INFORMATION THEORETIC COMBINATION OF CLASSIFIERS WITH APPLICATION TO FACE DETECTION

THÈSE N^{O} 3951 (2007)

PRÉSENTÉE LE 23 NOVEMBRE 2007

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE TRAITEMENT DES SIGNAUX 5

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Julien MEYNET

DEA signal, image, parole, télécoms, Institut national polytechnique de Grenoble, France
et de nationalité française

acceptée sur proposition du jury:

Prof. H. Bourlard, président du jury
Prof. J.-Ph. Thiran, directeur de thèse
Prof. A. Billard, rapporteur
Prof. H. Bunke, rapporteur
Prof. J. Kittler, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2007

Abstract

Combining several classifiers has become a very active subdiscipline in the field of pattern recognition. For years, pattern recognition community has focused on seeking optimal learning algorithms able to produce very accurate classifiers. However, empirical experience proved that is is often much easier finding several relatively good classifiers than only finding one single very accurate predictor. The advantages of combining classifiers instead of single classifier schemes are twofold: it helps reducing the computational requirements by using simpler models, and it can improve the classification skills. It is commonly admitted that classifiers need to be complementary in order to improve their performances by aggregation. This complementarity is usually termed as diversity in classifier combination community. Although diversity is a very intuitive concept, explicitly using diversity measures for creating classifier ensembles is not as successful as expected.

In this thesis, we propose an information theoretic framework for combining classifiers. In particular, we prove by means of information theoretic tools that diversity between classifiers is not sufficient to guarantee optimal classifier combination. In fact, we show that diversity and accuracies of the individual classifiers are generally contradictory: two very accurate classifiers cannot be diverse, and inversely, two very diverse classifiers will necessarily have poor classification skills. In order to tackle this contradiction, we propose a information theoretic score (ITS) that fixes a trade-off between these two quantities. A first possible application is to consider this new score as a selection criterion for extracting a good ensemble in a predefined pool of classifiers. We also propose an ensemble creation technique based on AdaBoost, by taking into account the information theoretic score for iteratively selecting the classifiers.

As an illustration of efficient classifier combination technique, we propose several algorithms for building ensembles of Support Vector Machines (SVM). Support Vector Machines are one of the most popular discriminative approach of pattern recognition and are often considered as state-of-the-art in binary classification. However these classifiers present one severe drawback when facing a very large number of training examples: they become computationally expensive to train. This problem can be addressed by decomposing the learning into several classification tasks with lower computational requirements. We propose to train several parallel SVM on subsets of the complete training set. We develop several algorithms for designing efficient ensembles of SVM by taking into account our information theoretic

score.

The second part of this thesis concentrates on human face detection, which appears to be a very challenging binary pattern recognition task. In this work, we focus on two main aspects: feature extraction and how to apply classifier combination techniques to face detection systems. We introduce new geometrical filters called anisotropic Gaussian filters, that are very efficient to model face appearance. Finally we propose a parallel mixture of boosted classifier for reducing the false positive rate and decreasing the training time, while keeping the testing time unchanged. The complete face detection system is evaluated on several datasets, showing that it compares favorably to state-of-the-art techniques.

Keywords: pattern recognition, classifier combination, information theory, support vector machines, AdaBoost, face detection.

Version Abrégée

La combinaison de classificateurs est devenue une sous-discipline très active dans le domaine de reconnaissance des formes. Pendant des années, les plus gros efforts dans le domaine de l'apprentissage automatique se sont concentrés sur l'élaboration d'algorithmes produisant des classificateurs optimaux, très robustes. Cependant, de nombreuses études empiriques ont montré qu'il s'avère plus aisé de trouver plusieurs classificateurs relativement performants plutôt qu'un seul prédicteur très robuste. Les avantages de combiner plusieurs classificateurs au lieu d'utiliser un classificateur unique, sont doubles : réduire la complexité des modèles d'une part, et améliorer les performances de classification d'autre part. Il est communément admis que les classificateurs doivent être complémentaires afin d'obtenir une combinaison efficace. Dans le domaine, cette complémentarité est habituellement appelée diversité. Bien que cette diversité soit un concept très intuitif, les méthodes ayant tenté d'employer explicitement les mesures de diversité pour créer des ensembles de classificateurs n'ont jusqu'alors pas eu le succès escompté.

Dans cette thèse, nous proposons d'abord un nouveau cadre théorique pour la combinaison de classificateurs, basé sur la théorie de l'information. En particulier, nous prouvons, à l'aide des outils de théorie de l'information tels que l'information mutuelle, que la diversité entre les classificateurs n'est pas suffisante pour garantir une combinaison optimale. En effet, nous montrons que la diversité et la précision des différents classificateurs sont généralement deux notions contradictoires: deux classificateurs très précis ne peuvent pas être divers, et inversement, deux classificateurs très divers auront nécessairement des performances de classification faibles. Afin de palier cette contradiction, nous proposons un score (appelé *ITS*) fixant un compromis entre la diversité et la moyenne des précisions des classificateurs. Une première application possible est de considérer ce nouveau score comme un critère d'optimisation pour extraire une combinaison optimale, parmi un ensemble prédéfini de classificateurs . Nous proposons également une technique de création d'ensembles basée sur AdaBoost, en tenant compte du score *ITS* pour entrainer itérativement des classificateurs.

Ensuite, nous illustrons ce cadre théorique en proposant plusieurs algorithmes pour construire des ensembles efficaces de machines à vecteurs de support (*SVM*). Les machines à vecteurs de support sont parmi les méthodes les plus populaires des approches discriminantes à la reconnaissance des formes. Elles sont souvent considérées comme état de l'art

en classification binaire. Cependant, ces classificateurs présentent un inconvénient majeur: ils deviennent très compliqués à entraîner lorsque la quantité de données d'apprentissage est importante. Ce problème peut être résolu en le décomposant en plusieurs sous-problèmes de moindre complexité. Nous proposons de former plusieurs *SVM* en parallèle, entraînés indépendamment sur des sous-ensembles des données d'apprentissage. Nous développons plusieurs algorithmes pour concevoir des ensembles efficaces de *SVM* en prenant en compte notre score *ITS*.

La deuxième partie de cette thèse se concentre sur la détection automatique de visages dans les images. Cette application peut être formulée comme un problème ambitieux de reconnaissance des formes. Ce travail se focalise principalement sur deux points: l'extraction d'attributs discriminants et l'utilisation de techniques de combinaison de classificateurs dans les systèmes de détection de visage. Nous présentons de nouveaux filtres géométriques appelés filtres gaussiens anisotropiques, qui s'avèrent très efficaces pour modéliser l'apparence des visages. Enfin nous proposons un ensemble de classificateurs parallèles, entraînés par Boosting, pour réduire le nombre de fausses détections et diminuer le temps de d'apprentissage, tout en gardant une vitesse de test stable. Le système complet de détection de visage est évalué sur plusieurs bases de données, montrant des améliorations significatives par rapport l'état de l'art.

Mots-clés: reconnaissance des formes, combinaison de classificateurs, théorie de l'information, machines à vecteurs de support, AdaBoost, détection de visages.

Remerciements

Au terme de cette thèse, j'aimerais remercier les personnes qui ont contribué à ce travail de recherche, par leur confiance, leur patience, leur compétence et leurs conseils précieux. Mes tous premiers remerciements s'adressent à mon directeur de thèse, Professeur Jean-Philippe Thiran qui, en m'intégrant dans son laboratoire, m'a donné l'opportunité de travailler au sein d'une équipe soudée et amicale. Je lui suis reconnaissant pour sa disponibilité, ses conseils judicieux et sa bonne humeur quotidienne qui contribue à l' excellente ambiance de travail. Je remercie également le Professeur Murat Kunt de m'avoir accueilli dans l'institut de traitement des signaux de renommée internationale. Ma reconnaissance va aussi aux membres du jury de thèse: Professeur Aude Billard, Professeur Horst Bunke, Professeur Josef Kittler et le président du jury Professeur Hervé Bourlard pour leurs commentaires productifs et suggestions pendant la défense de thèse. Je sais gré à Vlad, le "gourou" de reconnaissance des formes au sein du laboratoire, toujours disponible pour donner le bon conseil au bon moment, pour son humour, et pour son amitié durant ces dernières années. Je n'oublie pas de saluer les membres du laboratoires pour leur assistance administrative et informatique, ainsi que pour leur sourire chaque jour: Marianne Marion, Gilles Auric, Christophe Aeschliman et Simon Châtelain. Merci aussi à Matteo, mon collègue de bureau avec qui j'ai partagé à la fois éclats de rire et discussions fructueuses. J'adresse aussi mes remerciements à tous mes autres collègues du laboratoire, les grimpeurs, les pongistes et babyfootistes et autres, complices de tant d'instants inoubliables. Ma plus grande reconnaissance va à ma famille, soutien permanent et bien plus que cela: mes parents Christiane et Jean, mon frère Jérôme, Sonia et Amélie, mes grand-parents et tante. Et puis il y a Irene...

On peut apprendre à un ordinateur à dire: "je t'aime", mais on ne peut pas lui apprendre à aimer.
Albert Jacquard.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations and Symbols

Mathematical notations

A, B	matrices are denoted by capital letters
$\mathbf{x}, \mathbf{w}$	vectors are denoted by bold lower–case letters and are column vectors
$\mathbf{x}'$	the transposed of vector $\mathbf{x}$
A'	the transposed of matrix A
$\langle \cdot, \cdot \rangle$	inner product operator
$\| \cdot \|$	norm operator
$\lceil x \rceil$	the smallest integer grater than x
$\lfloor x \rfloor$	the largest integer smaller than x
$\|A\|$	determinant of a matrix A
$\|\Omega\|$	cardinality for a set Ω
$E[\cdot]$	expectation operator

List of Symbols

ANN	Artificial Neural Networks
BKS	Behavior Knowledge Space
DFFS	Distance From Feature Space
ESV	Ensemble Support Vectors
GA	Genetic Algorithm
GASVMs	Genetic Algorithms based Mupltiple Support Vector Machines
GF	Gaussian Filters
GSVMs	Gated Support Vector Machines
HF	Haar-like Filters
IT	Information Theory
ITA	Information Theoretic Accuracy
ITD	Information Theoretic Diversity
ITS	Information Theoretic Score
KA	Kernel Adatron
KASVMs	Kernel AdaTron for Mupltiple Support Vector Machines
$k-NN$	K-Nearest Neighbors

LDA	Linear Discriminant Analysis
LBP	Local Binary Patterns
MI	Mutual Infirmation
MLP	Multi Layer Perceptron
MSVMs	Multiple Support Vector Machines
MV	Majority Voting
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
QS	Q-statistics
RBF	Radial Basis Functions
ROC	Receiver Operating Characteristic
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
VC	Vapnik-Chervonenkis
WITA	Weighted Information Theoretic Accuracy
WITD	Weighted Information Theoretic Diversity
WITS	Weighted Information Theoretic Score

1 Introduction

Face detection has been an active application in the field of computer vision for the last two decades, mainly because of the increasing number of potential applications in biometrics, content-based image retrieval, video conferencing or intelligent human-computer interfaces. However, detecting human faces in images can also be viewed as a very challenging machine learning task and is commonly used for the evaluating the robustness of pattern recognition techniques, from feature selection and extraction steps to classification.

The challenge of face detection in still images comes from the large variability of the face appearance due to many intra-personal and extra-personal factors, such as the modification of the face appearance with changes in illumination or head pose. Face detection can be turned into a pattern recognition task by using a sliding window that scans the image at different scales. At each position, a classifier checks the presence of a potential face.

A large number of techniques have been proposed for solving this classification problem and, one very popular classification technique for this kind of applications is to combine several classifiers to form an ensemble of classifiers. Combining classifiers was originally employed for helping the implementation of learning algorithms having large computation requirements, like Neural Networks for instance, the underlying motivation being that it is generally much easier finding several quite good classifiers than building one single very performant classifier. Then, classifier combination became a very active field in the machine learning community as it proved to be very effective in many applications, not only for reducing training complexity but also in terms of classification performances. Face detection is a typical problem for which combining classifiers can improve significantly the performances. On the one hand, learning a robust face model requires a huge number of training patterns, and, dealing with very large training sets presents two major drawbacks: the training procedure becomes very computationally expensive and the risk of having outliers and noisy examples is increased. Face detection can take advantage of classifier combination

techniques from four different perspectives:

- Reduce the training complexity. This step is very important in face detection as the total training time can be reduced from weeks to days;
- Reduce testing time. As most of the applications require real-time face detectors, large efforts should be put on reducing the time needed to process one single frame;
- Improve the overall performances of the system. Combining classifier can help reducing number of false detections;
- Obtain sparser models. It can produce models with low storage requirements. This should be considered as a important factor in embedded systems.

Classifier combination techniques can be divided into two main categories, aggregation of classifiers outputs or ensemble creation techniques. On the first hand, aggregation techniques dispose of a set of predefined classifiers. A combination rule then takes the classification decision. The combination can be performed at the decision level (using the class labels) or at a score level (usually represented by posterior probabilities). In this category, usual applications concerns fusion of modalities (e.g. in biometrics) or combination of experts. On the second hand, ensemble creation techniques try to generate a set of classifiers such that their combination will be performant. Such algorithms are usually iterative procedures that grow ensembles until the expected classification performances are reached.

Many studies focused on understanding why ensemble methods - even simple voting schemes - were so successfull in most of supervised classification tasks. One of the main factors that can explain this empirical observation is that, performance of complementary classifiers can be improved by aggregation, supposing that errors committed by one classifier can be corrected by the others in the team. On the contrary, combining classifier that commit errors on the same data seems to be useless. Consequently, it is commonly admitted that classifiers need to be diverse in order to improve performances compared to the best individual classifiers. In this sense, numerous diversity measures have been proposed to analyze the efficiency of classifier ensembles. However, in practice, using these diversity measures as criteria for creating good ensembles is not as successful as expected. How to efficiently use explicit measures of diversity in classifier ensembles is still an open problem in the pattern recognition community.

The work in this thesis mainly focuses on the role played by diversity in ensemble methods. In particular, we will propose an information theoretic framework for proving that diversity is, in general, not self-sufficient, but needs to be coupled carefully with the average individual accuracy of the classifiers in the team. As an illustration, we will propose a complete analysis of one particular classifier combination technique: combination of Support Vector Machines. Support Vector Machines are one of the most popular discriminative approach of pattern recognition and is often considered as state-of-the-art in binary classification. However they present one severe drawback when facing a very large number of training examples. These classifiers become very computationally expensive to

train, mainly because of the model selection step that becomes quickly intractable. This problem can be addressed by decomposing the learning of the Support Vector Machine into several tasks with lower computational requirements. One possible solution is to split the training data into several subsets, and train in a parallel manner one Support Vector Machine on each subset. Not only does it reduce the complexity of the learning problem but, in practice, it also provides improvements in terms of classification accuracy. In fact, the implementation in a parallel structure can reduce influence of potential outliers or noise present in the training data. In this work we will propose several strategies for designing performant ensembles of Support Vector Machines, in particular based on our information theoretic framework.

1.1 Organization of the Thesis

This thesis is organized in two main parts. We will first give theoretical aspects of pattern recognition and classifier combination, in particular we will propose an information theoretic framework for combining classifiers and detail concrete example considering ensembles of support vector machines. Then, we will propose to use classifier combination techniques in a real-world application, namely frontal face detection.

Chapter 2 gives an introduction to pattern recognition from a discriminative perspective and review the main classifiers used throughout the thesis: Support Vector Machines, decision trees, K-nearest neighbors, etc.

Then, chapter 3 gives a general overview of classifier combination techniques. A greedy ensemble creation technique called AdaBoost is detailed, since it will be used extensively in the face detection application.

Chapter 4 introduces the information theoretic framework for combining classifiers. We propose the so called Information Theoretic Score that measures the efficiency of an ensemble by fixing a trade-off between diversity and individual accuracies. As an illustrative use of this score, several combination techniques will be proposed.

Chapter 5 introduces new techniques for combining efficiently Support Vector Machines.

The second part of the thesis will be divided into three main chapters. First, a general overview of existing face detection techniques is given in chapter 6.

Chapter 7 introduces new geometrical features that can be used to model efficiently face appearance, while chapter 8 shows how classifier combination techniques can be implemented successfully in a frontal face detection system.

Finally, chapter 9 will conclude by giving a short summary of the work and give the main perspectives.

1.2 Main Contributions

This thesis mainly targets the study of classifier combination and proposes, as a case study, the application to face detection. We study the concept of diversity in several classifier combination schemes. In particular, we propose to analyze the classifier combination problem in

an information theoretic framework and then propose solutions for tackling the limitations of pure diversity-based classifier combination techniques. We show how the new information theoretic framework can be used in the particular case of combination of Support Vector Machines. Finally we give an illustration of classifier combination techniques in a complete real-world system, namely face detection. Here is a short list of the main contributions of the present work:

- We propose an information theoretic framework to classifier combination. We will give theoretical insights justifying why an ensemble can outperform single classifier schemes. In the particular case of majority voting for combining the decisions, we show that the main challenge of optimal classifier combination is to find a trade-off between diversity in the ensemble and average individual accuracy.

- From empirical considerations we establish a link between these two contradictory quantities. We propose a new measure of the efficiency of an ensemble, the Information Theoretic Score (*ITS*).

- The new score is used in the context of over-production and selection of classifiers and evaluated on several common datasets.

- We then show how this framework can be extended to weighted majority voting through an example algorithm. We propose a modification of AdaBoost that takes into account the information theoretic score mentioned hereabove. AdaBoost is known for implicitly creating diversity between the hypotheses that are selected. We show that incorporating an explicit measure of diversity (through our new score) gives an algorithm that compares favorably with the standard version of AdaBoost.

- We then consider a particular classifier combination application: combination of several Support Vector Machines. This technique can be used to efficiently reduce the training complexity but we also show how it can improve classification performances by taking advantage of the classifier combination paradigm. In particular, we propose to use an on-line algorithm for training parallel Support Vector Machines jointly by taking into account the information theoretic criterion.

- Concerning the face detection application, we first present new geometrical features that efficiently model the face appearance. These new features proved to be more discriminative than the state-of-the-art features, while still being computationally efficient.

- Finally, we propose to use a classifier combination technique in the face detection application for improving both training time and classification performances. We show on standard frontal face datasets that our system compares favorably to state-of-the-art algorithms.

Part I

Theoretical Developments

Statistical Pattern Recognition

2

2.1 Introduction

The Machine Learning field aims at developping algorithms that are able to learn by experience. Learning by heart is a simple task from a mathematical perspective as it only involves memory considerations. However, machine learning refers to a more challenging task, that is learning a model from available information, such that the model will generalize to new situations. Human brain has very powerful learning capabilities, and many studies have concentrated on understanding the underlying biological phenomenon. This field of study is situated at a crossroad between various research communities such as probability and statistics, engineering, computer science, biology, medical research, signal processing, adaptive control theory, etc. In the last decades machine learning covered a wide range of applications, including automatic character recognition, speech recognition, but also medical diagnosis, data mining, and more recently biometrics. The large scientific research activity in this field is reflected by the publication of many books on machine learning [8, 32, 48, 57, 75, 146, 164, 167].

Pattern recognition (or classification) is the field of applications that aims at learning from a set of examples how to classify new data into a finite set of categories that are called classes. The input of a common pattern recognition system is thus an entity called *pattern* (e.g. an image, an audio signal, etc.) which we aim to associate to an output class.

Here are a few examples of typical pattern recognition tasks:

- Predict from various medical and demographic measurements if a patient presents risks of developing lung, prostate or breast cancers;
- Recognize automatically handwritten digits for automatically reading ZIP codes on postal envelopes;

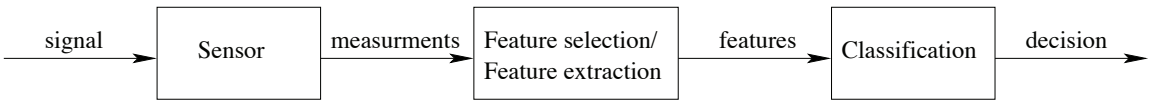


Figure 2.1: Different stages of pattern recognition systems.

- Recognize fingerprints or faces for biometric verification purposes;
- Develop an efficient SPAM filter.

A pattern recognition system is ususally decomposed into three main steps: data acquisition, feature selection and extraction and finally classification. A pattern is represented by a set of measurements that should contain relevant information with respect to the structure of the object that we want to classify. The measurements can potentially be collected from a large number of sensors, thus resulting in a high dimensionality vector of measurements. The first steps in a pattern recognition system consists in processing the raw data coming from the measurements, in order to find an adequate representation of the signals. This pre-processing of the data is called feature selection and extraction. The new data called features are then given to a classifier that takes the decision. An overall view of the main stages of a pattern recognition system is shown in figure 2.1. The training examples (or training patterns) are the known instances from which we want to learn a model that can generalize to previously unseen data.

Pattern Recognition can basically be split into two categories: supervised learning and unsupervised learning.

- In supervised learning, the classes in which we want to classify the data are known a priori. Each training pattern is associated to one of these known classes (one the the 10 digits, cancer or not, SPAM or not). An example of supervised learning is face recognition. Consider the problem of recognizing the identity of of person based an image of its face. The training examples are face images from which we know the associated identities. The set of classes then corresponds to the list of known identities that are present in the training set. A robust system should be able to recognize identities of new face images for identities belonging the set of classes.
- In unsupervised learning, the true classes of the training examples is not known a priori. We seek to find the various groups of pattern present in the set of training data. The task is thus to investigate the underlying structure of the data. Clustering techniques are often used to extract the groups in the training dataset. For example, given a collection of text documents, we want to organize them according to their content similarities.

Most of the work in this thesis concentrates on supervised learning.

From a mathematical perspective, pattern recognition can be seen as a decision making process that can be summarized as follows: A given pattern $\mathbf{x}$ is to be assigned to one of C classes $w_i, w_2, \ldots, w_C$ based on a vector of d measurements called features. A pattern $\mathbf{x}$

belonging to class w_i is viewed as an observation drawn randomly from the class-conditional probability function $p(\mathbf{x}|w_i)$. The optimal assignment for unknown examples $\mathbf{x}$ is the class that maximizes the posterior probability $p(w_i|\mathbf{x})\ \forall i \in 1, \ldots, C$. This principle is known as the maximum a posteriori criterion (MAP).

There are basically two different approaches for implementing the MAP principle. The first technique consists in finding directly decision functions between the classes in order to find the class that is more likely to generate the test example. This technique is known as discriminative approach. Most of the work in this thesis focus on this decision-driven strategy, that is why we will give more details in the next sections of this chapter.

The other technique tries to explicitly estimate the underlying class-conditional probability densities. It is based on Bayes decision rule [157]:

$$p(w|\mathbf{x}) = \frac{p(\mathbf{x}|w).p(w)}{p(\mathbf{x})}. \tag{2.1}$$

It mainly shows that posterior probabilities $p(w|\mathbf{x})$ can be expressed as a function of the class conditional densities $p(\mathbf{x}|w)$ and the class priors $p(w)$. In practice these class conditional densities are unknown and various techniques have been proposed to estimate these quantities from the training data. These techniques are called generative methods. The simplest generative method is to consider the class conditional densities to be Gaussian distributed and to estimate directly the Gaussian parameters from the training data. See [8] for a detailed overview of recent generative methods (e.g. Bayesian networks).

In the remaining of the chapter we will present the main concepts of statistical pattern recognition from a discriminative perspective.

Section 2.2 will give an overview of the main theoretical concepts of the discriminative approach to pattern recognition. Then section 2.3 gives a first example of classifier, namely Support Vector Machines. In section 2.4, we present other decision functions that will also be used throughout this thesis. Section 2.5 reviews the main principles of feature selection and extraction. Finally, section 2.6 gives more practical considerations about model selection and evaluation of the algorithms.

2.2 Theoretical Concepts of Discriminative Pattern Recognition

One possible formalism of the pattern recognition task is based in statistical learning theory proposed by Vapnik [164]. In this section, we give the theoretical concepts in broad lines. The common pattern recognition task is when the output space only contains two classes. The classification problem is then a binary classification task. Nevertheless, a multi-class problem can easily be decomposed into several coupled binary problems, that is why we will only consider the binary case if not specified otherwise. Note that several applications need to cope with a very large number of classes (e.g. speech recognition where each word represents one single class) but such cases a re out of the scope of this thesis.

Let us consider each pattern to be represented by a vector $\mathbf{x} \in \mathbb{R}^d$ of d features. Each

example to be classified belongs to one of the two classes represented by the labels $y = -1$ or $y = 1$.

Let us consider Z_n the set of n training patterns in the space $Z = \mathbb{R}^d \times \{-1, 1\}$:

$$Z_n = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}, \quad \text{with} \quad \mathbf{z}_i = (\mathbf{x}_i, y_i) \in Z = \mathbb{R}^d \times \{-1, +1\}. \tag{2.2}$$

Each sample $\mathbf{z}_i$ is assumed to be generated from an unknown (but fixed) probability distribution function (*pdf*): $P(\mathbf{x}, y)$. For each of the training examples $\mathbf{x}_i$, the true class label y_i is known (supervised learning).

