.

[88] K.M. Ting and I.H. Witten. Stacked generalization: when does it work ? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 866–871, 1997.

[89] K.M. Ting and I.H. Witten. Stacking bagged and dagged models. In *Proceedings of The Fourteenth International Conference on Machine Learning*, pages 367–375, 1997.

[90] O. Tsujii, M.T. Freedmann, and S.K. Mun. Automated segmentation of anatomic regions in chest radiographs using an adaptive-sized hybrid neural network. *Med. Phys.*, 25(6):998–1007, June 1998.

[91] K. Tumer. *Linear and Order Statistical Combiners for Reliable Pattern Classification.* PhD thesis, University of Texas, Austin, USA, 1996.

[92] K. Tumer and J. Ghosh. Boundary variance reduction for improved classification through hybrid networks. In *AeroSense 95: Applications and Science of Artificial Neural Networks, Proceedings of the SPIE*, volume 2492, pages 573–584, Orlando, 1995.

[93] K. Tumer and J. Gosh. Linear and order statistics combiners for pattern classification. In A. Sharkey, editor, *Combining Artificial Neural Nets*, pages 127–161. Springer-Verlag, 1999.

[94] N. Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Trans. PAMI*, 22(2):207–215, February 2000.

[95] M. van Breukelen, R.P. Duin, and D.M.J. Tax. Combining classifiers for the recognition of handwritten digits. In *Proc. 1st IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition*, pages 13–18, June 1997.

[96] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis. Soft combination of neural classifiers: a comparative study. *Pattern Recognition Letters*, 20:429–444, 1999.

[97] N. Wanas, M.S. Kamel, G. Auda, and F. Karray. Feature-based decision aggregation in modular neural network classifiers. *Pattern Recognition Letters*, 20:1353–1359, 1999.

[98] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[99] D.H. Wolpert and W.G. Macready. Combining stacking with bagging to improve a learning algorithm. Technical Report 96-03-123, Santa Fe Institute, Santa Fe, 1996.

[100] K. Woods, W.P. Kegelmeyer Jr., and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. PAMI*, 19(4):405–410, April 1997.

[101] L. Xu, A. Krzyzak, and C.Y. Suen. Methods for combining multiple classifiers and their applications to handwritten recognition. *IEEE Trans. SMC*, 22(3):418–435, 1992.

[102] X. Xu, K. Doi, T. Kobayashi, H. MacMahon, and M.L. Giger. Development of an improved CAD scheme for automated detection of lung nodules in digital chest images. *Med. Phys.*, 24(9):1395–1403, September 1997.

[103] H. Yoshida, S. Katsuragawa, Y. Amit, and K. Doi. Wavelet snake for classification of nodules and false positives in digital chest radiographs. In *19th International Conference IEEE/EMBS*, pages 509–512, Chicago,IL,USA, 1997.

[104] A.L. Yuille and J.M. Coughlan. Fundamental limits of bayesian inference: order parameters and phase transitions for road tracking. *IEEE Trans. PAMI*, 22(2):160–173, February 2000.

Index