2.2.1 Expected Risk Minimization

The problem of learning may be expressed as an optimization problem in which one wants to find the function f^* from a suitably chosen set of functions $\mathcal{F}$, which minimizes the risk of misclassifying new vectors drawn from the same *pdf* P. An example $\mathbf{x}$ is assigned to class $+1$ if $f^*(\mathbf{x}) \geq 0$ and to the class -1 otherwise. The best function can be obtained by minimizing the so-called expected risk R on $\mathcal{F}$ [164]:

$$R(f) = E_Z[\mathcal{L}(y, f(\mathbf{x}))] = \int_Z \mathcal{L}(y, f(\mathbf{x}))dP(\mathbf{x}, y), \tag{2.3}$$

where $\mathcal{L}$ is called loss functional. The loss functional penalizes the differences between the true labels y and the predicted ones $f(\mathbf{x})$. The most commonly used loss is the binary loss (or 0/1-loss):

$$\mathcal{L}(f(\mathbf{x}), y) = \begin{cases} 0, & \text{if} \quad f(\mathbf{x}) = y \\ 1, & \text{otherwise} \end{cases}. \tag{2.4}$$

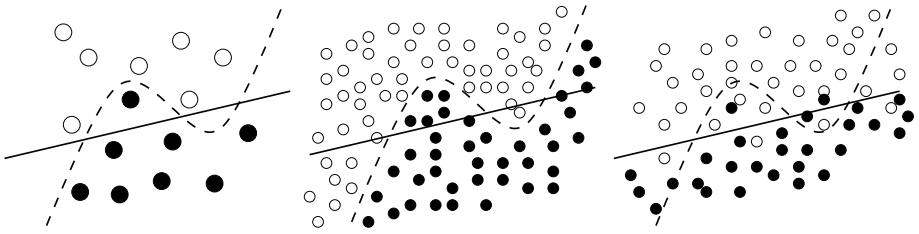
It simply counts the number of misclassified examples. Other loss functionals take into account the confidence that we may have for the prediction of each example. For example, the squared loss function is widely used in regression applications: $L(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2$. The logistic loss function is used to give a probabilistic interpretation of the output: $L(f(\mathbf{x}), y) = \log(1 + \exp(-f(\mathbf{x})y))$. More specific loss functionals will be given in section 3.5.

The optimal function f^* should be the one that generalizes best from training examples to new data. That is why the amount of new data that is misclassified by f^* is called generalization error.

Unfortunately, the risk $R(f)$ cannot be minimized directly, since the underlying distribution $P(\mathbf{x}, y)$ is unknown. Therefore, we need to find an estimate of f^* obtained from the available information: the training set. A simple technique called induction principle consists in only computing R (equation (2.3)) on the available data. The risk is then called the empirical risk:

$$R_{emp}(f, Z_n) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(\mathbf{x_i}), y_i). \tag{2.5}$$

It can be shown that the empirical risk is an unbiased estimate of the expected risk:



(a) Small training set (b) Solid line underfits (c) Dashed line overfits

Figure 2.2: Illustration of the overfitting and underfitting phenomena.

$E[R_{emp}(f, Z_n)] = R(f)$ (see [164]). The empirical risk minimization consists thus in selecting the function: $f^*_{Z_n} = \arg\min_{f \in \mathcal{F}} R_{emp}(f, Z_n)$. The proportion of training samples that are misclassified by the obtained function is called training error.

2.2.2 Capacity

Empirical risk minimization seems straightforward. However, it presents a major limitation known as overfitting. In order to understand the problem from a theoretical perspective, we will introduce the notion of capacity of a class of functions.

A very important step in the design of pattern recognition systems is the choice of the set of function $\mathcal{F}$ in which we seek the optimal solution.

Let us consider a simple 2-dimensional problem as depicted in figure 2.2. Consider that the training set is represented by the examples in figure 2.2(a). Given only this small sample set, either the solid or the dashed hypothesis might be true, the dashed one being more complex, but also having a smaller training error.

Only when more data are available we can judge which decision function reflects best the true distribution. Let us imagine that the true distribution is in fact the one depicted in figure 2.2(b). In this case, solid line is too simple to explain the data, while the dashed line seems to generalize well. We say that the solid line underfits the data. If the true distribution is the one represented in figure 2.2(c), then the dashed line is too complex, it overfits the data. In this last case, we see that a function that minimizes the training error (dashed line in figure 2.2(a)) is not a guarantee of small generalization error.

A possible solution for avoiding overfitting is to restrict the complexity of the function class $\mathcal{F}$. From a practical perspective, the main idea is that simple function that explains most of the data is preferable to a complex one. (this follows Occam's razor principle).

More theoretically, Vapnik et al [164] showed that the solution $f^*_{Z_n}$ found by minimizing the empirical risk is better on the given training set Z_n than on any other set drawn from $p(\mathbf{x}, y)$.

In fact, the complexity of the class of functions $\mathcal{F}$ plays a key role in this problem. The *capacity* of a set $\mathcal{F}$ gives a measure of this complexity:

Definition 2.1. *The capacity is the largest n such that there exits a set of example Z_n such that one can always find a function $f \in \mathcal{F}$ which gives the correct classification for all the*

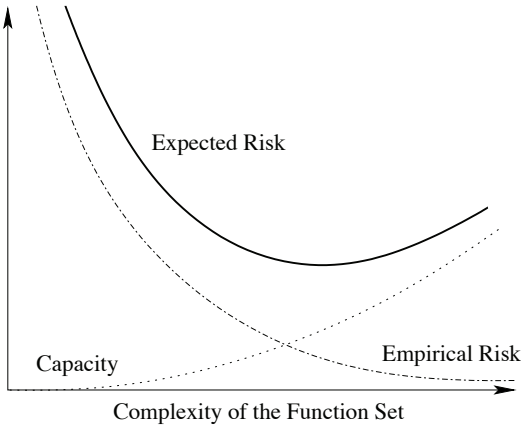


Figure 2.3: Illustration of the dilemma between empirical risk minimization and confidence of the function set (capacity).

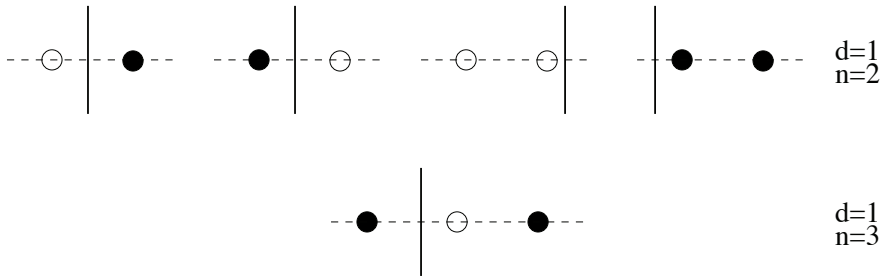


Figure 2.4: Illustration of the *VC* dimension in a 1-dimensional feature space. 2 points can be separated by an hyperplane with any labellings while 3 examples cannot. In this case, VC =2.

examples in Z_n, for any possible labeling

In practice, controlling the capacity of decision functions is achieved by adding a regularization term in equation (2.5) (e.g [119, 160]).

Finding an optimal set of functions and regularization term can be viewed as a model selection step. This step is on of the most essential in order to obtain a robust decision function.

Vapnik et al [164] proposed a theory for controlling the capacity of a set of function. They define the Vapnik-Chervonenkis (VC) dimension:

Definition 2.2. *Vapnik-Chervonenkis (VC) dimension of a class of function $\mathcal{F}$ is the number of points that can be separated for all possible labeling and using all the functions of the class $\mathcal{F}$.*

This VC dimension is not related to the number of parameters of the set of function but really gives a measure of the complexity. For example simple class of functions with one single parameter can produce infinite VC.

The main interest of VC is that it gives several formulations of upper-bounds on the expected risk. For example, the following theorem is proved in [164].

Theorem 2.1 (Vapnik et al [164]). *Considering the binary loss defined in equation* (2.4)*, for all* $\delta > 0$ *and* $f \in F$*, the inequality bounding the risk:*

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{1}{n}\left(VC(\log\frac{2n}{VC} - \log\frac{\delta}{4})\right)}, \tag{2.6}$$

holds with probability of at least $1 - \delta$ *for* $n > VC$*.*

From the bound in equation (2.6) we can distinguish two extremes:

- If the class of function $\mathcal{F}$ is very simple, then the squared root term vanishes but a large training error might occur.
- If $\mathcal{F}$ contains very complex functions, then the empirical error may be small but the regularization term may become large.

The best class of functions $\mathcal{F}$ is usually lies in between. Figure 2.3 gives an illustration of the trade-off between the empirical risk and the regularization term.

In fact the expected risk minimization faces a typical bias/variance dilemma. The bias comes from the choice of the set of function $\mathcal{F}$. There is a bias when the optimal solution is not included into $\mathcal{F}$. The variance is due to the fact that using another training set Z'_n would give a different solution.

2.3 Large Margin Classifiers

In section section 2.2 we introduced the main theoretical considerations of statistical pattern recognition. Basically the goal if to find the optimal function f^* in a class of functions $\mathcal{F}$, that minimizes the expected risk equation (2.3). In the following sections we will present several examples of decision functions that will be used widely in this thesis.

2.3.1 Introduction to Linear Machines

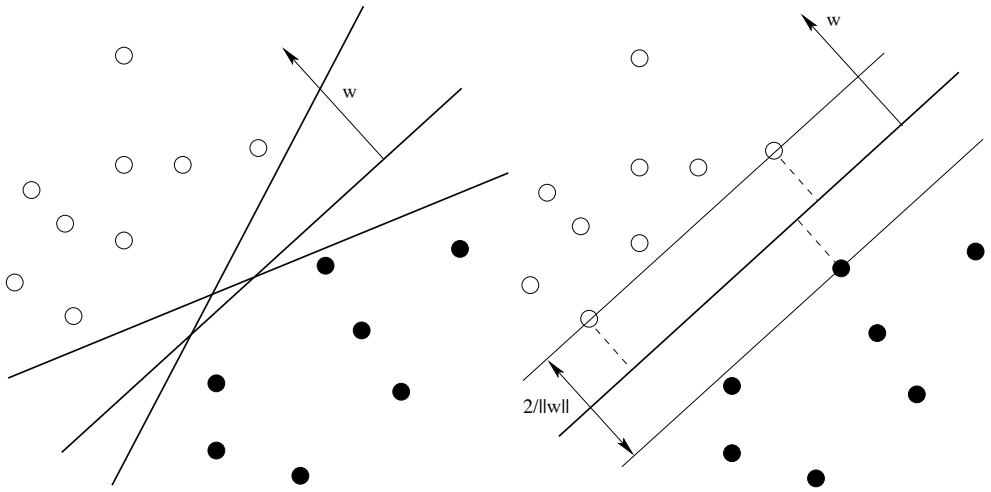
A first simple choice of function class $\mathcal{F}$ is to use simple hyperplanes for separating the two-class data. They are called linear discriminant functions, linear machines or linear classifiers. Let us consider a set of functions of the form*:

$$f_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{2.7}$$

where $\mathbf{w}$ and b are the parameters of the linear function. The set of points where $f_{\mathbf{w},b}(\mathbf{x}) = 0$ is called the hyperplane and the decision function is given by:

$$f(\mathbf{x}) = \mathrm{sgn}(f_{\mathbf{w},b}(\mathbf{x})) = \mathrm{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right). \tag{2.8}$$

*We use $\langle \cdot, \cdot \rangle$ to denote the inner product operator



(a) Possible Perceptron solutions (b) Hyperplane with maximal margin

Figure 2.5: Linear decision functions for linearly separable data. (a) Solutions for Perceptron. (b) The separating hyperplane that separates the data while maximizing the margin between the two classes following the idea of Support Vector Machines.

The optimization process of the learning is to find the best function parameters $\mathbf{w}$ and b. There are various motivations for using simple hyperplanes as decision functions. On the first hand it perfectly fits with Occam's Razor principle cited in a previous section. In fact, it can be shown that for the class of hyperplanes, the *VC* dimension can be bounded efficiently. More details can be found in [164].

The papameters $\mathbf{w}$ and b can be optimized such that the number of misclassified examples is minimized. This is the principle of Rosenblatt's perceptron [135]. It considers an optimization criterion which is directly proportional to the sum of the distances of misclassified examples to the decision boundary. A set of data is linearly separable if there exists a constant γ such that $y_i\langle \mathbf{w}, \mathbf{x}_i\rangle + b > \gamma$ for all i. The limitations of this simple mistake-driven procedure is that its convergence is only guaranteed for linearly separable data. Novikoff [107] proved that the perceptron algorithm converges after a finite number of iterations if the data set is linearly separable and the number of mistakes is bounded by $\left(\frac{2r}{\gamma}\right)^2$, where $r = max_i \parallel \mathbf{x}_i \parallel$. Moreover, the perceptron algorithm stops once the training data has been correctly separated and this induces that the solution of the perceptron is not unique. This non unicity is shown in figure 2.5(a). All the candidate linear functions drawn separate perfectly the training data. However, we have no information about which one will generalize best.

Fisher introduced another criterion for separating the data but trying to separate them as much as possible (e.g [48]). This criterion is the ratio of the between-class to within-class variances. This criterion aims at finding the best separation between the classes by taking into account their respective variances. However, it is only optimal under the assumption of two normal distributions.

2.3.2 Maximization of the Margin

A popular quantity that measures how much a linear hyperplane separates the data is the *margin*. The margin is defined as the minimal distance of a sample to the decision surface. Vapnik [164] showed that the *VC* dimension can be bounded in terms of the margin.

Linearly Separable Case

Let us first assume that the two classes are linearly separable (the non separable case will be discussed later on). The parameters $\mathbf{w}$ and b in equation (2.8) can be rescaled such that the closest points to the hyperplane satisfy $|\langle \mathbf{w}, \mathbf{x} \rangle + b| = 1$. This normalization gives the so-called *canonical representation* of the hyperplane. Now let us call $\mathbf{x}_1$ and $\mathbf{x}_{-1}$ two samples of each class satisfying $\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = 1$ and $\langle \mathbf{w}, \mathbf{x}_{-1} \rangle + b = -1$. The margin ρ is given by the distance between these two points, measured perpendicular to the hyperplane:

$$\rho = \frac{\mathbf{w}}{\|\mathbf{w}\|}.(\mathbf{x}_1 - \mathbf{x}_{-1}) = \frac{2}{\|\mathbf{w}\|}. \tag{2.9}$$

The hyperplane perpendicular to the margin is depicted in figure 2.5(b).

From this definition, the bound linking *VC* and the length of the weight vector $\mathbf{w}$ is:

$$VC \leq \Lambda^2 R^2 + 1, \tag{2.10}$$

where $\Lambda \in \mathbb{R}^+$ is a constant such that $\|\mathbf{w}\| < \Lambda$ and R is the smallest ball around the data: $\|\mathbf{x}\| < R$. As we cannot directly minimize the expected risk defined in equation (2.3), we try to minimize two terms: the empirical risk in equation (2.5) and the complexity term in equation (2.6). The empirical risk is forced to be zero as we constrain b and $\mathbf{w}$ to linearly separate the data, and the capacity is controlled by minimizing the bound in equation (2.10). It turns out that minimizing this bound means minimizing $\|\mathbf{w}\|^2$.

For a fixed training set Z_n, finding the optimal hyperplane (optimal in the sense of largest margin) is thus equivalent to solving a quadratic optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \tag{2.11}$$

subject to:

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1, \quad i = 1, \ldots, n. \tag{2.12}$$

A standard approach to optimization problems with equality and inequality constraints is the Lagrange formalism [41]. The dual optimization problem is obtained by introducing the Lagrange multipliers $\alpha_i > 0, \quad i = 1, \ldots, n$, (one for each constraint in equation (2.12)). We obtain the following Lagrangian function:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b - 1 \right). \tag{2.13}$$

This function has to be minimized with respect to $\mathbf{w}, b$ and maximized with respect to α. The optimal saddle point equation is found at: $\frac{\partial L}{\mathbf{w}} = 0$ and $\frac{\partial L}{b} = 0$. We obtain the following dual optimization problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \tag{2.14}$$

subject to:

$$\alpha_i \geq 0, \quad i = 1, \ldots, n, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.15}$$

By solving this dual optimization problem, we obtain a linear decision function that only depends on dot products between training patterns:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle \right). \tag{2.16}$$

We have up to now only considered the case of separable training data which corresponds to training error equal to zero. However, this might not be the optimal choice that minimizes the expected risk. This is particularly true in case of noisy data. Directly minimizing the empirical risk introduces a large risk of overfitting.

Non Linearly Separable Case

In order to find a better trade-off between empirical risk and capacity, a common strategy is to allow some training points to fall into the margin. This is done by introducing the so-called slack variables ξ_i that will relax the constraints defined in equation (2.12):

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad i = 1, \ldots, n. \tag{2.17}$$

We then add a constant $C > 0$ that penalizes training patterns that fall into the margin. This C controls the trade-off between empirical risk and capacity. It needs to be tuned through a model selection process. The new optimization problem:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i, \tag{2.18}$$

which can be turn into another dual problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \tag{2.19}$$

subject to:

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.20}$$

This formulation is known as a C-Support Vector Machine (C-SVM). Let us now analyze the decision function equation (2.16) obtained by the dual optimization problem.

A first interesting property of the *SVM* is given by the so-called Karush-Kuhn-Tucker conditions [163]. They show that for each training pattern:

$$\begin{cases} \alpha_i = 0 & \Rightarrow \quad y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} \quad \xi_i = 0 \\ 0 < \alpha_i < C & \Rightarrow \quad y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0 \\ \alpha_i = C & \Rightarrow \quad y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} \quad \xi_i \geq 0 \end{cases} \tag{2.21}$$

These conditions mean that the solution equation (2.16) is sparse in α. In other words, most of the training samples are outside the margin and their corresponding α_i are zero. This sparsity is one of the most attractive property of the *SVM* since the number of training examples the model depends on is small comparing to the size of the training set. These models are thus appealing as they have low storage requirements and usually lead to fast decision functions.

The main drawback of *SVM* is that finding the support vectors and their coefficients α requires solving a quadratic optimization problem. This becomes intractable in very large scale problems. Several on-line algorithms have been proposed to find iteratively the support vectors. The two well known techniques are Kernel Adatron (*KA*) [148] and Sequential Minimal Optimization (*SMO*) [117].

2.3.3 Non Linear Classifiers in Kernel Feature Spaces

We have seen that simple decision functions are preferred in order to ovoid the problem of overfitting. However, linear decision functions are not very realistic in most applications as they present the tendency to underfit the training data. There exists various algorithms for building more complex non linear decision functions. One possibility is to use Neural Networks. An Artificial Neural Network (*ANN*) is an information processing paradigm that is inspired by the way biological nervous systems (human brain) process information. The non-linear decision function is obtained by merging information from a structured collection of simple decision units called neurons. See [65] for a complete review on *ANN*. When these decision unit are simple linear perceptrons, the algorithm is called Multi-Layer Perceptron (*MLP*). The weights of each perceptron are learn iteratively using feedforward or backpropagation strategies.

Another way of extending the linear decisions to non linear functions is given by the notion of kernel [1, 148, 164]. We noticed hereabove that most of real data cannot be linearly separated in the original feature space. The idea is to find a non linear mapping Φ of the data such that the examples are projected onto a potentially much higher dimensional feature space, where the data can more easily be linearly separated. The motivation of projecting data into a high dimensional feature space is that simple hyperplanes can be sufficient for separating the new data. This phenomenon is represented graphically in figure 2.6. Consequently, the mapping allows to apply the principles established of linear separable data, but in the new complex feature space.

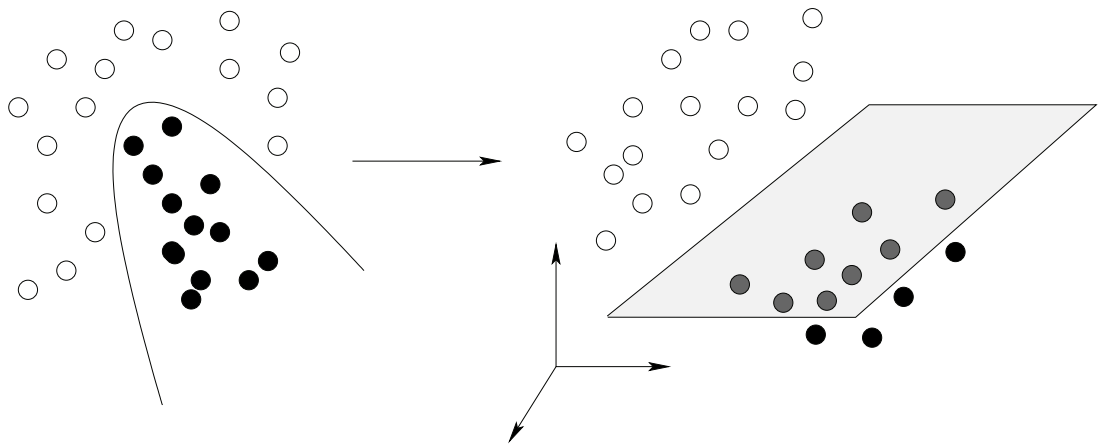


Figure 2.6: Mapping Φ that projects 2D data onto a 3D space where the data becomes linearly separable.

Let us define the non linear mapping from $\mathbb{R}^d$ onto a high dimensionality Hilbert space $\mathcal{S}$:

$$\begin{aligned} \Phi : \mathbb{R}^d &\rightarrow \mathcal{S} \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}). \end{aligned} \tag{2.22}$$

Given this mapping Φ, all the linear techniques presented in the previous sections can be applied in the new feature space $\mathcal{S}$ simply by replacing the data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ by the projected data $\{(\Phi(\mathbf{x}_1), y_1), (\Phi(\mathbf{x}_2), y_2), \ldots, (\Phi(\mathbf{x}_n), y_n)\}$.

The new problem is then how to choose the mapping Φ. We notice that only dot products between examples are involved in the decision function given in equation (2.16). The trick is that we do not really need to explicitly know the mapping Φ, the only information we need from the feature space $\mathcal{S}$ is how to compute dot products in that space. A very efficient trick for computing dot products in feature spaces is to use the so-called *kernel functions* [104]. A kernel function is defined as:

$$\begin{aligned} k : \mathbb{R}^d \times \mathbb{R}^d &\rightarrow \mathbb{R} \\ \mathbf{x}_1, \mathbf{x}_2 &\mapsto k(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1)\Phi(\mathbf{x}_2). \end{aligned} \tag{2.23}$$

A kernel function needs to fulfill several properties known as Mercers's conditions [99]:

Theorem 2.2 (Mercer)**.** *There exists a mapping Φ and an expansion k if and only if for any function g such that $\int g(x)^2 dx$ is finite, then $\int k(\mathbf{x}_1, \mathbf{x}_2) g(\mathbf{x}_1) g(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2 \geq 0$.*

The two most used kernel function are Polynomial kernel:

$$\text{Polynomial kernel} \qquad K(\mathbf{x}_1, \mathbf{x}_2) = (\langle \mathbf{x}_1, \mathbf{x}_2 \rangle + 1)^p \tag{2.24}$$

and Radial Basis Functions (RBF):

$$\text{RBF kernel} \qquad K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2). \tag{2.25}$$

Support Vector Machines can easily integrate this kernel trick in order to find large margin hyperplanes in the new high dimensional feature space. The decision function equation (2.16) becomes:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i)\right). \tag{2.26}$$

Using the kernel trick, we can obtain complex non linear decision functions. Once again, the capacity of the classifier needs to be controlled. In this case the capacity is controlled by the kernel coefficients. For example the variance parameter σ of the RBF kernel will determine the degree of non linearity of the decision function. A small σ will produce a complex decision function very likely to overfit while a larger σ will produce a smoother function. Concerning the polynomial kernel, the degree of the polynomial p will have the same effect.

2.4 Other Examples of Non-linear Decision Functions

In this section we present other common decision functions that will be used throughout this thesis. They differ from section 2.3.2 in the sense that their non linearity is intrinsic and is not caused by any extension from linear case, like for example, the kernel trick presented in the previous section.

2.4.1 K-Nearest Neighbors

We first present a very simple classifier that does not require a training process: K-Neareast Neighbors ($k-NN$). Let us assume that a distance function d is associated to the input feature space (e.g. the well-known l-2 norm: $d(\mathbf{x}_1, \mathbf{x}_2) = \| \mathbf{x}_1 - \mathbf{x}_2 \|_2$). For a given parameter $K \geq 1$ the decision function is built as follows: For any test example $\mathbf{x}$, select the K nearest training patterns according to $d(\mathbf{x}, \mathbf{x}_i)$ and keep their indices $s_1, s_2, \ldots, s_K$. In the 2-class case, the decision function is then given by:

$$f(\mathbf{x}) = \text{sign}\left(\frac{1}{K}\sum_{k=1}^{K} y_{s_k}\right). \tag{2.27}$$

There are basically two quantities to be chosen by the user: the distance metric d and the number of neighbors K. In fact the value of K controls the capacity. A small K will produce a complex decision function very likely to overfit and a too large K will tend to underfit the training data.

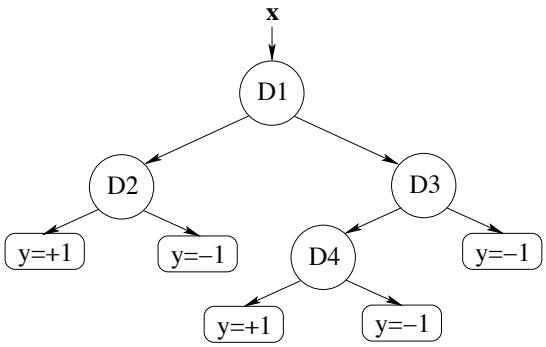


Figure 2.7: A simple decision tree with 4 decision nodes.

2.4.2 Decision Trees

A decision tree [13, 124], in its simplest form, produces decision boundaries that are parallel to the features axes. It is generally represented by a graph model containing a root node, several branches and leafs. A simple graphical example is given in figure 2.7. The root node takes as input the pattern $\mathbf{x}$ and each leaf corresponds to one of the C classes. In the simplest form of decision trees, the nodes are very basic classifiers that only consider one single input feature. Each node compares the feature value to a predefined threshold. The training process consists in subdividing iteratively the feature space until all the training patterns in a node are from the same class. The simplest decisions trees only have one split at the root node. They are called decision stumps. Then several tree growing techniques have been proposed, like the CART [13] or ID3 [124]. In order to avoid overfitting, the first solution is to stop the construction of the tree before the end. The other possibility is to grow completely the tree and then prune it.

2.5 Feature Selection and Extraction

2.5.1 Curse of Dimensionality

The generalization performances of a classifier depend on the interrelationship between the number of training samples n , the number of features d, and the complexity of the decision function. Imagine that we partition the feature space into hypercubes such that we associate a class label y to each cell. This label corresponds to the class having the largest number of training patterns in the cell. The problem is that the number of hypercubes grows exponentially with the dimensionality of the feature space d. The consequence of this is that the number of training samples required also becomes an exponential function of the feature dimension d [8]. This phenomenon is termed as *curse of dimensionality* [128]. An illustration of this problem is shown in figure 2.8. The same number of examples fill more of the available space when the dimensionality is low. In practice, curse of dimensionality means that, for a given sample size, there is a maximum number of features above which the performance of the classifier will degrade rather than improve.

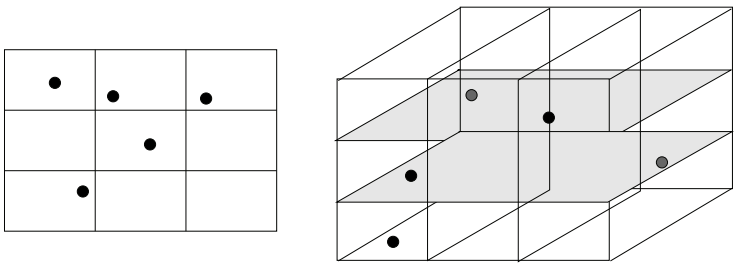


Figure 2.8: Curse of dimensionality. 5 examples fill more space in a 2-dimensional feature space than in 3 dimensions.

In order to avoid this curse of dimensionality, a wise technique is to restrict the set of features to only a small number of discriminant features. In practice, it is generally accepted that using at least ten times as many training samples per class as the number of features. (i.e. $\frac{n}{d} \geq 10$) is a good design choice [66]. The reduction of the number of features can be made into two steps: feature selection and feature extraction.

2.5.2 Feature Selection

Feature selection consists in keeping the most relevant features and discarding irrelevant or redundant features. This feature selection step is critical in applications where a huge number of features is encounterred. Complete reviews can be found in [67, 74, 103]. From a set of d features we want to find the optimal subset of d^* features (with $d^* \leq d$) such that the classification error is minimized. Finding the best subset requires the optimization of some criterion J. A natural choice is $J = 1 - P_e$ where P_e is an estimate of the classification error. A straightforward approach is to test all the possible subsets from the original feature set and keep the subset that minimizes J. This exhaustive search becomes intractable if d is not very small.

The only optimal and non exhaustive feature selection method is called the branch and bound algorithm [106]. It uses intermediate results in order to derive bounds on the optimal criterion value. However, in most applications this approach remains too computationally expensive, that is why most of the techniques use suboptimal sequential selection. Sequential Forward Selection (SFS) selects the best features, one at a time, while Sequential Backward Selection (SBS) deletes iteratively the poorest features. A complete comparative study of most of these techniques can be found in [28].

2.5.3 Feature Extraction

The purpose of feature extraction is to find better representations for a given set of features by applying either linear or non-linear transformations on the data. Feature extraction is equivalent to projecting the data onto a feature space where the separation of the classes becomes simpler. The most commonly used linear transformations include Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

PCA was first introduced in 1933 [62] for decorrelating the input data. It can simply be viewed as a rotation of the feature space along the main axis of the data called principal components. The main reason for performing *PCA* is to find a smaller group of underlying variables that describe the data. *LDA* is similar to *PCA* except that it takes into account a discriminative criterion for projecting the data. The principle is to find a projection that minimizes the within class variance (S_W) while maximizing the between class variance (S_B).

Then there are several non-linear feature extraction techniques. The most used is Kernel *PCA* [58]. It is an extension of *PCA* that maps the input data into some new high dimensionality feature space (the notion of kernel will be more detailed later in this chapter).

2.6 Model Selection and Risk Estimation

In this chapter, we reviewed several well known classification techniques. In each technique, there is at least one hyperparameter to be tuned trough a model selection step: numbers of neighbors K for $k-NN$, number of hidden layers for MLP, C and the kernel parameters for SVM, etc. These parameters are often determinant for obtaining good generalization properties.

Two strategies can basically be considered for tuning this parameters, depending on the amount of available data. If there is a consequent amount of training data available, a simple validation process is sufficient. Otherwise, a cross-validation technique needs to be implemented.

2.6.1 Simple Validation

Let us index the predefined class function $\mathcal{F}$ by the hyperparameter θ (that can be multi-dimensional) representing various parameter configurations. The principle is to divide the available data Z_n into two disjoint sets: a training set Z_{train} and a validation set Z_{val}. Z_{train} is used for training the classifiers with all possible valued of θ. The performance of each θ is then estimated on Z_{val}. We keep the value θ^* giving the minimal risk on the validation set:

$$\theta^* = \arg\min_{\theta} \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \mathcal{L}(f_\theta^*(\mathbf{x_i}), y_i) \tag{2.28}$$

where f_θ^* is the function of $\mathcal{F}$ that minimizes the empirical risk on Z_{train}.

The final function that is finally returned is found by searching the optimal solution $f_{\theta^*}^*$ on the whole training set Z_n, where θ^* is the best parameter discussed before.

2.6.2 Cross-Validation

In many applications, there is not a sufficient amount of data for having reliable estimates by splitting into one training set and one validation set. The solution is then to use a K-fold cross-validation [77]. The principle is rather simple. The original training set Z_n is split into K disjoint subsets $Z_n = \{Z_1, Z_2, \ldots, Z_K\}$. Then for each value of the parameter θ the estimate of the generalization error is described by the following process: First train

a classifier on $\{Z_1, \ldots, Z_{K-1}\}$ and measure the errors on the remaining subset Z_K. Then continue the procedure on other combinations of the subsets: train on $\{Z_1, \ldots, Z_{K-2}, Z_K\}$ and measure the errors on Z_{K-1}, etc. Finally the estimated validation error is the average of the K rounds.

The choice of K will determine a trade-off between computation complexity and confidence in the estimation error. The particular case $K = n$ is called leave-one-out (1 single pattern is kept as test set at each round).

2.6.3 Estimation of the Risk

In some application, the task of the pattern recognition problem is to not only to return the best classifier but also to give an estimation of the expected risk. This estimation requires another splitting strategy additional to model selection.

If possible, a third separate test set Z_{test} is used for estimating the risk. Otherwise, another cross-validation procedure can take place. Depending on the amount of data, several scenarios are possible:

- Large datasets: Use three disjoint subsets for training, validation and estimation of the test error.
- Intermediate-sized datasets: Split the dataset into two: Z_{sel}, Z_{test}. Perform cross-validation on Z_{sel} for model selection and measure the error rate on Z_{test}.
- Small datasets: Perform a double cross-validation. One run for validation and another cross-validation into each subset for the error estimation.

2.6.4 Comparison of Classifiers

Finally, if the ultimate goal of the pattern recognition task is to find a good learning methods in opposition to designing the best classifier, the comparison of the classification errors is not sufficient. If possible, the methods should be trained on several datasets, and statistical tests should be used for showing significance of the results. A review of the possible goals of pattern recognition as well as the main statistical tests that are commonly used is given in [29].

2.7 Summary

In this chapter we briefly reviewed the main principles of the discriminative approach to pattern recognition. This review is far from being exhaustive as only techniques that are used in the remaining of the thesis have been presented. In particular, we presented the Support Vector Machines that are often considered as state-of-the-art of discriminative techniques. *SVM* will be extensively used in chapter 5 through a multiple classifier system. The other techniques presented, *k–NN* and tree classifiers will be used as baseline classifiers, mainly in chapter 4.

In the next chapter, we will review a very active field of pattern recognition: multiple classifier systems. We will discuss the motivations for combining classifiers and reviewing in broad lines the main techniques. We will pay particular attention to Boosting as it will be use extensively throughout the remaining of the thesis.

3 On Combining Classifiers

3.1 Introduction

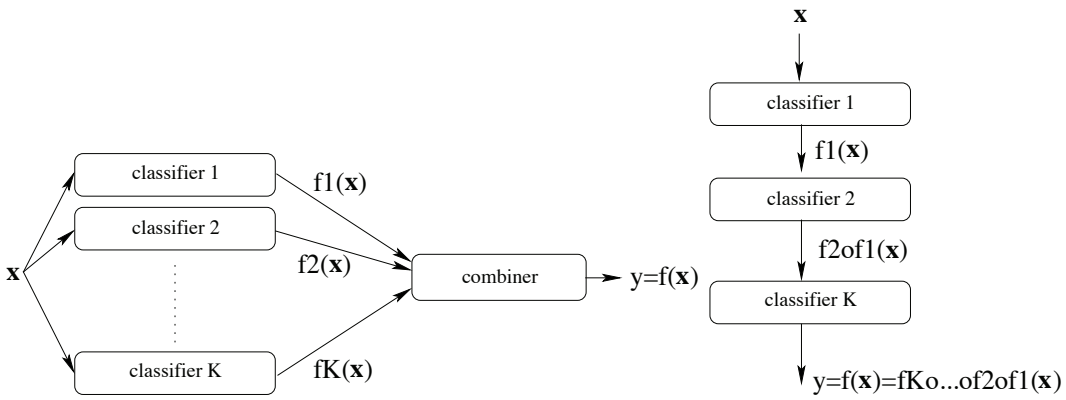
In this chapter, we will review the main classifier combination techniques. Combining classifiers is an established research area in the field of statistical pattern recognition to develop highly accurate systems. In most of the applications, there is no unique optimal learning algorithm. Moreover, the generalization properties of each algorithm are very sensitive to the model selection step. By combining several of these classifiers instead of keeping only one, we can hope increasing both reliability [88] and accuracy [30].

In the beginning of the 1990s, lots of studies consisted in splitting one complex classification task into several lower complexity tasks using the notion of local experts (e.g. [70, 71, 113]). In fact, it is generally much simpler finding several relatively good classifiers than finding one single very discriminant classifier. This chapter will describe the main situations in which combining classifiers can be applied successfully. Then, we will give an overview of the main classifier combination techniques, from simple majority voting to more complex ensemble creation methods. We will particularly describe the theoretical foundations of AdaBoost in section 3.5 as this ensemble creation technique will be employed widely throughout the remaining of the thesis.

A complete overview of classifier combination techniques can be found in Kuncheva's book [82]. Many different terms have been used to describe classifier combination techniques, the most common being: combination of multiple classifiers [53, 76, 82, 88, 172], mixture of experts [3, 20, 64], consensus aggregation [6], classifier ensembles [30, 85, 170].

A possible general definition of classifier ensemble is: *An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples* [30].

There are several possible strategies for combining classifiers. We can basically distin-



(a) Parallel structure (b) Hierarchical structure

Figure 3.1: Two possible structures for combining classifiers.

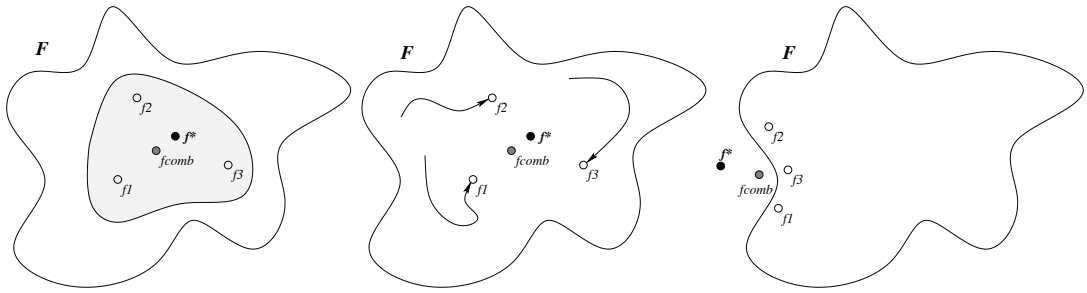
guish the combination at the decision level and the combination from a design perspective. Concerning the combination at the decision level, several classifiers are collected independently and the purpose is to find the best consensus decision. In this case the role of the combination is to centralize information coming from different sources. Some examples of decision level combination scenari are listed bellow:

- In some applications, several classifiers can be obtained from different sources (different training patterns or different features). A typical example is the fusion of several modalities for biometric authentication: face, speech, fingerprint, etc. ;
- Sometimes, more than one training set is available. Different training sets can be collected at different times or in different conditions.

On the second hand, the classifier combination can be used for performance considerations. The goal is to design an ensemble that improves the accuracy and, if possible, without increasing the complexity of the testing process. This is usually achieved by one of the following ensemble design strategies:

- Combine different learning algorithms instead of only keeping the one giving the lowest estimation of the expected risk;
- For a given learning algorithm, combine several candidate solutions. Several candidates can be obtained by various initialization procedures or various model selection strategies (as discussed in section 2.6);
- Train several classifiers on subset of the training set or feature set. Some classifiers can be more efficient on local subspaces and act like experts on their local area.

In the section 3.2 we will analyze several motivations for using ensembles instead of one single classifier. Then, we will briefly review the main approaches proposed in the literature for fusing classifier outputs in section 3.3. Finally we will review some popular ensemble creation methods: bagging and variants in section 3.4 and Boosting in section 3.5.



(a) Statistical reason (b) Computational reason (c) Representational reason

Figure 3.2: Motivations for combining several classifiers instead of only one according to [30].

3.2 Motivations

In [30] Dietterich gives three main reasons explaining why en ensemble of classifier may work better than a single classifier.

- **Statistical Reasons:** In many classifier design procedures, the best solution f^* is not unique and there is no criterion for picking one particular solution. For example, when the training set is small, many classifiers may obtain good performances on this training set. It has been shown that combining several of these solutions we are likely to find a solution with better generalization properties. This is depicted in figure 3.2(a);

- **Computational Reasons:** Many learning techniques use local searches to converge toward the solution (e.g. neural networks), with the risk of staying stacked in local optima. Running several searches (e.g. using different initializations) and combining the solutions can improve the performances. See figure 3.2(b);

- **Representational Reasons:** Finally, when we choose a learning algorithm, we fix the class of functions $\mathcal{F}$ in which we seek the solution. However, the true optimal solution may lie outside of this space. By combining several solutions in $\mathcal{F}$ we can reach functions outside $\mathcal{F}$. The simplest illustration is to combine several linear decision functions in order to obtain complex non-linear decision boundaries. See figure 3.2(c).

Another fundamental advantage of combining classifiers is that it can protect from overfitting. Freund et al [43] explain that averaging classifiers reduces the variance of the decision function. They give theoretical properties in the case of Bayesian averaging.

From a learning theory perspective, it has been shown that the expected risk of the average of a set of models is better than the average of the expected risk of these models.

For example, consider a simple ensemble g over K models f_k:

$$g(\mathbf{x}) = \sum_k f_k(\mathbf{x}). \tag{3.1}$$

The mean square error risk of a single classifier f_k at $\mathbf{x}$ is $e_k(\mathbf{x}) = E[(y - f_k(\mathbf{x}))^2]$. The average risk of a model is $\bar{e}(\mathbf{x}) = \sum_k e_k(\mathbf{x})$. The average risk of the ensemble g is $e(\mathbf{x}) = E[(y - g(\mathbf{x}))^2]$. If we define the diversity of model f_k by $d_k(\mathbf{x}) = (f_k(\mathbf{x}) - g(\mathbf{x}))^2$, then the average diversity becomes $\bar{d}(\mathbf{x}) = \sum_k d_k(\mathbf{x})$. It can be shown that $e(x) = \bar{e}(\mathbf{x}) - \bar{d}(x)$.

3.3 Fusion of Classifier Outputs

A first simple way to combine classifiers is to combine the outputs of each classifiers for deciding the label associated to each test example. The combination process can be viewed as a mapping f_{comb} from the space of classifiers outputs to a label y as shown in equation (3.2). This mapping is usually called combiner. Let us consider that we have K classifiers to be combined with decisions d_i, $i \in \{1, \ldots, K\}$. Each decision d_i can be either a class label y_i or a posterior probability $p_i(y|\mathbf{x})$ on any other classifier output information (e.g. margin).

$$\begin{aligned} f_{comb} : \mathbb{R}^K &\rightarrow \{-1, +1\} \\ (d_1, d_2, \ldots, d_K) &\mapsto y = f_{comb}(d_1, d_2, \ldots, d_K). \end{aligned} \tag{3.2}$$

There are basically two possibilities for choosing f_{comb}: fixed rules and trainable rules. On the first hand, fixed rules (also called non-trainable rules) are simple combiners that fuse decisions of given classifiers. They do not require any additional data. On the second hand, trainable rules can be much more complicated combiners. In fact finding a good trainable combiner can be viewed as a new pattern recognition problem, taking as training features the outputs of all classifiers. In order to have an unbiased training process, this strategy requires an additional training set that was not used for training or testing the individual classifiers [30].

3.3.1 Non Trainable Combiners

The simplest non-trainable combiner is probably the most widespread in the multiple classifier system community. Majority voting simply returns the class with the highest number of votes. There has been a huge amount of research concerning theoretical aspects of majority voting for several decades, and, despite its simplicity it has proved to be very efficient is most applications.

In some applications, additional information can be derived from each classifier. Instead of only using the final class labels to which each example is estimated to belong, it is also possible to produce a ranked list of the candidate classes. The combination of the classifiers can directly use such lists. The most popular ranked majority voting strategy is called Borda Count [9].

Majority voting only takes into account the labels output by each individual classifier. A natural way to use more information is to use posterior probabilities for taking into account a *confidence* measure for each classifiers.

Several probability rules have been proposed, each of them being built on particular probability assumption. A detailed review of these assumptions is given in [76].

Let us recall the context. Consider that we want to classify a pattern $\mathbf{x}$ in one of the C classes $(c_1, ..., c_C)$. We model each of the C classes by the probability density functions $p(\mathbf{x}|c_k)$ and its a priori probability by $P(c_k)$. Assume that we have K classifiers to be combined. Let us denote by $p_j(c_k|\mathbf{x})$ the posterior probability estimated from the j-th classifier that $\mathbf{x}$ belongs to class c_k. The Bayes decision rule equation (2.1) states that an example $\mathbf{x}$ is assigned to the class c_i if:

$$P(c_i|\mathbf{x}) > P(c_k|\mathbf{x}), \text{ for } k = 1, ..., C; k \neq i \tag{3.3}$$

Equation (3.3) relies on the Bayesian framework. By making various assumptions on the posterior conditional distributions of each classifier, we can derive several combination rules. We give hereafter five simple probabilistic rules.

- Product rule: Example $\mathbf{x}$ is assigned to the class c_i if for $k = 1, ..., C; k \neq i$:

$$P(c_i) \prod_{j=1,..,N} p_j(\mathbf{x}|c_i) > P(c_k) \prod_{j=1,..,N} p_j(\mathbf{x}|c_k) \tag{3.4}$$

  This rule derives directly from Bayes theorem by assuming that the measurements of the different classifiers are conditionally independent;

- Sum rule: Example $\mathbf{x}$ is assigned to the class c_i if for $k = 1, ..., C; k \neq i$:

$$(1-N)P(w_i) + \sum_{j=1,..,N} P_j(c_i|\mathbf{x}) > (1-N)P(w_k) + \sum_{j=1,..,N} P_j(c_k|\mathbf{x}) + \sum_{j=1,...,N} P_j(c_k|\mathbf{x}) \tag{3.5}$$

The sum rule derives from the product rules by assuming that the posterior probabilities only slightly differs from the know priors [76]. Then from equation (3.4) and equation (3.5) we extract three other combination rules. For example, the sum rule in equation (3.5) can be approximated by the maximum posterior probabilities. In all the cases, example $\mathbf{x}$ is assigned to the class c_i if for $k = 1, ..., C; k \neq i$:

- Max rule:

$$\max_{j=1,...,N} P_j(c_i|\mathbf{x}) > \max_{j=1,...,N} P_j(c_k|\mathbf{x}) \tag{3.6}$$

- Min rule:

$$\min_{j=1,...,N} P_j(c_i|\mathbf{x}) > \min_{j=1,...,N} P_j(c_k|\mathbf{x}) \tag{3.7}$$

- Median rule:

$$\text{median}_{j=1,\ldots,N} P_j(c_i|\mathbf{x}) > \text{median}_{j=1,\ldots,N} P_j(c_k|\mathbf{x}) \tag{3.8}$$

3.3.2 Trainable Rules

Majority voting considers the decisions of each classifier uniformly, which means that each classifier will have the same weight in the final decision. However, some classifiers may be more robust than others and we would like to add more weights to these classifiers. This can be done by using weighted majority voting. The goal is to design a weight distribution on the classifiers outputs.

The optimal choice of the weights w_i, $i \in \{1, \ldots, K\}$ is given by the following theorem [82]

Theorem 3.1 (Kuncheva, 2004)**.** *Consider an ensemble of K independent classifiers with individual accuracies $p_1, p_2, \ldots, p_K$. The outputs are combined by weighted majority voting. Then the accuracy of the ensemble P_{wmaj} is maximized by assigning weights:*

$$w_i \propto \log \frac{p_i}{1 - p_i} \tag{3.9}$$

Another possible trainable combination is called Behavior Knowledge Space (BKS) [63]. It uses a multinomial combination of classifiers. The posterior probabilities are estimated for each classifier but also for each possible combination of votes. A look-up table (the BKS table) is designed through a training process. This table evaluates the optimal decision for each combination of votes. This technique proved to have very good generalization properties on large sample cases but is very sensitive to overfitting in small sample cases [127].

Finally a combiner can be viewed as a new pattern recognition task where the input patterns corresponds to outputs of the classifiers. For example, Collobert et al [20] proposed to use Neural Networks for combining several SVM.

3.3.3 Fixed Rules vs Trained Rules

Evaluating what is best between fixed on trainable rules is basically application dependent. However there are some commonly admitted general rules. A first notion that can be taken into account is the reliability of each classifiers. In some cases, the confidence expressed by a particular classifier may not be reliable. This problem is discussed in the framework of stacked generalization [171]. For example, consider a biometric identification system in which we combine the speech modality with a face identification system [130]. Let us imagine the scenario where the face image is captured in very bad conditions (e.g. poor lighting conditions or partial occlusion of the face due to external factors). The face modality becomes then less reliable than speech.

Many other characteristics can be considered for comparing fixed and trainable rules (see [33, 134]):

- Size of the dataset: trainable rules are preferred if the training dataset is large but if only a small validation set is available, simple rules like majority voting can outperform trainable rules. The complexity of the trained rule should be adapted to the size of this validation set. This is one reason why BKS overfits quickly on small datasets;
- Size of the ensemble: Small ensemble are preferred for trainable rules while large ensemble reduce the bias of fixed rules;
- Degree of *balance*: If the output of the classifiers are not of the same nature or uncalibrated: trainable rules generally outperform fixed rules.

In this section we presented several techniques for combining classifiers assuming that the classifiers are given. These methods use outputs of the given classifiers for taking the final decision. In the following section we will present ensemble creation methods. In other words, techniques that directly train the classifiers such that they can be combined optimally.

3.4 Bagging

3.4.1 Introduction

The first popular ensemble creation method is called Bagging (Bootstrap AGGregatING) [10]. The principle is to combine classifiers that are trained on bootstrap replicates of the training set. Given a training set Z_n, bagging generates K replicates $Z_n^{(1)}, Z_n^{(2)}, \ldots, Z_n^{(K)}$ of the same size n by randomly drawing elements of the original training set. The same element can be used several times in the same set. Moreover, classifiers are trained on each training set with the same prediction rule. Finally the ensemble decision is made by averaging the decisions. The bagging procedure is sketched in Algorithm 3.1. In order to obtain an efficient combination, parallel classifiers need to have diversity between them (this notion of diversity will be discussed more deeply in chapter 4). As each replicate comes from the same training set, the prediction rule needs to be *unstable* in order to guarantee diversity between classifiers. A classifier is known to be unstable if a slight change in the training set can give significant changes in the decision function. Typical unstable classifiers are neural networks or decision trees.

Breiman [10] first presented bagging as a variance reduction procedure mimicking averaging over several training sets. The underlying approximation should be kept in mind: averaging is performed on bootstrap replicates of a single training set, and not on different training sets. Thus, although experimental results often shown the expected variance reduction [10, 143], several other arguments have been given to explain the success of bagging. Schapire et al. [143] provide bounds for voting algorithms, including bagging, relating the generalization performance of aggregated classifiers to the margin distribution of examples. Unlike boosting, bagging does not explicitly maximize margins, but experiments show that for complex classifiers, bagging produces rather large margins. A phenomenon that cannot be explained by the variance reduction (which is an asymptotic analysis) is that outliers

Algorithm 3.1: Bagging algorithm [10]

- Training
  1. Given a training set Z_n create K bootstrap replicates $Z_n^{(k)}$
  2. For each bootstrap $Z_n^{(k)}$ select $f^*(Z_n^{(k)}) = \arg\min_{f \in \mathcal{F}} R_{emp}(f, Z_n^{(k)})$
- Testing
  Given an input $\mathbf{x}$, the corresponding output is:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} f^*(Z_n^{(k)})(\mathbf{x})$$

are particularly well handled by bagging [31]. In [52] Grandvalet explains the success of bagging by the stabilization provided by spreading the influence of examples, rather than reducing the variance.

3.4.2 Random Forests

Breiman proposed Random Forests [12] as a variant of Bagging. A random forest is an ensemble creation method that uses tree classifiers (see section 2.4.2) as base classifier. An ensemble of decision trees is built by generating independent identically distributed random vectors Θ_k, $k = 1, \ldots, K$. One tree is grown from each vector. The difference with Bagging is that the vectors Θ_k can be built by sampling from feature sets, sample set or by varying some parameters of the tree (e.g. number of nodes).

In [131], Rodriguez et al. proposed a variation of random forest called Rotation forests that simply adds a *PCA* pre-processing in order to decorrelate the training data. They show experimental improvements on many datasets.

3.5 Boosting

3.5.1 Boosting Theory

The underlying idea of boosting is to combine simple rules iteratively to form an ensemble that will improve the performances of each single member. Boosting theory has its roots in Probably Approximately Correct (*PAC*) learning [162]. *PAC* gives a nice formalism for deciding how much training data we need in order to achieve a given probability of correct predictions on a given fraction of future test data. In [162], Valiant showed that simple rules, each performing only slightly better than random guessing, can be combined to form an arbitrarily good ensemble. The challenge of boosting is to find a *PAC* algorithm with arbitrarily high accuracy.

Boosting seeks a function f of the form:

$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}), \tag{3.10}$$

where α_t is a weighting coefficient at iteration t and h_t is the simple rule used at iteration t (this rule is usually called weak hypothesis). Note that we employ the subscript t instead of k to underline that it is an iterative process.

First boosting algorithms were proposed by Shapire in 1990 [141] and Freund in 1995 [42]. However, some strong assumptions prevented to use efficiently these algorithms in practical situations: They need prior knowledge of the accuracy of the weak hypotheses.

3.5.2 AdaBoost

A first step towards more practical Boosting is AdaBoost (Adaptive Boosting) algorithm [44]. Adaboost is adaptive in the sense that a new hypothesis is selected given the performances of the previous iterations. Unlike bagging, this allows the algorithm to focus on the *hard* examples. This adaptive strategy is managed by a weight distribution D over the training samples. A weight $D(i)$ is given to each training pattern $\mathbf{x}_i$. Examples with large weights will have more impact for choosing the weak hypothesis than those with low weights. Then at each round, the weight distribution is updated such that weight of misclassified examples is increased.

Let us consider, as usual, a training set $Z_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$. We suppose that we have a base learning algorithm (or weak learner) which accepts as input a sequence of training samples Z_n along with a distribution D over the training samples. Given such input, the weak learner constructs a weak hypothesis $h : \mathbb{R}^d \to \mathbb{R}$. The predicted label y is given by $\text{sign}(h(\mathbf{x}))$ while the confidence of the prediction is given by $|h(\mathbf{x})|$. We also assume that the corresponding weighted training error is smaller than $\frac{1}{2}$. This means that the weak hypotheses have to be at least slightly better than random guessing with respect to the distribution D. The distribution D is first initialized uniformly over the training samples. It is then iteratively updated such that the likelihood of the objects misclassified in the previous iteration is increased. The general formulation of AdaBoost algorithm is given in 3.2.

The loss function $\mathcal{L}$ used for updating the weights (step 4 in the main loop) is usually an exponential loss-function as shown in equation (3.11), but other loss functions have been proposed (Logitboost [68] or arcing[11]).

$$\mathcal{L}(\alpha_t, y_i h_t(\mathbf{x}_i)) = \exp(-\alpha_t y_i h_t(\mathbf{x}_i)). \tag{3.11}$$

Two critical choices in AdaBoost are how to choose the weak hypothesis h_t and what is the

Algorithm 3.2: AdaBoost algorithm[44]

Input: $Z_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, number of iterations T

Initialize: $D_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$

for $t = 1, \ldots, T$, **do**

1. Train weak learner with respect to the weighted sample set $\{Z_n, D^{(t)}\}$
2. Obtain hypothesis $h_t : \mathbb{R}^n \to \mathbb{R}$
3. Choose optimal $\alpha_t \in \mathbb{R}$
4. Update the weights:

$$D_i^{(t+1)} = \frac{D_i^{(t)} \mathcal{L}(\alpha_t, y_i h_t(\mathbf{x}_i))}{N_t},$$

where N_t is a normalization constant such that $\sum_{i=1}^{n} D_i^{(t+1)} = 1$.

end

Output: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

optimal α_t. Let us define the training error of the ensemble H:

$$\mathcal{L}_{0/1}(H) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{0/1}(H(\mathbf{x}_i), y_i), \tag{3.12}$$

where $\mathcal{L}_{0/1}$ is the standard 0/1 loss function defined in equation (2.4). Shapire and Singer [144] give a bound on the training error:

Theorem 3.2 (Shapire, 1999)**.** *The training error $\mathcal{L}_{1/0}(H)$ of the output hypothesis H is bounded by:*

$$\mathcal{L}_{1/0}(H) \leq \prod_{t=1}^{T} N_t, \tag{3.13}$$

where $N_t = \sum_i D(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$ is the normalization factor defined in Algorithm 3.2.

According to Theorem 3.2, the training error can be minimized by greedily minimizing N_t on each round of boosting. For choosing the optimal α_t, we will consider several cases in the following sections.

3.5.3 Discrete AdaBoost

Let us first consider the original version of AdaBoost, called Discrete AdaBoost, when the range of each weak hypothesis is restricted to the labels $\{-1; 1\}$. Then the optimal α_t can

be found by approximating N_t as follows:

$$N_t = \sum_i D(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \tag{3.14}$$

$$\leq \sum_i D(i) \left(\frac{1 + y_i h_t(\mathbf{x}_i)}{2} \exp(-\alpha_t) + \frac{1 - y_i h_t(\mathbf{x}_i)}{2} \exp(\alpha_t) \right). \tag{3.15}$$

Following equation (3.14), the coefficient α_t that minimizes N_t is found analytically :

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right), \tag{3.16}$$

where $r_t = \sum_i D(i) y_i h_t(\mathbf{x}_i)$.

This choice of α_t leads to the following theorem concerning the training error:

Theorem 3.3 (Shapire, 1999)**.** *With the choice of α_t given in equation* (3.16)*, the training error of the output hypothesis H is bounded by:*

$$\mathcal{L}_{1/0}(H) \leq \prod_{t=1}^{T} \sqrt{1 - r_t^2}. \tag{3.17}$$

With this setting, the optimal hypothesis h_t is the weak hypothesis that maximizes $|r_t|$. This quantity r_t is a natural measure of the correlation of the predictions of h_t and the labels y_i with respect to D_t. From equation (3.17) it follows that:

$$\mathcal{L}_{1/0}(H) \leq \exp \left(- \sum_{t=1}^{T} \frac{r_t^2}{2} \right), \tag{3.18}$$

from which we infer that the condition $\sum_{t=1}^{T} r_t^2 \to \infty$ suffices to guarantee that the training error $\mathcal{L}_{1/0}(H)$ converges towards 0 when the number of iteration increases. Clearly this holds if $r_t \geq r_0 > 0$ for some positive constant r_0 (recall that weighting training error of of weak hypothesis is lower than 0.5).

3.5.4 Real AdaBoost

A more general formulation of Discrete AdaBoost uses confidence levels of each weak classifier instead of just binary outputs. It is called Real AdaBoost [144]. Unlike Discrete AdaBoost, the output space of the weak classifiers is not restricted to $\{-1; 1\}$, but can take values in $\mathbb{R}$. More specifically, let us consider a partition of $\mathbb{R}$ into disjoint blocks $X_1, X_2, \ldots, X_N$ for which $h(x') = h(x) = c_j$ for all $(x, x') \in X_j \times X_j$. Let us assume that the partitioning is given. The task is to find the optimal c_j, $i = 1, \ldots, N$. Let

$$W_j^+ = \sum_{i : \mathbf{x}_i \in X_j, y_i = +1} D(i),$$

be the weighted fraction of positive samples falling into partition X_j, and

$$W_j^- = \sum_{i:\mathbf{x}_i \in X_j, y_i = -1} D(i),$$

the equivalent for the negative class. Then the normalization factor N_t in Algorithm 3.2 can be rewritten as:

$$N_t = \sum_j \left(W_j^+ \exp(-c_j) + W_j^- \exp(c_j) \right). \tag{3.19}$$

This expression of N_t that we want to minimize gives the optimal c_j:

$$c_j = \frac{1}{2} \log \left(\frac{W_j^+}{W_j^-} \right). \tag{3.20}$$

This technique proved to be very effective in many application, outperforming Discrete AdaBoost by taking into account confidence measure of the weak hypotheses. A good example of weak learner that can be used using this partitioning technique is a simple decision tree. The leaves of the tree directly define the partition of the domain. Note that in practice, W_j^+ or W_j^- is very small which would produce inconsistency in equation (3.20). That is why we generally use a smoothed version of the coefficients:

$$c_j = \frac{1}{2} \log \left(\frac{W_j^+ + \epsilon}{W_j^- + \epsilon} \right), \tag{3.21}$$

with $\epsilon > 0$ being an appropriately small constant.

3.5.5 Generalization Error

So far we only considered training error convergence to prove the efficiency of AdaBoost, but, as described in chapter 2, minimizing the empirical risk is not a guaranty of good generalization. However, in practical situations, AdaBoost seems to be very unlikely to overfit and presents one more interesting property: the test error continues decreasing with the number of iterations, even if the training error has reached 0. In order to explain this phenomenon, let us analyze AdaBoost from a *margin* perspective. Let us recall that the ensemble hypothesis has the form:

$$H(\mathbf{x}) = \text{sign}\left(f(\mathbf{x}) \right),$$

with

$$f(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x}).$$

Without loss of generality, we can consider that $\sum_t \alpha_t = 1$. Let us call $\mathcal{H}$ the space of the hypotheses h_t. Similarly to margins in Support Vector Machines, we can define the margin of example $\mathbf{x}$ with label y to be $\rho = yf(\mathbf{x})$. The value of the margin can be seen as a

confidence in the prediction of H. The following theorem concerning generalization error has been proved in [144]:

Theorem 3.4 (Shapire, 1999)**.** *The Generalization error can be upper bounded with probability* $1-\delta$ *for all* $\theta > 0$ *and for all* f *by:*

$$P_{Z_n}\left(yf(\mathbf{x}) \leq \theta\right) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\left(\frac{d\log^2(n/d)}{\theta^2} + \log\left(\frac{1}{\delta}\right)\right)^{1/2}\right), \tag{3.22}$$

where P_{Z_n} *denotes the probability with respect to choosing an example* $(\mathbf{x}, y)$ *uniformly from the training set* Z_n *and* d *can be viewed as the VC dimension of* $\mathcal{H}$*.*

The bound expressed in 3.4 does not depend on the number of classifiers in the ensemble T but remains quite loose in practical applications. However, it shows the tendency that larger margins lead to better generalization. In fact, the generalization continues improving after the training error has reached zero because the margin still increases. Note that several studies have tried to interpret AdaBoost as a soft margin classifiers [97, 125, 126, 137, 142].

3.5.6 A Probabilistic Interpretation of Boosting

In [68], Friedman et al. propose a statistical interpretation of AdaBoost by fitting an additive model $\sum_t f_t(\mathbf{x})$ with $f_t(\mathbf{x}) = \alpha_t h_t(\mathbf{x})$. A simple way would be to minimize squared-error loss $E[(y - \sum_t f_t(\mathbf{x}))^2]$ in a forward stagewise manner. However, squared-error loss is commonly used in linear regression but is not appropriate for classification (the examples too well classified are penalized, as shown in figure 3.3). A better choice of loss function is to use a binomial log-likelihood. It is depicted in figure 3.3 with other standard loss functions. The procedure is then called LogitBoost.

The additive logistic model has the form:

$$\log\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})} = \sum_{t=1}^{T} f_t(\mathbf{x}). \tag{3.23}$$

Others standard AdaBoost versions like Discrete AdaBoost and Real AdaBoost can also be interpreted as a additive logistic regression model by considering an exponential criterion:

$$J(H) = E[\exp(-yH(\mathbf{x}))]. \tag{3.24}$$

The function H that minimizes $J(H)$ is the symmetric logistic transform of $p(y=1|\mathbf{x})$:

Theorem 3.5 (Friedman 2000)**.** $E[\exp(-yH(\mathbf{x}))]$ *is minimized at*

$$H(\mathbf{x}) = \frac{1}{2}\log\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}. \tag{3.25}$$

Hence

$$p(y=1|\mathbf{x}) = \frac{\exp(H(\mathbf{x}))}{\exp(-H(\mathbf{x})) + \exp(H(\mathbf{x}))}, \tag{3.26}$$

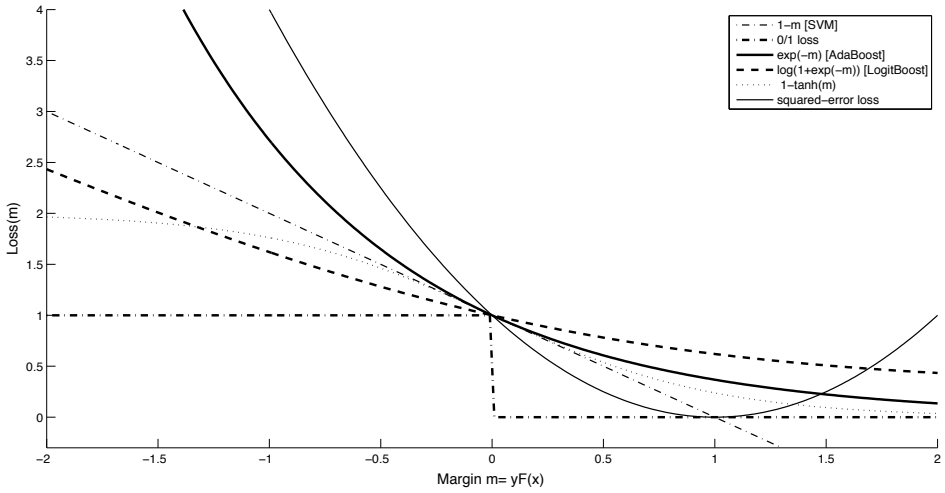


Figure 3.3: Examples of loss functions including exponential loss (AdaBoost) and logistic loss (LogitBoost).

$$p(y = -1|\mathbf{x}) = \frac{\exp(-H(\mathbf{x}))}{\exp(-H(\mathbf{x})) + \exp(H(\mathbf{x}))}. \tag{3.27}$$

This analysis gives a rather simple new interpretation of boosting. This probabilistic interpretation will be used later in this thesis, in particular for efficient posterior probability estimation.

3.6 Analysis

In the previous sections we presented several ensemble creation methods like Bagging and Boosting. Depending on the application, some techniques perform better than others. In this section we will give a general overview of the advantages and drawbacks of these algorithms, trying to show why these ensemble creation techniques work.

Several studies ([45, 78, 110]) have proposed theories for studying the effectiveness of Bagging and Boosting. They are based on a bias plus variance decomposition of classification error as proposed in [49]. In this decomposition we can view the expected error of a learning algorithm on a particular target function and training set size as having three components:

1. A bias term measuring how close the average classifier produced by the learning algorithm will be to the target function;

2. A variance term measuring how much each classifier will vary with respect to each other (how often they disagree);

3. A term measuring the minimum classification error associated with the Bayes optimal classifier for the target function (this term is sometimes referred to as the intrinsic

target noise).

Using this framework it has been suggested [10] that both Bagging and Boosting reduce error by reducing the variance term. Freund and Shapire [44] argue that Boosting also attempts to reduce the error in the bias term since it focuses on misclassified examples. Such a focus may cause the learner to produce an ensemble function that differs significantly from the single learning algorithm. In fact, Boosting may construct a function that is not even producible by its component learning algorithm (e.g. changing linear predictions into a classifier that contains non-linear predictions). This property makes Boosting an appropriate algorithm for combining the predictions of weak learning algorithms. Though the bias-variance decomposition is interesting, there are certain limitations to applying it to real-world data sets. To be able to estimate the bias, variance, and target noise for a particular problem, we need to know the actual function being learned. This is unavailable for most real-world problems. To deal with this problem Kohavi and Wolpert [78] suggest holding out some of the data. The main problem with this technique is that the training set size is greatly reduced in order to get good estimates of the bias and variance terms.

One major limitation of AdaBoost is its tendency to overfit in presence of noisy data. In fact the method for updating the weight distribution of the sample tends naturally to increase the weights of noisy samples. This will be discussed more deeply in the case of face detection in chapter 8.

3.7 Summary

This chapter was a general introduction to multiple classifier systems. We reviewed both aggregation techniques and ensemble creation methods like Bagging or Boosting. The next chapter will propose an information theoretic framework for combining classifiers. We will study diversity between members of the ensemble and its relationship with average individual accuracy. Then, chapter 5 will present a efficient multiple classifier technique that involves parallel SVM.

Information Theoretic Classifier Combination

4

4.1 Introduction

In the previous chapter we gave a broad overview of classifier combination techniques. In particular, we have seen that using an ensemble is only justified if the committee becomes better than the best individual member. In general, an ensemble will be performant if the classifiers are complementary in the sense that they commit errors on different data. In other words, the members of the ensemble need to be diverse. The notion of diversity appears to be a key element in ensemble methods, that is why, in this chapter, we will analyze more deeply diversity as a major feature for obtaining good ensembles. In the combination techniques described in chapter 3, diversity is very often implicitly used to improve the ensemble accuracy. However there are also techniques that directly grow ensembles by maximizing diversity between classifiers. Thus, many explicit diversity measures have been proposed in the literature.

In this chapter, we will propose an Information Theoretic (IT) framework for analyzing diversity and its close relationship with average individual accuracy of the ensemble members. More precisely, we will show that these two quantities appear to be contradictory. We will consequently propose an information theoretic measure that exhibits a trade-off between diversity and average individual accuracy.

In section 4.2 we will review the main existing diversity measures and show their limitations for building efficient ensembles. Then in section 4.3 we will briefly review information theoretic principles for classification. Section 4.4 will propose the IT framework to classifier combination techniques. Then a new IT measure will be defined in section 4.5. Finally the last sections will give some example of practical use of the new measure. In particular, section 4.7 proposes an modification of AdaBoost that incorporates the new measure for selecting weak classifiers.

4.2 Diversity in Ensembles of Classifiers

It is commonly admitted that a large diversity between classifiers in a team is preferred. However, diversity can be understood differently depending on the context. It can be viewed as a measure of dependence, complementarity or even orthogonality between classifiers [85]. In practice, diversity can be used in three different philosophies. First, it can be directly used as an optimization criterion for training the ensemble. Then, it is often used implicitly in the ensemble growing techniques, where the ensembles become progressively diverse (e.g. AdaBoost). Finally it can be used for controlling the pertinence of ensemble by checking if it is diverse enough.

In [27], Cunningham et al. claim that "any work with classification ensembles should explicitly measure diversity in the ensemble". Giacinto et al. [50] states that classifiers in an ensembles need to be "accurate and diverse". Several studies focused on understanding how diversity was handled on various ensemble creation techniques like AdaBoost or Bagging [80, 149]. Finally, many techniques have been proposed for exploiting diversity for finding good ensembles [14, 56, 79, 98, 151, 169]. It was even proposed to voluntarily overtrain the classifiers in order to create diversity between them [26]. In all these studies, various diversity measures have been proposed. We give hereafter a general overview of the most significant ones.

4.2.1 Diversity Measures

Diversity measures can be splitted into pairwise and non pairwise diversity measures. On the first hand, pairwise diversity measures require consideration of the diversity between each pair of classifiers and then averaging the $\binom{K}{2}$ diversity values. Let us consider two classifiers represented by their outputs y_1, y_2 and denote y the true label. We define the following probabilities of the respective pairs of correct/incorrect classifications:

$$a = P(y_1 = y, y_2 = y) \tag{4.1}$$

$$b = P(y_1 \neq y, y_2 = y) \tag{4.2}$$

$$c = P(y_1 = y, y_2 \neq y) \tag{4.3}$$

$$d = P(y_1 \neq y, y_2 \neq y) \tag{4.4}$$

The most used diversity measure is certainly Yule's Q-statistic [181] (QS). It is defined by:

$$Q = \begin{cases} \frac{ad-bc}{ad+bc}, & \text{if} \quad a, b, c, d < 1 \\ 1, & \text{otherwise.} \end{cases} \tag{4.5}$$

As shown in equation (4.5), two statistically independent classifiers will have $Q = 0$. Q varies between -1 and 1, the lower the value the more diverse the classifiers. Classifiers that tend to recognize the same objects correctly will have positive values of Q, and those which commit errors on different objects will render Q negative.

Then the correlation coefficient ρ is defined by:

$$\rho = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}. \tag{4.6}$$

It is very similar to Q, i.e. it can be shown that Q and ρ have the same sign.

The Disagreement Measure [156] simply corresponds to the total proportion of examples for which the two classifiers disagree:

$$D = b + c. \tag{4.7}$$

It varies between 0 and 1, the higher the value the more diverse the classifiers.

Double fault [50] counts the proportion of examples misclassified by both classifiers:

$$DF = b + c. \tag{4.8}$$

In the following of the chapter we will also consider a diversity measure based on the Mutual Information (MI) between the output of two classifiers y_1 and y_2:

$$MI = I(y_1, y_2) = \sum_k \sum_j p(y_1 = k, y_2 = j) \log \frac{p(y_1 = k, y_2 = j)}{p(y_1 = k)p(y_2 = j)}. \tag{4.9}$$

For all these pairwise measures, the diversity of the ensemble of K classifiers is computed by averaging the $\binom{K}{2}$ measures.

On the other hand, there exists several non-pairwise diversity measures. The most used in the Kohavi-Wolpert variance [78]:

$$KWv = \frac{1}{2}\left(1 - \sum_{k=1}^{K} P(y = y_k|\mathbf{x})^2\right). \tag{4.10}$$

The diversity measures presented in this section clearly present correlation between them and, as pointed out in [80], there is no best diversity measure that can be used for building ensembles with minimal error. Moreover, finding a systematic relationship between these diversity measures and ensemble accuracy is a more challenging task.

4.2.2 Limits of Diversity Measures

These diversity measures have been extensively used in many applications, particularly in the context of classifier selection from a large set of classifiers [50]. However, is has been observed in partice that maximizing the diversity measures is not necessarily a guarantee for obtaining the best ensembles. In this section we will see through a practical example that diverse ensembles can even decrease the performances compared to the best classifier in the team.

In order to show some of these limitations, we perform simple experiments that evaluate how diversity measures are correlated to the performances of an ensemble of classifiers. In

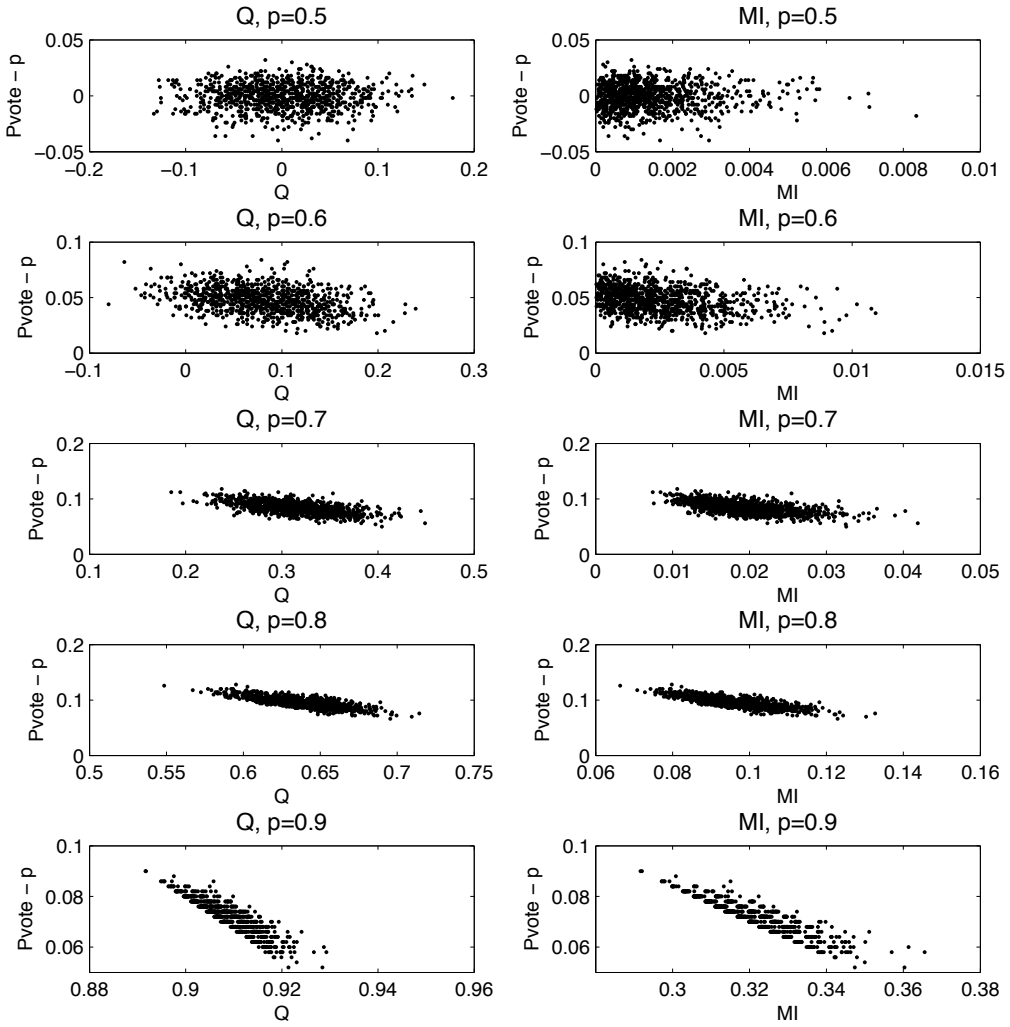


Figure 4.1: Improvement of the ensemble with respect to the average individual accuracy $p = \{0.5, 0.6, 0.7, 0.8, 0.9\}$, function of 2 diversity measures.

this study we consider two diversity measures: the QS (equation (4.5)) and the information theoretic measure based on Mutual Information (MI) (equation (4.9)).

In the first experiment, we consider classifiers with strictly equal individual accuracies $p \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Considering this constraint, 1000 binary outputs were randomly generated for 3 classifiers: $\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3} \in \{-1, 1\}^{1000}$. The committee decision is taken by majority voting among the 3 classifiers. This process is then repeated several times. For each trial, we measured the ensemble accuracy *pvote* and we compared two diversity measures: the average Q-statistics and the average MI. Results are reported in figure 4.1.

In the second experiment p is randomly distributed in the interval $p \in [0.7\,, 0.8]$ (see

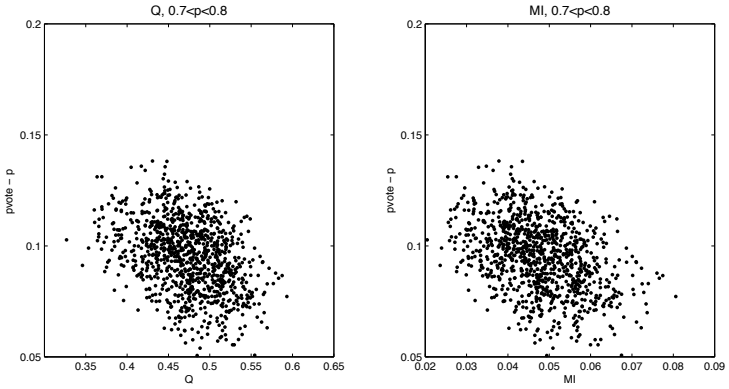


Figure 4.2: Improvement of the ensemble w.r.t. the average individual accuracy $p \in [0.7\ 0.8]$, function of 2 diversity measures.

figure 4.2).

It turns out that the diversity (both QS and MI) seems to be a relevant feature when the classifiers have similar individual accuracies, supposing that this accuracy is not too low (figure 4.1 with $p \geq 0.7$). When they have equal but low accuracies, large diversity does not necessarily imply improvements (figure 4.1 with $p < 0.7$). Moreover, figure 4.2 shows that even if the classifiers have only slight differences in terms of individual accuracy, diversity between them is not a highly discriminant feature for choosing the best ensemble (this remark is important as in practical applications we cannot ensure that the classifiers have exactly the same individual accuracy).

These considerations explain why, at least for majority voting combination, diversity is usually only used for visualization (plot pairs of classifiers according to their diversity), or overproduction and selection of classifiers. In [150], Shipp et al. also underline that practice does not give a clear positive relationship showing that highly diverse ensembles have high accuracy on combination. Kuncheva also reported in [83] that the improvement on the best individual accuracy by forcing diversity is negligible. More details about diversity and how to create diversity in ensemble are given in [14].

The remaining of this chapter will investigate this paradoxical behavior of diversity. On the one hand it is known as a major factor in ensemble design, on the other hand it is not a relevant feature that systematically leads to improved performances compared to the best member of the team.

4.3 Introduction to Information Theoretic Classification

4.3.1 Motivations

In this section we will introduce an information theoretic framework for combining classifiers. We will first motive our choice by showing how IT can help tackling the ambiguity around diversity as explained in the previous section. We will use information theoretic

tools to understand why selecting the most diverse ensemble is not necessarily the best choice.

Information theory is commonly used in coding and communication applications and more recently, is has also been used in classification area. Information theoretic classification was first introduced by Principe et al. in [123]. It presents a slightly different view compared to the pattern recognition framework presented in chapter 2. Basically, a learner is viewed as an agent that gathers information from some external sources. Information theoretic quantities have been widely used for feature extraction and selection: Fisher et al. [40], Hild et al. [60] or Sindhwani et al. [153], who proposed a feature selection technique for support vector machines and neural networks. Recently, Butz et al. [17] proposed to apply this framework to multi-modal signal processing. This work has also been applied to audio-visual speaker detection [7] or audio-visual speech recognition [55].

Multi-modal signals represent several signals of different modalities but coming from the same physical scene. The underlying idea is that the information contained in one signal can help for the processing of other modalities, and, *IT* offers a variety of tools for handling the exchange of information between the source and the signals, and between the signals of several modalities.

In this work, we propose to model classifier combination as a similar problem, considering that several classifiers are trained from examples coming from the same physical sample distribution. *IT* can thus provide efficient tools for measuring and analyzing dependence between classifiers and of course accuracy of the classifiers.

4.3.2 Information Theoretic Definitions

This section reviews some basic *IT* concepts that will be used in the remaining of the chapter. More details can be in found in [22].

We first introduce the concept of entropy, which measures the amount of uncertainty of a random variable. Let X be a random variable with probability density function $p(x)$.

Definition 4.1. *Shannon's entropy $H_S(X)$ of a discrete random variable X is defined by*

$$H_S(X) = -\sum_k p(x_k) \log p(x_k). \tag{4.11}$$

The entropy is expressed in bits. We can also define the conditional entropy of a random variable Y given X:

Definition 4.2. *The conditional entropy of a random variable Y given X is defined by:*

$$H_S(Y|X) = \sum_k p(x_k) H_S(Y|X = x). \tag{4.12}$$

Another important notion is the mutual information (MI) which measures the dependence between two random variables X and Y.

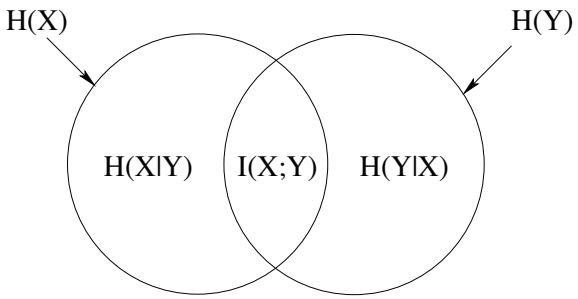


Figure 4.3: Venn Diagram representing the concept of entropy and mutual information. Mutual information can be viewed as the intersection between the marginal entropies.

Definition 4.3. *Consider two random variables X and Y with a joint probability density function $p(x, y)$ and marginal probability density functions $p(x)$ and $p(y)$, then Shannon's mutual information $I_S(X;Y)$ between X and Y is defined by*

$$I_S(X;Y) = \sum_k \sum_j p(x_k, y_j) \log \frac{p(x_k, y_j)}{p(x_k)p(y_j)} \tag{4.13}$$

MI is in fact the relative entropy between the joint distribution and the product distribution. For notation simplicity and if not specified otherwise, Shannon's definitions $H_S(Y|X)$ and $I_S(X;Y)$ will be written $H(Y|X)$ and $I(X;Y)$. The relationships between entropy and mutual information are summarized by the following theorem:

Theorem 4.1.

$$\begin{aligned} I(X;Y) &= H(X) - X(X|Y), && (4.14)\\ I(X;Y) &= H(Y) - X(Y|X), && (4.15)\\ I(X;Y) &= H(X) + H(Y) - H(X,Y), && (4.16)\\ I(X;Y) &= I(Y;X), && (4.17)\\ I(X;X) &= H(X). && (4.18) \end{aligned}$$

The relationships between entropy and mutual information given in Theorem 4.1 can be represented graphically by means of Venn diagrams as shown in figure 4.3. Mutual information represents the information that is shared by both variables X and Y. It can thus be represented by the intersection between both marginal entropies $H(X)$ and $H(Y)$.

Shannon's definitions of entropy and mutual information have been extended to the more general Renyi's definitions:

Definition 4.4. *For an entropy order $\alpha > 0, \alpha \neq 1$, Renyi's entropy is defined by*

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \sum_k p^\alpha(x_k). \tag{4.19}$$

Definition 4.5. *For an entropy order $\alpha > 0, \alpha \neq 1$, Renyi's mutual information is defined by*

$$I_\alpha(X;Y) = \frac{1}{1-\alpha} \log \sum_k \sum_j \frac{p^\alpha(x_k, y_j)}{p^{\alpha-1}(x_k) p^{\alpha-1}(y_j)}, \quad \alpha > 0, \alpha \neq 1. \tag{4.20}$$

Although Renyi's definitions have a discontinuity at $\alpha = 1$, it can be seen that the limit of Renyi's entropy as α goes to one is Shannon's entropy. In fact this statement is true for all Renyi's definitions. [36]

Then the notion of Markov chains will be used to model the classification process.

Definition 4.6. *Random variables X, Y, Z are said to form a first order Markov chain in that order if the conditional distribution of Z depends only on Y and is conditionally independent of X. The Markov chain is represented by:*

$$X \rightarrow Y \rightarrow Z \tag{4.21}$$

An important information theoretic theorem demonstrates that no processing of Y, deterministic or random, can increase the information that Y contains about X:

Theorem 4.2. *(Data Processing inequality) If $X \rightarrow Y \rightarrow Z$ then*

$$I(X;Y) \geq I(X;Z). \tag{4.22}$$

4.3.3 Information Theoretic Classification

Classification process can be modeled using an information theoretic framework. Let us first introduce some notations and variables concerning information theoretic classification, that will be used in the remaining of the chapter.

Introduction

In order to formulate the supervised classification problem in an information theoretic framework, let us first see how the information theoretic definitions presented in previous section can be adapted to the context of classification. Let us assume that we have a set X of n examples, obtained from a physical signal that we denote S. The true class labels of the examples are represented by a random variable C. C is defined over the set of classes Ω_c ($\Omega_c = \{-1; 1\}$ in a binary classification task). Let us denote F the feature vectors of the examples, obtained by feature selection and extraction. The class labels estimated by the classification process, (i.e the output of the classifier) is called $\hat{C}$. The common classification problem can be summarized by a simple processing chain (as in figure 2.1, chapter 2): acquisition of the signal, feature selection and extraction and classification. In information theory, this processing chain can be formulated as a first order Markov chain ([17, 123]). The Markov chain is depicted in figure 4.4, along with the main classification steps.

The ultimate goal in classification is to minimize the difference between the true labels and the estimated class labels. This can be modelled by considering a random variable

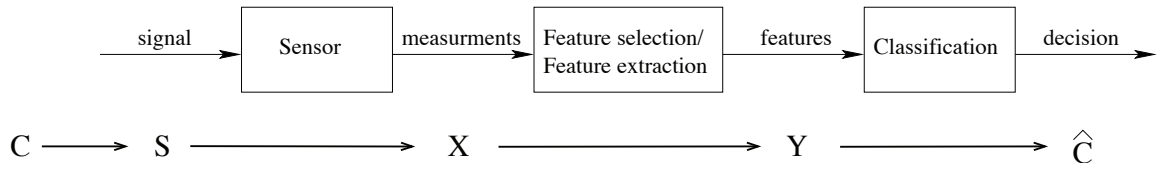


Figure 4.4: Different stages of pattern recognition systems, formulated as a first order Markov chain.

E taking values into $\{1, 0\}$. The probability of making an error during the classification process is thus:

$$P_e = P(E = 1) = P(\hat{C} \neq C). \tag{4.23}$$

In this work, we will consider the classification as a general problem of classifying examples to classes. We will thus integrate the pre-processing steps (signal acquisition, feature selection and feature extraction) into the classification step, resulting in one single random variable $\hat{C}$ for the whole classification process. Data processing inequality Theorem 4.2 proves that these pre-processing steps cannot increase the amount of information between C and $\hat{C}$.

The complete classification process can thus be simplified into the following first order Markov chain:

$$C \to \hat{C} \to E. \tag{4.24}$$

Intuitively, the estimated class labels $\hat{C}$ should contain as much information about the class labels C as possible. In other words, we would like to maximize the mutual information between the true labels and the estimated labels $I(C; \hat{C})$. This intuitive idea can be formalized by trying to minimize bounds on the error probability P_e (equation (4.23)).

Maximizing Mutual Information

The error probability P_e can be lower-bounded by applying Fano's inequality to the Markov chain equation (4.24). Fano's inequality [38] is given in the following theorem:

Theorem 4.3. *Considering the first order Markov chain defined in equation* (4.24)*, the probability of making an error is bounded by:*

$$P_e \geq \frac{H_S(C|\hat{C}) - 1}{\log |\Omega_c|} = \frac{H_S(C) - I_S(C; \hat{C}) - 1}{\log |\Omega_c|}, \tag{4.25}$$

where $|\Omega_c|$ *is the number of classes.*

From this lower bound, Erdogmus et al. [36] also derived an upper bound using Jensen's inequality described in [22]. The bounds are summarized in the following theorem:

Theorem 4.4. *[36]*
Considering the first order defined in equation (4.24)*, the probability of making an error*

is bounded by:

$$\frac{H_S(C) - I_\alpha(C;\hat{C}) - h_S(P_e)}{\log|\Omega_c| - 1} \leq P_e \leq \frac{H_S(C) - I_\beta(C;\hat{C}) - h_S(P_e)}{\min_k H_S(C|e,\hat{c}_k)}, \tag{4.26}$$

where $h_S(P_e) = -P_e \log P_e - (1-P_e)\log(1-P_e)$ *is the binary Shannon's entropy, and* $I_{\alpha,\beta}(C;\hat{C})$ *represents Renyi's definition of the mutual information with* $\alpha, \beta \in \mathbb{R}^+ \setminus \{1\}$.

The tightest bounds in Theorem 4.4 are obtained when the Renyi's entropy coefficients (α, β) tend to 1 in which case Renyi's definitions correspond to Shannon's ones [36]. As the number of classes $|\Omega_c|$ is fixed, the entropy of the class labels $H_S(C)$ does not depend on the classification process. Bounds in equation (4.26) point out that maximizing the MI between the two random variables C and $\hat{C}$ will tend to minimize both bounds, thus increasing the chances of having a low error probability P_e. Clearly minimizing both bounds in Theorem 4.4 does not mean necessarily minimizing P_e, however, if the lower bound is high, the error we be also be high.

In the extreme case where the classifier predicts correctly all the samples then we obtain the maximal possible mutual information: $I(C;\hat{C}) = H(C) = H(\hat{C})$.

In the next section we will extend these properties to the framework of multiple classifiers.

4.4 Information Theoretic Combination of Classifiers

The information theoretic framework introduced in the previous section was referring to the general classification task. This section shows how it can be extended to the case where the classification problem is more specifically a combination of several classifiers.

Let us assume that we have a team of K given classifiers. Let us now denote $\hat{C}$ the random variable representing the estimated class labels obtained by aggregation of the individual decisions. The aim is thus to find the best combination of members in the sense that it will maximize $I(C;\hat{C})$.

Let us call C_i, $i = 1, \ldots, K$, the random variables representing the decisions of classifiers $i = 1, \ldots, K$. Each classifier can be modelled separately using the Markov chain in equation (4.24). A more logical notation for the output of classifier i would be $\hat{C}_i$, in order to emphasize on the fact that the output labels are estimates of the true labels C. However, for notation simplicity we can write C_i instead of $\hat{C}_i$ as there is no risk of misunderstanding thanks to the subscript.

Conceptually, the combination process can be summarized by the Venn diagram shown in figure 4.5. The accuracy of an individual classifier C_i is represented by intersection between $H(C_i)$ and $H(C)$. The mutual information between two classifiers C_i and C_j is the intersection of the two corresponding marginal entropies $H(C_i)$ and $H(C_j)$.

For simplification and without loss of generality, let us consider a two class problem with labels $\{-1, 1\}$. The main relationships between the classifiers and the true classes are measured by the mutual information (MI) between them:

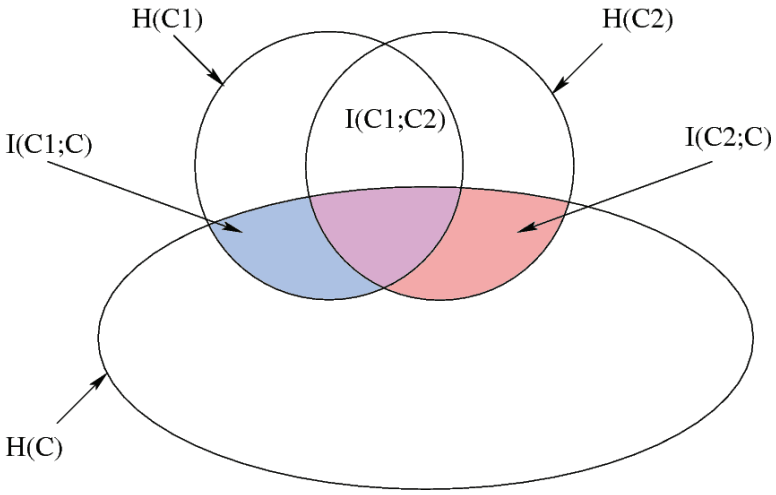


Figure 4.5: Venn Diagram representing relationships between two classifiers $C1$, $C2$ and the true class labels C.

- The MI between the output of individual classifier i and the true labels is $I_{C;C_i}$, $i \in \{1, \ldots, K\}$

- The MI between two classifiers is: $I_{C_i;C_j}$, $i, j \in \{1, \ldots, K\}$, $j > i$

Let $P_{C;\hat{C}}$ be the joint probability density function between true classes and the ensemble decision and P_C, $P_{\hat{C}}$ the corresponding marginal probabilities. The quantity that we want to maximize is the MI between the true labels and the labels obtained by aggregation of the individual decisions: $\hat{C}$:

$$I_{C;\hat{C}} = \sum_{k=-1,1} \sum_{j=-1,1} P_{C;\hat{C}}(k,j) \log \frac{P_{C,\hat{C}}(k,j)}{P_C(k) P_{\hat{C}}(j)}. \tag{4.27}$$

As described in the chapter 3, the combination can be implemented using variety of strategies. The simplest combination rule is the majority voting (MV). Despite its simplicity, MV has proved to be an effective rule for many combination tasks. Moreover, MV can easily be extended to weighted majority voting which is widely used in the multiple classifiers community. For example, the decision of AdaBoost [44] is a weighted majority vote of weak classifiers. Many studies [81, 89, 105, 138] have focused on analyzing why majority voting was as effective as more complicated schemes in improving the recognition results. In the remaining of the chapter, we restrict the combination rule to majority voting.

4.4.1 Majority Voting for Combining Classifiers

Considering a MV combination scheme, the probability $P_{\hat{C}}(y)$ that $\hat{C}$ outputs y is is related to each voter classifier by [147]:

$$P_{\hat{C}}(y) \leq \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} P_{C_i,C_j}(y). \tag{4.28}$$

Then, considering an odd number of independent classifiers $K > 1$ and assuming that they all have the same accuracy denoted p, the accuracy of the ensemble is:

$$P_{maj} = \sum_{m=\lfloor K/2 \rfloor+1}^{K} \binom{K}{m} p^m (1-p)^{K-m}. \tag{4.29}$$

The following result is known as the Condorcet Jury Theorem (1785):

Theorem 4.5. *[83]*

1. *If* $p > 0.5$ *then* P_{maj} *is monotonically increasing and* $P_{maj} \longrightarrow 1$ *as* $K \longrightarrow \infty$
2. *If* $p < 0.5$ *then* P_{maj} *is monotonically decreasing and* $P_{maj} \longrightarrow 0$ *as* $K \longrightarrow \infty$
3. *If* $p = 0.5$ *then* $P_{maj} = 0.5$.

This theorem supports the idea that we can expect improvements over the individual accuracy p only when p is larger than 0.5. This result can be extended to the case where p is different for each member, supposing that the distribution of individual accuracies p_i is symmetrical about the mean [147].

Nevertheless, these assumptions on the individual accuracies and independence of the classifiers are of course too strong in our framework as each classifier is trained using features extracted from the same data. Moreover, this theorem gives asymptotic behavior of the majority rule. We are more interested in small numbers of classifiers.

The following theorem gives the relationship between the ensemble accuracy and the individual accuracies as a function of the numbers of voters:

Theorem 4.6. *[147]*
Consider a group of odd size K with any distribution of the individual accuracies $(p_1, \ldots, p_K)$, where $p_i > 0.5\ \forall i$. The probability to reach the correct decision, when utilizing the simple majority rule, is larger than the probability $p = \frac{1}{K}\sum_{i=1}^{K} p_i$ of a random group member to do so.

In our case this theorem leads to:

$$P_{C,\hat{C}}(y) \geq \frac{1}{K} \sum_{i=1}^{K} P_{C,C_i}(y). \tag{4.30}$$

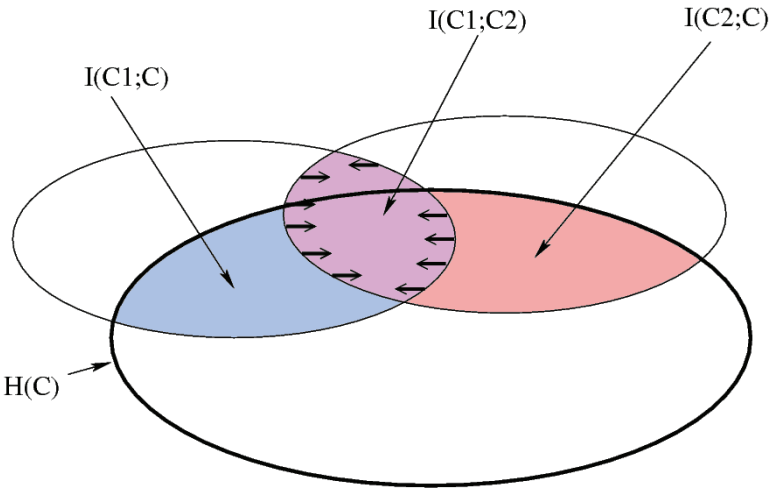


Figure 4.6: Venn Diagram showing how to optimize classifier combination.

Considering bounds equation (4.28) and equation (4.30) on each term of the mutual information in equation (4.27), we see that minimizing the MI between each pair of classifier $I_{C_i;C_j}, i \neq j$ and maximizing the MI between each single classifier and the true class labels $I_{C;C_i}$ will tend to maximize the mutual information between the ensemble decision and true labels: $I_{C;\hat{C}}$.

As introduced in section 4.3, $I_{C;C_i}$ represents the accuracy of classifier i. $I_{C_i;C_j}$ measures the similarity between the two classifiers i and j. In other words, by minimizing $I_{C_i;C_j}$, we maximize the diversity between the two classifiers.

This reasoning can be summed up in the following theorem that we propose:

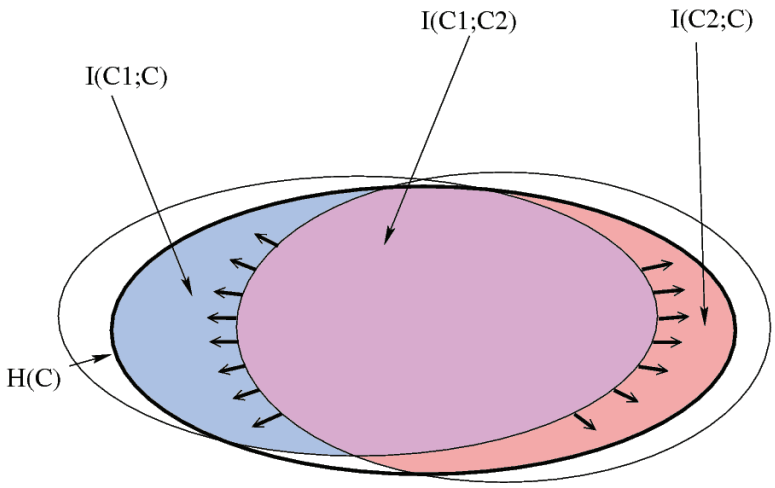
Theorem 4.7. *Let $C_1, C_2, \ldots, C_K$ be K random variables representing the output labels of K classifiers and C a random variable representing the true class labels. Maximizing $I_{C;C_i}$ $\forall i \in \{1, \ldots, K\}$ and minimizing $I_{C_i;C_j}$ $\forall i \in \{1, \ldots, K\}$, $\forall j \in \{1, \ldots, K | j > i\}$, will maximize $I_{C;\hat{C}}$. $\hat{C}$ represents the estimated class labels obtained from $C_1, \ldots, C_K$ by majority voting.*

It is important to note that Theorem 4.7 represents a sufficient condition for maximizing $I(C;\hat{C})$, but it is clearly not a necessary condition. In fact, it is possible to have an accurate ensemble of classifiers which does not maximize diversity between classifiers. This will be discussed experimentally in section 4.6.

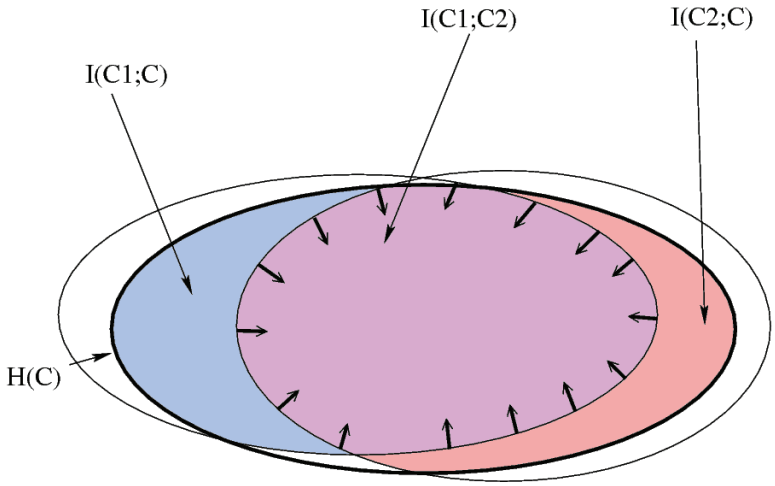
Theorem 4.7 can be summarized graphically by the Venn diagram shown in figure 4.6.

4.4.2 Diversity/Accuracy Dilemma

The relationships shown in Theorem 4.7 reveal a paradox in the sense that the two measures involved are somehow contradictory. In fact, two very good classifiers will clearly have very low diversity, while two poor classifiers, say slightly better than random guessing, will be very likely diverse. This paradox can easily be seen using Venn diagrams figure 4.7(a)



(a) Maximize both accuracies



(b) Maximize diversity

Figure 4.7: Diversity Accuracy dilemma.

and figure 4.7(b). Figure 4.7(a) shows that maximizing both individual accuracies will tend to expand the intersection between the two marginal entropies, which is equivalent to increase similarity between classifiers. Inversely, figure 4.7(b) shows that minimizing similarity between classifiers will force to decrease individual accuracies.

This phenomenon can be discussed through the following formalism. Consider two random variables C_1, C_2 representing two classifiers. Let C be the true class labels.

To establish a probabilistic link between the two classifiers, a parallel is made with the work of Butz et al in [17] concerning processing of multi-modal signals. First recall some pattern recognition definitions (see section 2.2 for details). We consider that the training and testing examples are generated from an unknown but fixed probability density function (*pdf*) and the task is to find a function that minimizes the risk of misclassifying new vectors

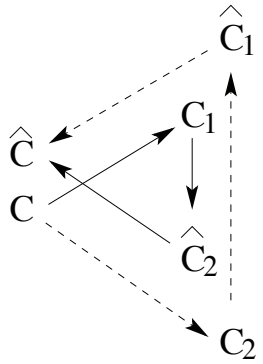


Figure 4.8: Coupled Markov chains for 2 classifiers trained differently from the same input data.

drawn from the same *pdf*. We can consider that the inputs of both classifiers C_1 and C_2 come from this *pdf*. Two coupled Markov chains can be built:

$$\begin{cases} C \to C_1 \to \hat{C}_2 \to \hat{C} \to E \\ C \to C_2 \to \hat{C}_1 \to \hat{C} \to E. \end{cases} \tag{4.31}$$

These coupled Markov chains are depicted in figure 4.8. The probability densities of C_1 and $\hat{C}_1$, resp. C_2 and $\hat{C}_2$, are both estimated from the same data sequences. Therefore we can write $I(C_1;\hat{C}_2) \approx I(C_2;\hat{C}_1) \approx I(C_1;C_2)$. Then, the data processing inequality (Theorem 4.2) gives: $I(C_1;C_2) \geq I(C;C_2)$ and $I(C_1;C_2) \geq I(C;C_1)$. This implies that:

$$I(C_1;C_2) \geq \frac{I(C;C_1) + I(C;C_2)}{2}. \tag{4.32}$$

Maximizing the individual accuracies represented by $I(C;C_1), I(C;C_2)$ will consequently maximize $I(C_1;C_2)$, the similarity between the classifiers. Inversely, minimizing $I(C_1;C_2)$ (maximizing the diversity) will tend to minimize the classifiers accuracy.

This phenomenon reflects the experiments presented in section 4.2.2 which showed that diversity was not a sufficient condition for improving the classification skills compared to the best individual member. To address this contradiction presented here, a trade-off needs to be introduced. A study of how the diversity evolves depending on the classifiers accuracies is given in the next section.

4.5 Information Theoretic Score

4.5.1 Estimation of the Relationship Between Diversity and Classifiers Accuracy

This section proposes an empirical estimation of the relationship between diversity and accuracy in order to give a computable measure of the ensemble performance. This link is estimated with the following experiment. Outputs of two classifiers (C_1, C_2) with equal

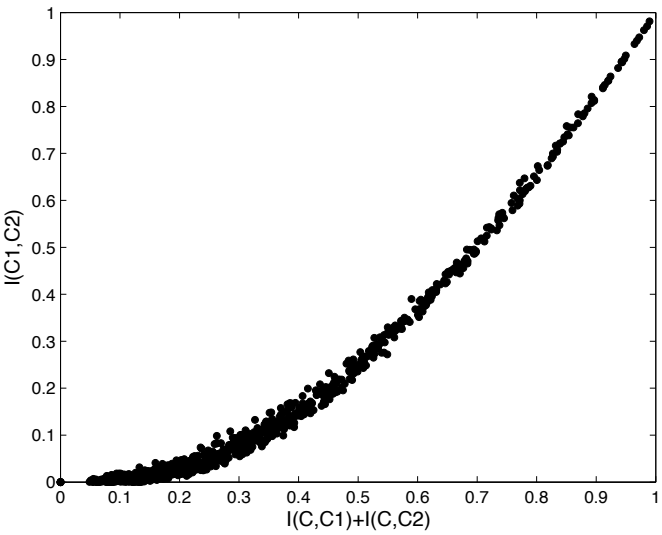


Figure 4.9: The similarity of 2 classifiers $I(C_1;C_2)$ function of the average individual accuracy $\frac{I(C_2;C)+I(C_1;C)}{2}$. The 2 classifiers have the same individual accuracy.

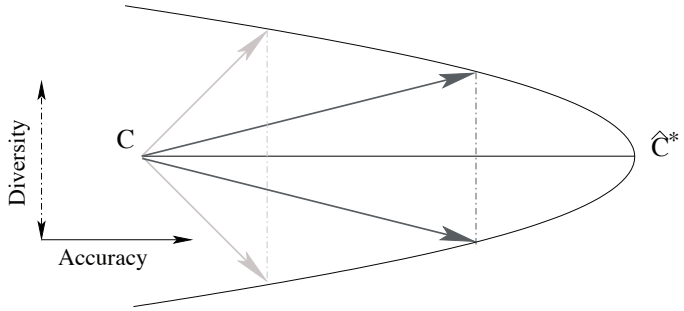


Figure 4.10: Graphical representation of Accuracy/Diversity dilemma.

accuracies are iteratively simulated. We report in figure 4.9 the similarity between output labels $I(C_1;C_2)$ for each trial as a function of the individual accuracy $\frac{I(C;C_1)+I(C;C_2)}{2}$.

A simple possible modelling of the relationship is to approximate similarity by a quadratic function of the average individual accuracy. Figure 4.10 gives a graphical interpretation of this approximation. A classifier is represented by a vector. Its projection onto the horizontal axis measures its individual accuracy while the difference between vertical projections of two vectors measures the diversity between them. The dash line represents the maximal diversity allowed between two classifiers with identical accuracy. This fits with the remark that two poor classifiers can have large diversity while two accurate classifiers cannot be so diverse.

This model can be extended to K classifiers. In the following, we will consider two terms based on the mutual information between classifiers, one measuring average accuracy, the other measuring diversity.

Definition 4.7. *The average accuracy of the K classifiers called Information Theoretic Accuracy (ITA):*

$$ITA = \frac{\sum_{i=1}^{K} I(C; C_i)}{K}. \tag{4.33}$$

Definition 4.8. *The average diversity between the classifiers is called Information Theoretic Diversity (ITD):*

$$ITD = \frac{\binom{K}{2}}{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} I(C_i; C_j)}. \tag{4.34}$$

In this work we propose to use a simple first order statistic to measure the individual accuracies and diversities, for two main motivations. On the first hand, the goal is to design a simple global score that can be used in various classifier combination applications. Using other statistics could increase significantly the computational costs of the score. On the second hand, Theorem 4.7 does not help us to discriminate beween ensembles having a large variance in the individual accuracies (thus large diversity) and ensembles having low variance between the individual accuracies and possibly less diversity. In this case we propose to keep the ensembles with high average accuracy even if diversity between them is penalized. It avoids the limitations of diversity-based techniques presented in section 4.2.2.

In order to design a score that relects Theorem 4.7, we need to consider the second order modelling of the similarity between the classifiers and the average accuracy presented before. This relationship can be written as $ITA^2 \propto \frac{1}{ITD}$. In fact, the diversity term ITD already contains relevant information about the average individual accuracy. We thus propose to compensate this information by considering the following Information Theoretic Score (ITS) as a function of ITS and ITD:

Definition 4.9. *The Information Theoretic Score (ITS) of an ensemble of K classifiers combined by majority voting is defined by:*

$$ITS = (1 + ITA)^3(1 + ITD). \tag{4.35}$$

This score will tend to select the best ensemble of classifiers by only considering diversity when it becomes a relevant feature. Compared to standard diversity based techniques, it will penalize ensembles with low ITA and large ITD. This model is a choice and other similar modeling could be chosen. The next section tries to validate this definition in the context of overproduction and selection of classifiers.

4.5.2 Validation of the ITS

To evaluate the intrinsic behavior of the ITS, we first consider artificial classifier outputs. By generating random outputs we can explore the complete space of output labels. It presents the advantage of being completely independent of the process of feature selection and independent of the learning algorithm. We can thus perform an unbiased evaluation of the ITS. Let us consider the following simple experimental setup. We randomly generate output vectors for three classifiers. For each run, we measure the accuracy of the majority

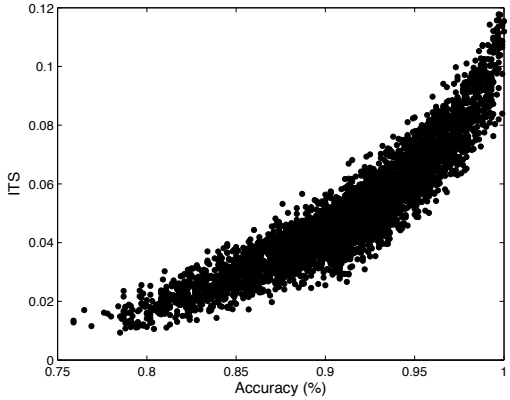


Figure 4.11: Score behavior with synthetic class labels

voting ensemble and the *ITS*. The results are shown in figure 4.11. Note that in this experiment we do not impose the individual accuracies to be identical, we only constraint them to fall between 0.5 and 1.

As expected, ensembles with high *ITS* are accurate. Moreover, an ensemble can be accurate but with a low *ITS*, therefore, the condition for maximizing $I(C;\hat{C})$ is sufficient but not necessary.

4.5.3 ITS in Multi-class problems

In section 4.5.1, *ITS* was defined according to empirical considerations based on binary classification purposes. However, we will show in this section that the *ITS* can also be used in multi-class problems. From a practical point of view, increasing the number of classes will increase the chances of having different outputs between the classifiers. This phenomenon is reflected by a higher diversity for a fixed individual accuracy. In order to check this multi-class behavior, we performed the same experiments as in section 4.5.2 with simulated output labels but with various number of classes from 2 to 6. Results are reported in figure 4.12. As expected, the accuracy/diversity representation still holds for multi-class problems, the diversity being an increasing function of the number of classes. The global relationship between *ITD* and *ITA* does not depend on the number of classes. Nevertheless, for very large number of classes, an adaptation of the diversity term can be imagined. In the experiments of this chapter, we will consider datasets with up to 10 classes.

4.5.4 Discussion

ITS fixes a trade-off between average individual accuracies and diversity. It can be used as a global measure of ensemble efficiency. However it presents some drawbacks. The main underlying hypothesis we made in the experiments for designing the new measure, is that the variance of individual accuracies in the ensemble is small. Clearly, if the individual accuracies in the ensemble are really balanced, approximating the individual accuracies by

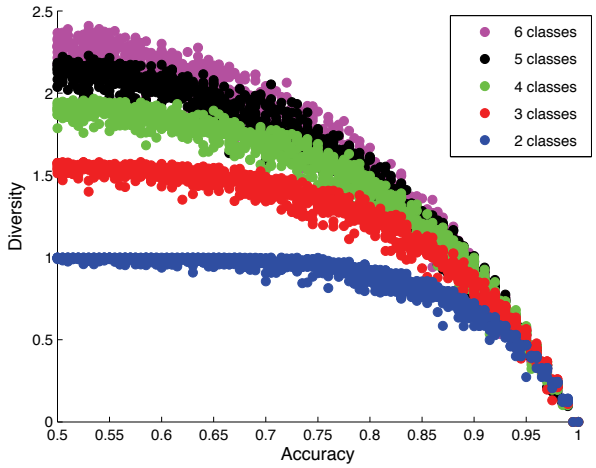


Figure 4.12: *ITS* in Multi-class problems.

their average becomes irrelevant. In this case, no global relationship between accuracy and diversity can be found. For example, combining two classifiers, one very accurate and one poor cannot be optimized by measuring diversity between them. However, in such cases, the contribution of *ITA* term in *ITS* will be more important than *ITD*, resulting in the selection of the best classifiers even if they are not diverse.

In fact, the general idea of *ITS* is to adapt the contributions of both terms *ITA* and *ITD* depending on the context. If the classifiers to be combined have almost identical individual accuracies, then the contribution of *ITD* in the *ITS* will be discriminant. It there exits more difference between individual accuracies, then *ITA* becomes more relevant and it then preferable to choose performant classifiers even if they have more redundancy between them.

The other drawback of this information theoretic framework is that the proposed criterion is not differentiable. It cannot be used directly as optimization criterion for building performant ensembles. There are several alternatives to tackle this limitation. On the one hand, we will propose to use *ITS* for controlling the performance of classifiers ensembles. This can be done for example in the context of overproduction and selection of classifiers. As we cannot maximize *ITS* analytically, we will also propose techniques for incrementally increasing the *ITS* of the ensemble. An important remark is that, we do not really need to find the best ensemble in the sense that is maximizes the *ITS*. As pointed out in Theorem 4.7, the conditions on the mutual informations between classifiers and true classes do not imply finding the best ensemble but means finding one of the best. To summarize, the procedures that will be proposed in the remaining of the chapter and in chapter 5 will concentrate on finding classifiers that are both diverse and individually accurate without forcing too much diversity between them.

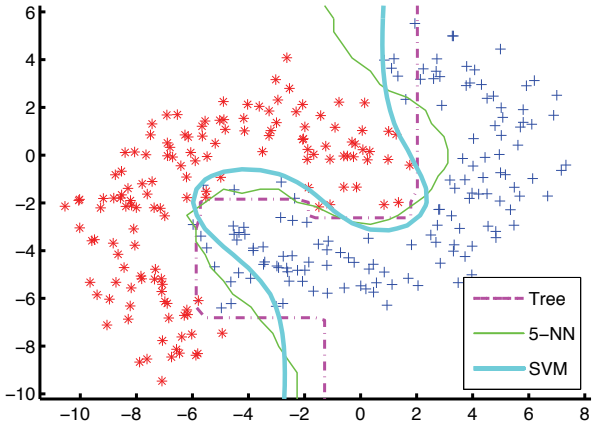


Figure 4.13: Example of Banana distribution. 3 decision functions a re also plotted: a decision tree, a SVM and a $k-NN$.

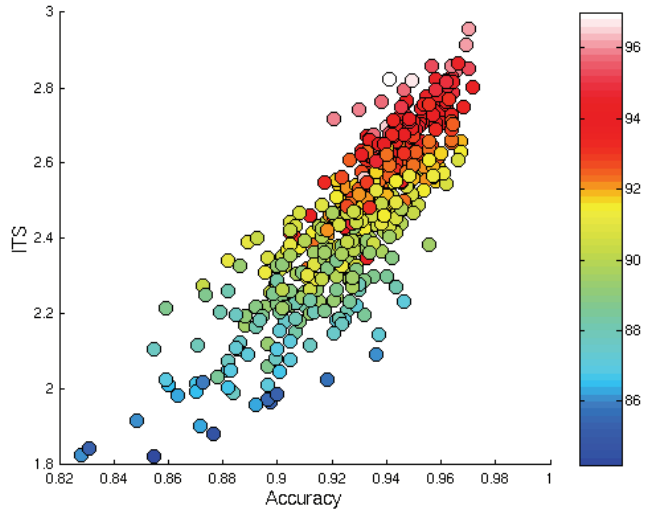
4.6 Application to Overproduction and Selection

4.6.1 A Simple 2-dimensional Binary Problem

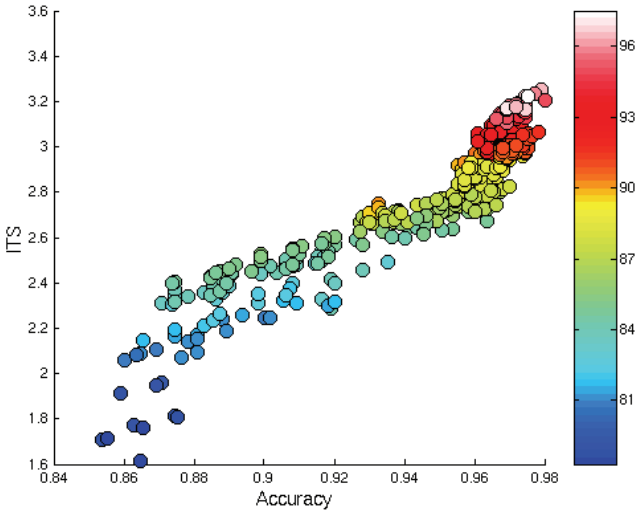
For evaluating the relevance of the ITS defined above on a real classification task, we first consider a 2 class toy problem using the Banana dataset available in the Matlab Pattern Recognition Toolbox [35]. An example of the data distributions is shown in figure 4.13. We generate 1000 training examples for both classes and we split this training set into 15 smaller subsets by random sampling. We then train one classifier with each subset. A first experiment (figure 4.14(a)) consists in training 15 Support Vector Machines (SVM) with 3rd order polynomial kernels (the C parameter being evaluated by cross-validation). The 455 possible combinations of three classifiers (called triplets in the following) are exhaustively tested. For each triplet, we measure the ITS on the training set, the ensemble accuracy on a large test set and we also compute the average individual accuracy of the three classifiers. This average accuracy is represented by the gray level of the circles in figure 4.14(a).

In the second experiment, three different learning algorithm are used. We trained 5 SVM, 5 linear classifiers and 5 K-nearest neighbors $k-NN$ and again ITS is measured for each triplet. Results are reported in figure 4.14(b).

As expected, the triplets of classifiers with low ITA (blue circles) lead to low ensemble classification accuracy. When the three individual classifiers are accurate individually (red and white circles in figure 4.14(a) and figure 4.14(b)), the final classification is generally accurate. However, in both configuration, the white points (which means the 3 best classifiers combined together) do not necessarily give the best combination. This phenomenon is more visible in the case of 15 SVM as they only have slight differences in their individual accuracies. In any case, the ensembles with high ITS are very accurate. These experiments



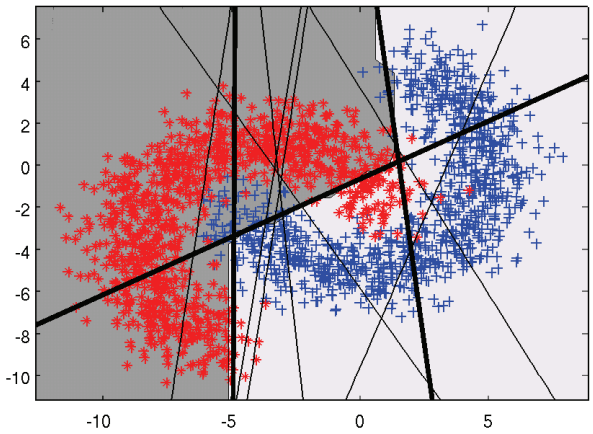
(a) 15 *SVM*



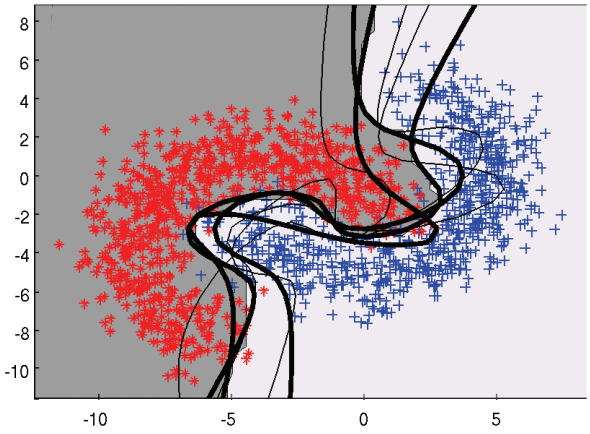
(b) 5 *SVM*, 5 $k-NN$, 5 linear classifiers

Figure 4.14: Combination accuracy and ITS for each triplet of classifiers. (a)15 *SVM* with *RBF* kernels and (b)5 *SVM* with *RBF* kernels, 5 $k-NN$ classifiers and 5 linear classifiers. The color of the circle is proportional the the average accuracy of the ensembles.

show that, at least in toy problems, the *ITS* can overcome the limitations of diversity as presented in section 4.2.



(a) Voting of 3 linear classifiers.



(b) Voting of 3 *SVM* polynomial, d=3.

Figure 4.15: Example of ensemble selection with *ITS*. Classifiers are generated on subsets of the complete training set. Bold lines represent the 3 selected candidates.

In figure 4.15(a) and figure 4.15(b), we show a graphical examples of classifier selection by *ITS*. We generated 10 linear classifiers in figure 4.15(a) and 10 *SVM* with 3rd order polynomial kernels in figure 4.15(b). The decision functions selected by maximal *ITS* are drawn in bold. The two class subspaces are represented by different gray backgrounds.

Dataset	# classes	# features	# examples	Best Classifier	ITS, K=3
BreastWC	2	33	198	39.43±0.05	32.21±0.02
Glass	6	9	214	36.91±0.04	33.74±0.05
Image	7	19	2310	23.14±0.01	22.60±0.02
Ionosphere	2	34	351	15.92±0.01	15.45±0.01
Iris	3	4	150	5.21±0.12	5.27±0.14
Pen	10	16	10992	4.16±0.03	3.78±0.04
Prima	2	8	768	33.37±0.01	32.16±0.01
Wine	3	13	178	29.87±0.02	27.61±0.03
Zoo	7	16	101	12.11±0.04	12.10±0.04

Table 4.1: Results on UCI datasets. Summary of the datasets used. Number of samples and dimensionality of the input space. We report error rates (in %) of the best single classifier and an ensemble of $K = 3$ classifiers created by maximizing ITS.

4.6.2 Real World Datasets

In this section we report experiments on real world datasets taken from the UCI Machine Learning repository [2]. The datasets cover a wide range of applications with number of classes between 2 and 10, with small sample size and large sample size cases. A summary of the datasets used is given in the three first columns of table 4.1.

In these experiments, we first trained a set of 15 decision trees on random subspaces of the training set. The choice of the base learner was motivated by the notion of classifier stability. As in Bagging (see section 3.4), unstable classifiers should be preferred in order to obtain performant ensembles. Unlike $k-NN$ for example, decision trees are known to be unstable classifiers.

For each single classifier we measured the error rates by cross-validation for small sample datasets or using a separate test set if available (for Image and Pen datasets). For each possible combination of K=3,5 and 7 classifiers, we measure the performance of the ensemble having the highest ITS. The sampling and training procedure has been repeated 10 times in order to obtain reliable classification statistics. In last column of table 4.1 we report the mean and standard deviation of the error rates concerning the best individual classifier and the statistics of the ensemble of 3 classifiers selected by ITS. It clearly shows that even a small ensemble of 3 classifiers compares favorably with the best individual classifier in most datasets.

In order to compare with a pure diversity based ensemble selection, we also extracted, at each run, the ensemble having the largest average QS. In other words, we compare the ITS measure with the ensemble the most diverse. Results are reported in table 4.2 for various numbers of classifiers: $K = 3$ and $K = 5$. The first comments concerns the QS selection. It some datasets, it really decreases performaces compared to the best individual classifier. This confirms the limitation of the pure diversity based techniques as described in section 4.2.2. On the second hand, ITS-based selection outperforms QS selection in most situations.

Dataset	Best	Q		ITS	
		K=3	K=7	K=3	K=7
BreastWC	39.43±0.05	38.32±0.09	34.12±0.12	32.21±0.02	31.76±0.09
Glass	36.91±0.04	35.42±0.05	33.32±0.02	33.74±0.05	29.71±0.04
Image	23.14±0.01	23.23±0.01	21.09±0.02	22.60±0.02	19.94±0.02
Ionosphere	15.92±0.01	14.68±0.06	14.23±0.01	15.45±0.01	13.72±0.01
Iris	5.21±0.12	25.67±0.12	20.22±0.11	5.27±0.14	5.05±0.09
Pen	4.16±0.03	5.13±0.07	4.36±0.03	3.78±0.04	3.20±0.04
Prima	33.37±0.01	32.87±0.03	32.12±0.05	32.16±0.01	31.56±0.01
Wine	29.87±0.02	30.52±0.02	28.28±0.04	27.61±0.03	27.03±0.04
Zoo	12.11±0.04	16.10±0.10	9.95±0.08	12.10±0.04	9.75±0.01

Table 4.2: Results on UCI datasets. Comparison of error rates of various methods : best individual classifier, selection by maximal *ITS* and selection by maximal QS.

Dataset	ITS		
	K=3	K=5	K=7
BreastWC	32.21±0.02	32.17±0.01	31.76±0.09
Glass	33.74±0.05	29.78±0.08	29.71±0.04
Image	22.60±0.02	20.92±0.03	19.94±0.02
Ionosphere	15.45±0.01	13.92±0.02	13.72±0.01
Iris	5.27±0.14	5.17±0.09	5.05±0.09
Pen	3.78±0.04	3.25±0.05	3.20±0.04
Prima	32.16±0.01	32.94±0.01	31.56±0.01
Wine	27.61±0.03	27.20±0.02	27.03±0.04
Zoo	12.10±0.04	9.81±0.00	9.75±0.01

Table 4.3: Results on UCI datasets. Influence of the number of classifiers in the ensemble. We report error rates (mean and standard deviation) for ensembles of K=3,5 and 7 classifiers.

Finally, table 4.3 shows the influence of number of members in the ensemble. As expected, increasing the number of classifiers in the ensembles increases the performances but in general, small ensembles already give significant improvements compared to the best individual member.

4.6.3 Discussion

The purpose of the experiments presented in previous section was to show that the *ITS* is a relevant measure of ensemble efficiency. It can tackle the limitations of pure diversity based selection methods. For this we tested all possible combinations of $K = 3, 5, 7$ classifiers in a pool of $M = 15$ classifiers. This means that for each experiment, we needed to test $\binom{M}{K}$. For instance, selecting 7 classifiers in a pool of 15 means testing 6435 ensembles. As many datasets require cross-validation techniques for estimating errors, *ITS* and QS, the exhaustive search is very computationally expensive.

Moreover, in practical applications, the number of classifiers in the pool (M) may be much larger that 15, and the number of classifier to select (K) is not known a priori. As in feature selection where we want to keep only the features that are discriminant and not redundant, selection of classifiers in a pool can be seen as selecting only classifiers that are accurate and, if possible, diverse. There are various sub-optimal alternatives to avoid the exhaustive search of the best classifiers, mainly using greedy algorithms.

For example, we propose to use the following selection procedure. First select the best individual classifier C_{1^*} :

$$C_{1^*} = \underset{C_i, i=1,...,M}{\arg\max}\ I(C_i, C). \tag{4.36}$$

Then, as we need an odd number of classifiers, we need to find two more classifiers maximizing the ITS between C_{1^*} and them:

$$(C_{2^*}, C_{3^*}) = \underset{(C_i,C_j), i,j \in \{1,...,M\} \setminus 1^*}{\arg\max} ITS(C_{1^*}, C_i, C_j). \tag{4.37}$$

This procedure can continue recursively until a given number of classifier is reached or until the improvements by adding 2 more classifiers becomes small enough. Once the first classifier has been selected, we need to extract 2 other classifiers from the $M-1$ remaining. Consequently, each iteration i of the procedure, we need to perform $\binom{M+1-2i}{2}$ tests in order to find the two new optimal classifiers. The total number of test for selecting K classifiers from M is:

$$N_{tests} = M + \sum_{i=1}^{\lceil \frac{K-1}{2} \rceil} \binom{M+1-2i}{2}, \tag{4.38}$$

which appears to be much lower than $\binom{M}{K}$ (except when K is very close to M, but in that case, classifier selection becomes useless.). An example is shown in figure 4.16. It shows the number of tests that need to be performed using either exhaustive search (red) or iterative selection (blue), for selecting classifiers from $M = 100$ classifiers.

4.7 Application to Adaboost

The procedure presented in previous section for iteratively selecting classifiers can be extended to a boosting strategy. In fact, we will propose an extended version of ITS that will consider weighted majority voting for combining decisions.

In AdaBoost, diversity is implicitly managed by the adaptive update of the sample distribution. Weights of misclassified examples is increased such that the next weak classifier will focus on these examples, thus introducing diversity with respect to previously selected weak classifiers. At this step, we could wonder how does the ITS evolves during AdaBoost learning. Intuitively the weak classifiers selected at each iteration should be both diverse (because of the sample distribution) and accurate (as they minimize weighted training error). We thus would expect a high ITS measure. An illustration is given in figure 4.17. At each iteration and for each candidate weak classifier, we measure an estimate of the ITS

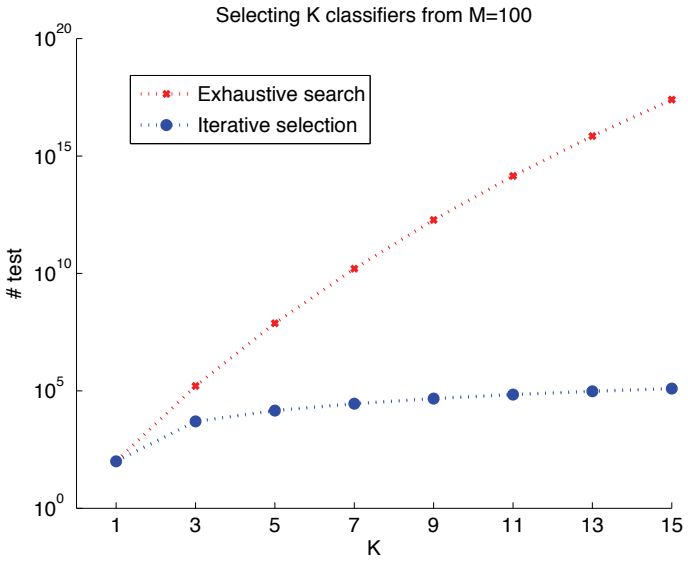


Figure 4.16: Number of tests that need to be performed for classifier selection using either exhaustive search (red) or iterative selection (blue).

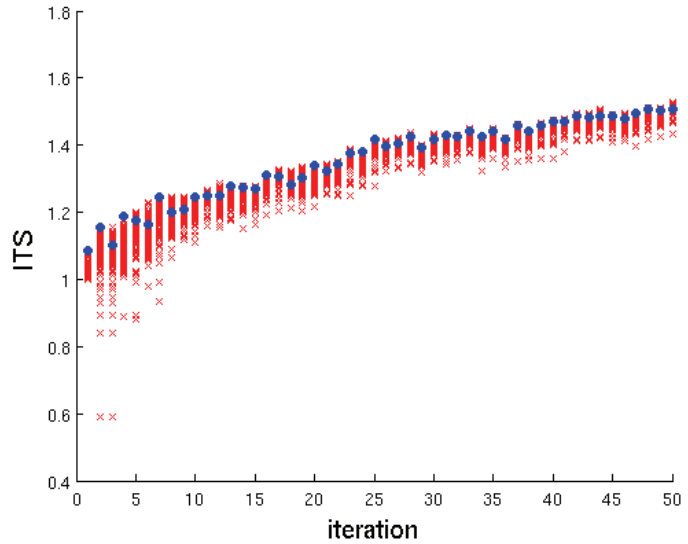


Figure 4.17: Comparison between AdaBoost and its modified version based on the *ITS* criterion.

(red crosses). The score of the selected weak classifier is represented by a blue dot. As expected, the selected weak classifier generally gives a high score.

Algorithm 4.1: ITS-Boost algorithm

Input: $Z_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, Number of iterations T

Initialize: $D_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$. $WITA(0) = 0$, $WITD(0) = 0$;

for $t = 1, \ldots, T$, **do**

Train weak learner with respect to the weighted sample set $\{Z_n, D^{(t)}\}$

foreach *weak classifier* h **do**

Compute the corresponding $\alpha_h(t)$.

Compute the ITS:

$$WITA_h(t) = WITA(t-1) + \alpha_h(t) I\left(Y; h(Y) | D^{(t)}\right)$$

$$WITD_h(t) = WITD(t-1) + \sum_{s=1}^{t-1} \frac{1}{2} \left(\alpha_h(t) + \alpha(s)\right) I\left(h(Y); h_s(Y) | D^{(t)}\right)$$

$$WITS_h(t) = \frac{(1 + \frac{1}{Na_t} WITA(t))^3}{1 + \frac{1}{Nd_t} WITD(t)},$$

where Na_t and Nd_t are two normalization constants:

$$Na_t = \sum_{s=1}^{t-1} \alpha(s) + \alpha_h(t),$$

$$Nd_t = \frac{1}{2} \left(\sum_{s=1}^{t-2} \sum_{r=s+1}^{t-1} \left(\alpha(s) + \alpha(r)\right) + \sum_{s=1}^{t-1} \alpha(s) + \alpha_h(t) \right).$$

end

Obtain hypothesis $h_t : \mathbb{R}^n \to \mathbb{R}$, that maximizes $WITS_h(t)$.

Update the corresponding values of $WITA(t)$ and $WITD(t)$.

Choose optimal $\alpha_t \in \mathbb{R}$

Update the weights:

$$D_i^{(t+1)} = \frac{D_i^{(t)} \mathcal{L}(\alpha_t, y_i h_t(\mathbf{x}_i))}{N_t},$$

where N_t is a normalization constant such that $\sum_{i=1}^n D_i^{(t+1)} = 1$.

end

Output: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x})\right)$

As explained in section 3.5.2, training in the case of AdaBoost comes to finding the weak classifiers and their corresponding weights. We naturally propose a modified version of it that selects weak classifiers that maximizes a weighted *ITS* (*WITS*) instead of picking the weak classifier that minimize the weighted training error. We call this new algorithm ITS-AdaBoost. The convergence properties of AdaBoost are not affected by the change as far as the weighted training error of each selected classifier remains at least slightly better

than random guessing. The detailed algorithm is given in Algorithm 4.1.

The *WITS* measure used in Algorithm 4.1 takes into account the distribution on the samples $D^{(t)}$ and the coefficients of the linear combination α_t. At each AdaBoost iteration we thus compute incrementally the weighted average accuracy (*WITA*) and weighted diversity scores (*WITD*). At iteration t, the *WITA* for each candidate weak classifier h is:

$$WITA(t) = \sum_{i=1}^{t-1} \alpha(i) I\left(Y; h_i(Y)|D^{(i)}\right) + \alpha_h(t) I\left(Y; h(Y)|D^{(t)}\right). \tag{4.39}$$

Diversity term is also incrementally computed by adding the diversites between each new pair of classifier. *WITD* is initialized with $WITD(0) = 0$ and then computed recursively by:

$$WITD(t) = WITD(t-1) + \sum_{s=1}^{t-1} \frac{1}{2} \left(\alpha_h(t) + \alpha(s)\right) I\left(h(Y); h_s(Y)|D^{(t)}\right). \tag{4.40}$$

Then the weighted score *WITS* is computed as in equation (4.35) but by considering weighted versions of each term:

$$WITS(t) = \frac{(1 + \frac{1}{Na_t} WITA(t))^3}{1 + \frac{1}{Nd_t} WITD(t)}. \tag{4.41}$$

Na_t represents the sum of linear weights at each iteration:

$$Na_t = \sum_{s=1}^{t-1} \alpha(s) + \alpha_h(t), \tag{4.42}$$

and Nd_t is the sum of the average α coefficients for each pair of classifiers:

$$Nd_t = \frac{1}{2} \left(\sum_{s=1}^{t-2} \sum_{r=s+1}^{t-1} \left(\alpha(s) + \alpha(r)\right) + \sum_{s=1}^{t-1} \alpha(s) + \alpha_h(t) \right). \tag{4.43}$$

As in the original version of AdaBoost, the *WITS* measure obtained in equation (4.41) will assign more weight to the classifiers having a large α_t. Note that the computation load is slightly increased with respect to AdaBoost as we need to compute the new *WITS* (t) for each candidate weak classifier. But the additional computations are generally negligible compared to the time spent for learning the weak classifiers.

We test this algorithm in several datasets. As usual, we consider several UCI datasets. The purpose of this section is to compare the proposed ITS-AdaBoost with AdaBoost. That is why the multi-class problems presented in table 4.1 are thus transformed into binary classification tasks. The merging of the classes is done so as to respect the balance between positive and negative classes as much as possible. On the second hand, we consider a face class modeling application. We used 19×19 pixels face and non face images (see [101]). From the gray intensity pixels, we extract 18 features by keeping 17 principal component

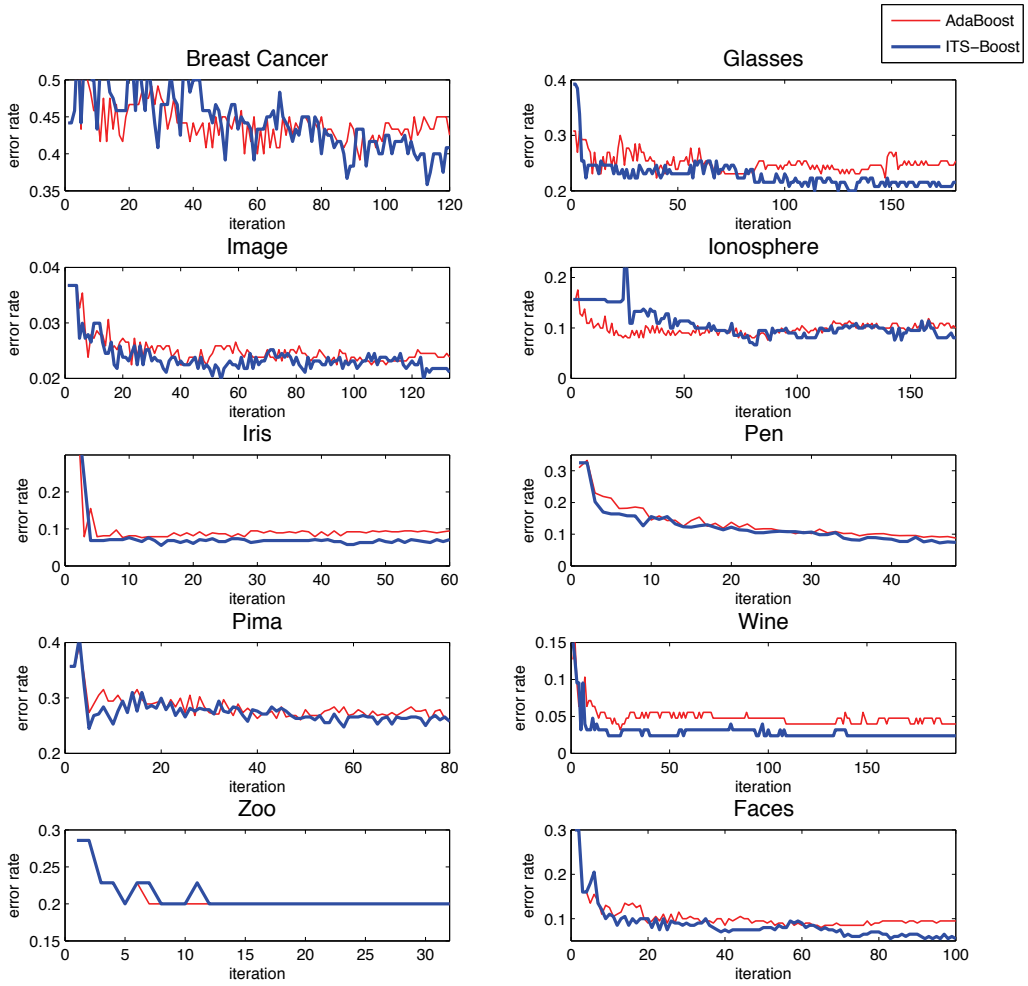


Figure 4.18: Comparison between AdaBoost and its modified version based on the *ITS* criterion. For each dataset, we report test error rates as function of the number of boosting iterations.

and adding the Distance from feature Space (DFFS). We considered slightly more than 3700 faces for training and roughly 4300 for testing. Non face images were selected by bootstrapping on randomly selected images. We used 5000 images for training and 10000 for testing. In this work, simple decision stumps (thresholding on each feature) are used as weak classifiers. Figure 4.18 gives a comparison between default AdaBoost and the ITS-AdaBoost on several UCI datasets. In most datasets, it turns out that the generalization is improved compared to AdaBoost (even if the training convergence is slower). This implementation basically gives a new possible interpretation of AdaBoost that explicitly takes into account a diversity measure between the weak classifiers that still considers a weight distribution over the samples.

4.8 Conclusions

This chapter presents a new ensemble learning framework using information theoretic concepts. It provides a tool for measuring the goodness of an ensemble by taking into account a trade-off between individual accuracy and diversity. A first possible use of this information theoretic criterion (called *ITS*) is in the context of overproduction and selection of classifiers. We show that *ITS* can help obtaining classifiers that are both diverse and accurate. We also propose a modification of AdaBoost that explicitly takes into account this diversity-based measure for iteratively building the ensemble. These techniques have been evaluated on sereval datasets and show improvements in terms of classification rates. In the next chapter, we will proposed techniques for obtaining ensembles of support vectors such that the *ITS* between *SVM* is maximized.

Ensembles of Support Vector Machines

5

In the previous chapter, we analyzed diversity in multiple classifier systems in an information theoretic framework. We defined the Information Theoretic Score (ITS) and we demonstrated how it can be used for selecting classifiers in a predefined team of classifiers. We also proposed a modification of AdaBoost based on this score. This chapter will concentrate more deeply on the design of one particular category of ensemble, namely Multiple Support Vector Machines ($MSVMs$). We will propose to use classical combination techniques to create efficient combination of SVM. We will also investigate how ITS can be employed in iterative optimization problems for obtaining performant ensembles.

5.1 Introduction to Multiple SVM

Training a SVM means finding the hyperplane that maximizes the margin between the two classes. As explained in section 2.3.2, finding this hyperplane requires solving a quadratic optimization problem. Consequently, the main drawback of SVM is that the complexity of the training varies at least quadratically with the number of training examples, which becomes intractable in large scale problems. To overcome this difficulty, several studies proposed to decompose the learning task into several lower complexity problems by using Multiple SVM ($MSVMs$). The principle of $MSVMs$ is to train several parallel SVM on subsets of the complete training set. A candidate test example is first given to each SVM. Then, a combination rule takes the final decision using either classification labels returned by the parallel SVM or continuous confidence measures. The global architecture of a $MSVMs$ is shown in figure 5.1. There are two main components in the system: the algorithm for choosing the data used to train each SVM, and the aggregation rule.

There are basically two types of possible splitting strategies depending of the type of available data and the motivations for using $MSVMs$: The subsets can be obtained by

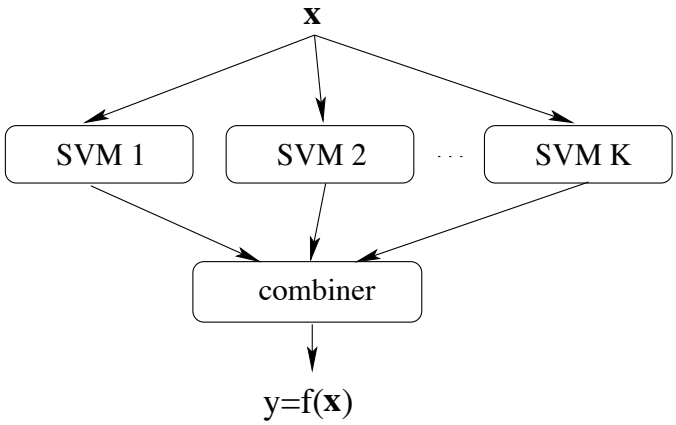


Figure 5.1: Multiple Support Vector Machines (*MSVMs*)

partitioning the feature space into smaller subpaces or by splitting the training examples into subsets.

- Feature subspace techniques are generally used to tackle problems with high dimensional data. In this case, the splitting can be viewed as a feature extraction step. Training local experts on local feature subspaces can be much easier than training one single *SVM* on the complete feature space. Generally the most correlated features are clustered together such that the learning algorithm can model efficiently the subspace and generally produce sparse models (i.e with few support vectors). The first technique referring to Mixtures of *SVM* (Kwok [86]) used this approach for improving classification performances of the system. The problem is that it does not reduce the complexity of large scale problems, in other words, problems with a large set of training patterns.

- The other alternative is to split the training examples into subsets. This directly reduces the complexity of the learning. The simplest strategy is to use disjoint subspaces obtained by random sampling. This technique was used for example by Collobert et al. [20]. The main goal is to simplify the learning task but it generally also improves classification skills by reducing the influence of noise and potential outliers in the training data. The data subsets can also be obtained by clustering [102]. The *SVM* obtained then act like local experts on their own domains.

The other main component of a *MSVMs* system is the choice of the combination rule for aggregating decisions from each parallel *SVM*. As in any multiple classifier system, the two main possibilities are fixed rules or trainable rules. The choice will be guided by the splitting strategy. Basically, systems that use clustering techniques for splitting the data (either in feature space or in the example set) will require trainable rules to give more weights to the experts having higher confidence levels. For random sampling techniques the difference between fixed or trainable rules is not that clear. It will be discuss throughout this chapter. In [20], Collobert et al proposed to use Neural Networks for combining the

decisions of the individual classifiers. The neural network gater was trained in order to minimize a squared error cost function. We will discuss in section 5.2 and section 5.3 other possible aggregation techniques.

MSVMs, in their random sampling version, are similar to Bagging as introduced in [10] but differ in several points. First, the sampling is done without replacement, thus generating replicates smaller than the initial dataset. Then the aggregation of the outputs can be done in many different ways, not necessarily averaging like in Bagging. A variant of *MSVMs* has also been proposed for image retrieval using relevance feedback [159].

These studies on *MSVMs* lie on interesting theoretical foundations showing the interest of using *MSVMs* rather than one single *SVM* trained on the complete training set. An interesting work focusing on generalization bounds of such kernel machines ensembles is presented in [37]. Let us define the ensemble margin of an example (in contrast to the margin of a single SVM). For each point $(\mathbf{x}, y)$ we define its ensemble margin to be simply $yf(\mathbf{x})$, where f is the ensemble decision function (as depicted in figure 5.1). This is exactly the definition of margin in [143]. For any given $\delta > 0$ we define EE_δ to be the number of training points with ensemble margin $< \delta$ (empirical error with margin δ) . In, [37], Evgeniou et al. proposed an upper bound on the leave-one-out error of the ensemble. It is given by the following theorem:

Theorem 5.1. *The leave-one-out error $\mathcal{L}((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$ of M parallel SVM is upper bounded by:*

$$\mathcal{L}((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) \leq EE_1 + \frac{1}{M} \sum_{m=1}^{M} \frac{D_m^2}{\rho_m^2}, \tag{5.1}$$

where EE_1 is the margin empirical error with ensemble margin 1, D_m is the radius of the smallest sphere centered at the origin, in the feature space induced by m-th kernel, containing the support vectors of the m-th SVM, and ρ_m is the margin of m-th SVM.

In some cases, this bound is smaller that the bounds of single *SVM*. For example suppose that the *SVM* use most of their training points as support vectors, then clearly the D_m of each *SVM* in the ensemble is smaller than that of the single *SVM*. Moreover the margin of each individual *SVM* is expected to be larger than that of single *SVM*.

In the remaining of this chapter, we propose different strategies for designing *MSVMs*. We first present two simple extensions of the classical *MSVMs* using different combiners: A second layer *SVM* trained on the margins of the parallel *SVM* is presented in section 5.2, then simple probabilistic rules are proposed in section 5.3. Section 5.4, gives a first alternative to train *MSVMs* using *ITS* as optimization criterion. In this section, Genetic Algorithms are considered. In order to face the complexity of the Genetic Algorithm approach, we then propose a learning based on the so-called Kernel Adatron for training on-line *SVM*. The techniques proposed trains jointly parallel *SVM* by allowing some training patterns to be used by more than one classifier. In section 5.6 we give the experimental setup and main results for comparing various *MSVMs* techniques. Finally, in section 5.7 we study the case where only few training samples are available. In particular, we show that the *MSVMs* architecture presented above looses its two main advantages in small sample

scenarios: reducing the complexity and improving generalization. Nevertheless, we propose an alternative that takes advantage of the cross-validation techniques usually used as model selection, by considering *ITS* between folds of the cross-validation. This will demonstrate that variant of *MSVMs* can also be used in small sample cases.

5.2 Second Layer SVM Trained on the Margins

In this section we propose a first aggregation rule that uses a second layer *SVM* for aggregating the decisions [102]. We call the K parallel *SVM* first layer *SVM*. Basically, the input space for the 2^{nd} layer *SVM* is the space of margins generated by the 1^{st} layer SVMs. We can represent the output of such a mixture of K experts as follows:

$$h_{\alpha_i,b}(\mathbf{x}) = \sum_{i,\alpha_i>0} y_i \alpha_i K\langle \mathbf{m}_i(\mathbf{x}_i), \mathbf{m}(\mathbf{x})\rangle + b, \tag{5.2}$$

where $\mathbf{m}(\mathbf{x})$ is the k-th dimensional vector of margins output by the K *SVM* in the first layer given the input $\mathbf{x}$.

Assuming that we want to train K *SVM* in the first layer, we will need $K+1$ training sets (an additional one is used to train the second layer *SVM*). We use two different approaches for generating the $K+1$ subsets. One consists of a random partitioning of the original training set. In the second strategy, we first randomly draw a sample that will be used for training the second layer and then we use a clustering algorithm, like k-Means [96] , for building the K subsets needed for training the first layer *SVM*.

In both cases we train each SVM-L1-i using a cross-validation process to select the best parameters then we use the $K+1$-th dataset for training the second layer *SVM* (SVM-L2): we let each of SVM-L1-i to classify the examples from this dataset and we take the margins output by the SVM-L1-i as input for SVM-L2. The margin can be seen as a measure of confidence in classifying an example, so, in some sense, the second layer *SVM* learns a non linear function that depends on the input vector and which assembles the confidences of each individual expert.

From a practical point of view, we have decomposed a problem of $O(n^2)$ complexity in $K+1$ problems of $O(\lceil \frac{n}{K+1} \rceil^2)$ complexity. As $n >> K$ this decomposition is clearly advantageous, and has the potential of being implemented in parallel, reducing even more the training time. Another issue that should be mentioned here is related to the robustness of the final classifier. In the case of a single *SVM*, if the training set contains outliers or some examples heavily affected by noise, its performance can be degraded. However, the chances of suffering from such examples are less important in the case of *MSVMs*.

5.3 Probabilistic Rules for Combining SVM

This section proposes a simple alternative for combining decisions of the first layer *SVM* [101]. We simply use probability rules as defined in [76]. The second layer *SVM* presented

in the previous section is trained on uncalibrated values of the margins. However in order to efficiently use the output of each *SVM* it would be interesting to have a posterior probability output. One way of transforming the margins in posterior probabilities consists in training directly a kernel classifier using maximum likelihood. A more appropriate method was proposed by Platt in [118]. He uses a sigmoid function to map the margins into probabilities. The advantage of this technique is that we directly obtain estimates of the posterior probabilities $P(y = +1|f)$ instead of estimating the class conditional densities. Equation (5.3) shows the form of such a sigmoid:

$$P(c = +1|f) = \frac{1}{1 + \exp(Af + B)}. \tag{5.3}$$

The parameters A and B are trained using maximum likelihood estimation from the training set, see [118] for details.

Then, the final decision is made accordingly to some combination rules based on the posterior probabilities estimated from the margins output by the *SVM* using equation (5.3). As presented in section 3.3.1, we can use the product, sum, min, max, mean median and majority voting rules.

The main advantage compared to *MSVMs* using a second layer *SVM* is that this strategy does not require any additional independent subset for training the combiner. However, estimating posterior probabilities from margin distribution may seem unnatural as *SVM* are known to be discriminative pattern recognition techniques (in contradiction to generative models, as discussed in chapter 2). Moreover the fitting of the sigmoid is specific to each *SVM* and comparing posterior estimates of parallel *SVM* can introduce a bias due to uncalibrated estimates.

5.4 Genetic Algorithms

5.4.1 Motivations

In the two first *MSVMs* techniques presented here above, the splitting of the training set is usually done by random sampling. The obtained subsets are completely disjoint. Consequently, this will lead to classifiers that are similar in the sense that they are trained from data that are just various estimates of the same distribution. Nevertheless, we saw in section 4.2 the importance of having diversity between classifiers (or at least moderate diversity). In this case, the individual *SVM* should have roughly equal accuracies. The contribution of *ITA* in *ITS* is expected to be high, while the diversity term *ITD* should be very small.

In the case of clustering for obtaining the subsets, the resulting classifiers would probably be very diverse, by acting like experts on their own local area. However, simple fixed combination rules would not be sufficient as each local classifier may behave badly on the whole space. To summarize, *ITD* should be very high, but the average individual accuracies *ITA* is probably poor.

It thus turns out that the problem of finding optimal subsets can be viewed as a possible application of the Information theoretic Score (*ITS*) defined in chapter 4. We thus propose techniques for finding subdivisions of the initial training set that will finally maximize our *ITS*. Let us recall that the *ITS* was defined in the context of majority voting combination, only this combiner will be considered in the following. If not specified otherwise, *MSVMs* will now concern multiple Support Vector Machines combined by majority voting.

5.4.2 Genetic Algorithms with ITS

A first brute force answer to this optimization problem is to use Genetic Algorithms (*GA*)[168]. *GA* have been found to be a robust and practical optimization method. A candidate solution is represented by a chromosome. A set of chromosomes (called population) is first generated randomly and it is then iteratively modified to converge towards the optimal. The relevance of each chromosome is measured using a fitness function. In our study, one chromosome represents one training set configuration for all the classifiers in the ensemble. It means that for each example, the chromosome encodes which classifiers will use this example in their training set. We use a coding based on a Venn diagram similar to the one proposed in [84] except that they work in the context of feature selection. The encoding is detailed in figure 5.2. For example, the code is 0 if none of the classifiers uses the example (should be obtained for very noisy examples or outliers), 1 if only classifier 1 uses it, 4 if only classifiers 1 and 2 use it and 7 if the three classifiers have this training sample in their training set.

The evolution of the population if performed by crossover and mutations for adapting training sets of the 3 classifiers.

5.4.3 Experiments

An illustration of the performance of *MSVMs* trained by *GA* (called *GASVMs*) is given in table 5.1. In this experiment, we consider the face dataset already used previously in this thesis and detailed in [101]. Error rates reported in table 5.1 correspond to equal error rates between face and non face classes. For several population sizes, we train the *GA* based on *ITS* and compare to a single *SVM* trained on the whole training set, and *MSVMs* as presented in section 5.3 with a simple mean rule as combiner.

With only few generations (e.g. 6 generations) the recognition rates are largely improved compared to simple random sampling introduced in [101].

This *GA* optimization guarantees to have an ensemble with large *ITS*. However the training process is very computationally expensive. In fact running the *GA* requires training $K \times p \times g$ classifiers, where p is the size of the population and g the number of generations. While it remains efficient with simple classifiers (e.g. k-nearest neighbors), it becomes practically infeasible with *SVM*. Another drawback of this *GA* strategy is that the training complexity varies dramatically with the number of classifiers in the ensemble. In this work, only a setup with 3 classifiers has been tested and extending it to larger ensembles can become quickly intractable. However this *GA* approach remains interesting for comparison

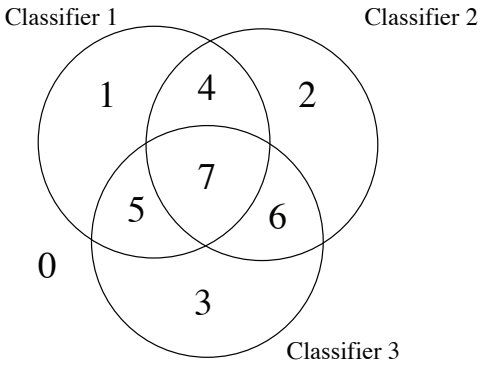


Figure 5.2: Chromosome encoding for *GA* optimization. Circles represents training samples that are used for learning each classifier. The code is 0 if none of the classifiers uses the example, 1 if only classifier 1 uses it, 4 if only classifiers 1 and 2 use it and 7 if the three classifiers have this training sample in their training set.

# generations	*GASVMs*	Single *SVM*	*MSVMs* (random sampling)
1	17.93(%)		
6	12.91(%)		
10	10.54(%)	20.86	18.72
16	9.23(%)	(%)	(%)
20	8.55(%)		

Table 5.1: Comparison of 3 classification techniques on face detection dataset. A single *SVM* trained on the complete train set, multiple *SVM* trained by random sampling (*MSVMs*) and multiple *SVM* obtained by *GA* using *ITS* as fitness function (*GASVMs*). For each classifier, classification error rates are reported.

purposes. It shows the classification rates that can be reached as well as the *ITS* measures obtained for good ensembles. In the following we propose a technique for achieving similar classification rates but with a much lower training complexity.

5.5 Kernel Adatron for Maximizing ITS

As pointed out in [150], directly generating ensembles that are both accurate and diverse is still a very challenging task. In previous section, we proposed a computationally expensive technique for achieving this goal. In this section we will detail an algorithm for easily building parallel *SVM* that are both diverse and accurate in the sense that their combination will maximize the *ITS*. In this work we use an on-line algorithm for training *SVM* which is called Kernel Adatron (*KA*) [46]. An overview as well as implementation considerations can be found in [148]. *KA* simply uses a gradient ascent to solve the convex quadratic optimization described in equation (2.11). The algorithm is described in Algorithm 5.1. In fact, the optimization differs from standard gradient ascent from two points:

- The coefficients α_i are changed directly without using temporary variables (this tech-

nique is known as stochastic gradient ascent);

- The learning rate η_i may be different for each training pattern.

There are various possible stopping criteria. The first is to monitor the Karush-Kuhn-Tucker conditions (see equation (2.21)) in order to guarantee that they stay statisfied. The other criterion is to stop when the increase of the dual objective function becomes small. One sufficient condition of convergence is given by the following bounds:

$$0 < \eta_i K(\mathbf{x}_i, \mathbf{x}_i) < 2, \tag{5.4}$$

that is why a good choice or η_i is $\eta_i = \frac{1}{K(\mathbf{x}_i, \mathbf{x}_i)}$. It becomes 1 in case of RBF kernels [148].

Algorithm 5.1: Kernel Adatron

1 **Initialize** $\forall i \in \{1, \ldots, n\}$,
$\alpha_i = 0$

2 **Calculate** $\forall i \in \{1, \ldots, n\}$,

- $z_i = \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$
- $\delta\alpha_i = \eta_i(1 - y_i z_i)$
- $\alpha_i = max(min(\alpha_i + \delta_i, C), 0)$

3 **Calculate new margins** $\forall i \in \{1, \ldots, n\}$,
$\gamma = \frac{1}{2}(\min_{\{i|y_i=+1, \alpha_i<C\}}(z_i) + \max_{\{i|y_i=-1, \alpha_i<C\}}(z_i))$

4 **Break if** $\forall i \in \{1, \ldots, n\}$, maximum number of iteration reached or the margin γ has approached 1

5.5.1 KA-MSVMs Algorithm

The general idea of this algorithm is to train jointly SVM such that, at each iteration, both ITA and ITD terms as increased. In the specific case of SVM, the decision functions are defined by the support vectors and their Lagrange coefficient. If all the classifiers have the same support vectors, then the decision functions will be very similar. In order to increase individual accuracies, different classifiers need to share some important support vectors, whereas, in order to add diversity, we need to reduce the number of common support vectors between al the classifiers.

We extend the standard KA algorithm to train jointly M SVM such that the ITS of the ensemble will be increased. The complete algorithm is described in Algorithm 5.2, where the index $x^{(c)}$ indicates that the variable x refers to the classifiers number c. The standard formulation of KA is found by setting $M = 1$, $\mu = 0$. The adaptation to multiple classifiers appears in the function f_{ITS} weighted by the factor μ. It depends on the output of all the current classifiers. Basically, the Lagrange coefficients of the support vectors that

are misclassified by the ensemble will be modified such that a majority of classifier classify correctly the support vectors. More precisely, let

$$f^{(m)} = \text{sign}\left(\sum_{j \in \text{sv}^{(\text{m})}} y_j \alpha_j^{(m)} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j)\right), \tag{5.5}$$

be the decision of the m-th *SVM* at the current iteration. Then for all the support vectors, we measure the number of parallel classifiers that correctly classify the example: if $\max\{\alpha^{(m)}\}_{m=1,\ldots,M} > 0$,

$$L = \sum_{m=1,\ldots,M} \mathbb{I}\left(f^{(m)}(\mathbf{x}_i) y_i > 0\right), \tag{5.6}$$

where $\mathbb{I}$ is an identity function such that $\mathbb{I}(true) = 1$ and $\mathbb{I}(false) = 0$. We then define the function that handles the joint term between the classifiers. Let us define $f_{ITS}^{(m)}$ by:

$$\begin{aligned} L^* &= \max\left(0, \lceil \tfrac{M}{2} \rceil + 1 - L\right) \\ f_{ITS}^{(m)} &= \begin{cases} +1 \quad \text{for the } L^* \text{ largest } \alpha_i^{(m)}, \\ \qquad \text{with } m \in \{1, \ldots, M | f^{(m)}(\mathbf{x}_i) y_i < 0\} \\ 0 \quad \text{otherwise} \end{cases} . \end{aligned} \tag{5.7}$$

The parameter μ is a weighting coefficient that affects the convergence speed. Empirical studies using crossvalidation showed that μ should be chosen in the range $0.5\eta_i \leq \mu \leq 1.5\eta_i$. Clearly setting a too large μ will tend to overfit the training data.

Algorithm 5.2: ITS - Kernel Adatron

1 **Initialize** $\forall m \in \{1, \ldots, M\}$, $\forall i \in \{1, \ldots, n\}$
$\alpha_i^{(m)} = 0$

2 **Calculate** $\forall m \in \{1, \ldots, M\}$, $\forall i \in \{1, \ldots, n\}$

- $z_i^{(m)} = \sum_{j=1}^{n} \alpha_j^{(m)} y_j K(\mathbf{x}_i, \mathbf{x}_j)$
- $\delta\alpha_i^{(m)} = \eta_i(1 - y_i z_i^{(m)}) + \mu f_{ITS}^{(m)}$
- $\alpha_i^{(m)} = max(min(\alpha_i^{(m)} + \delta_i^{(m)}, C), 0)$

3 **Calculate new margins** $\forall m \in \{1, \ldots, M\}$, $\forall i \in \{1, \ldots, n\}$
$\gamma^{(m)} = \frac{1}{2}(\min_{\{i|y_i=+1, \alpha_i<C\}}(z_i^{(m)}) + \max_{\{i|y_i=-1, \alpha_i<C\}}(z_i^{(m)}))$

4 **Break if** $\forall m \in \{1, \ldots, M\}$, maximum number of iteration reached or the margin $\gamma^{(m)}$ has approached 1

From the practical point of view, the update rule equation (5.7) tries to encourage support vectors that are misclassified by the ensemble at the current iteration to also become support vector of other *SVM* in the ensemble. This procedure could be seen as extending

the notion of support vectors to Ensemble Support Vectors (ESV). The ESV are those examples that are support vectors of at least $\lceil \frac{M}{2} \rceil + 1$ parallel SVM. Figure 5.3 shows a comparison between standard $MSVMs$ and the $KASVMs$. It gives a simple illustration of one single iteration of $KASVMs$. The 3 solid lines are linear decision functions obtained by $MSVMs$ with random sampling. The solid bold line is the ensemble decision function by voting. The support vectors misclassified by the ensemble (bold examples) are included into the training sets of other SVM and then become ESV. The ensemble decision function is updated such that all support vectors are correctly classified. In fact, this technique performs an implicit clustering of the data such that each member of the ensemble behaves like an expert with respect to the ensembles. This clustering is restricted to the only examples that are support vectors, it could thus be seen as a sparse clustering.

5.5.2 Experiments

In order to keep a fast convergence of the algorithm, we first perform few iterations of the standard KA before introducing f_{ITS}. This algorithm has been tested first on the face dataset presented previously in this thesis. We first ran 5 iterations of standard KA (by setting $\mu = 0$) for training 3 SVM on independent subsets, then μ is set to $\mu = \eta$. At each iteration, we measure the ITS of the ensemble on a separate test set and compare it to the best ITS obtained by GA. Results are depicted in figure 5.4. It clearly shows that the ITS increases significantly after the introduction of the joint term (5 iterations). It also shows that we quickly reach the ITS level of the computationally expensive GA technique. (The ITS level of the GA in figure 5.4 is a mean of 10 trials). Apart from its good classification skills, this algorithm presents also the advantage of being computationally friendly.

5.6 Comparison Study

The main goal of the experiments reported here is to investigate the behavior of the kernel Adatron adaptation of the $MSVMs$ and to compare it with the other standard techniques. $MSVMs$ are particularly effective in large scales applications that is why in our experiments we used the large large datasets of the UCI repository [2] presented in table 4.1. A first complete comparative study considers the largest available dataset: the Forest dataset. We transformed the original multi-class problem into a binary classification task where the goal was to discriminate class 2 from all the other six classes, this kind of partitioning making the two new classes of roughly the same size. We took 10000 examples of each class for training and 30000 for testing. SVM are trained using LIBSVM [18] by 5-fold cross-validation. We compare the following techniques: Single SVM trained on the complete dataset, Multiple SVM ($MSVMs$) [101], Gated SVM ($GSVMs$) [20], Genetic Algorithms using ITS as fitness function ($GASVMs$), 1 monolithic SVM trained using Kernel Adatron (KA 1 SVM), and finally 3 SVM trained jointly using KA ($KASVMs$). The results are reported in table 5.2.

The first comparison concerns a single SVM trained either by quadratic optimization or the kernel Adatron implementation. They perform quite identically in terms of error

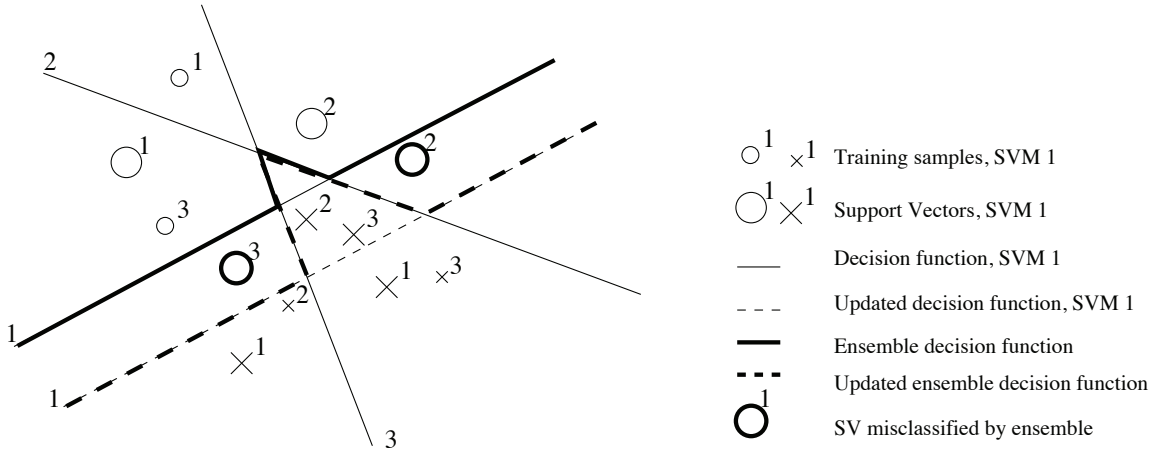


Figure 5.3: Illustration of the principle of Algorithm 5.2. The circles and crosses represent respectively positive and negative training samples. The number $\{1, 2, 3\}$ associated to each example means that the example belongs to training set of the corresponding SVM. The solid lines give the linear decision functions obtained by training 3 linear SVM using standard $MSVMs$ technique. The dashed line gives the modification of the decision functions after one iteration of $KASVMs$. The bold examples correspond to the ESV. The bold lines represent the ensemble decisions by voting.

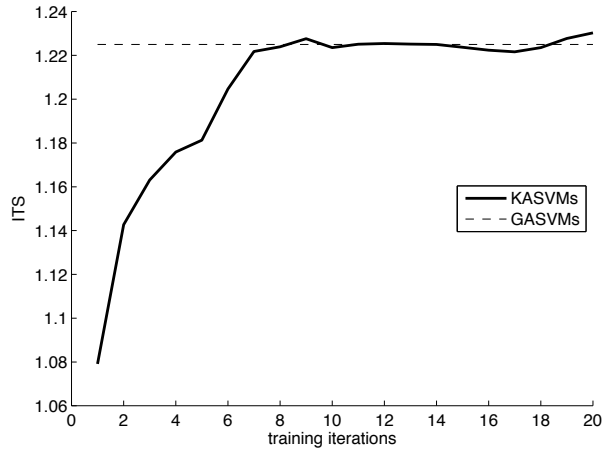


Figure 5.4: Comparison between ITS-Kernel Adatron ($KASVMs$) and Genetic Algorithms (GA)

rates but the training time is much lower in the KA case. Then we see that the techniques $MSVMs$ and $GSVMs$ improve the single classifier implementation as expected. $GASVMs$ and $KASVMs$ show how using a better subset selection than only random sampling, significantly improves the results. Finally the $KASVMs$ version converges much faster than $GASVMs$. We notice that the number of support vectors is very high for the $KASVMs$. This can be explained by the fact that contrarily to $MSVMs$, one input sample can be support vector for several of the 3 SVM at the mean time.

Experiment	#SV	Error rate (%)	Training time (min)
1 *SVM*	966	27.6	195
MSVMs [101]	633+628+644	26.73 ± 0.12	70
GSVMs [20]	143+127+176	26.67 ± 0.14	62
GASVMs	963+878+1035	24.98 ± 0.23	307
KA 1 *SVM* [46]	1331	27.23	19
KASVMs	1326+1345+1324	25.14 ± 0.12	23

Table 5.2: Comparison of various multiple *SVM* techniques on UCI Forest database [2]. For each technique we report the number of support vectors (#SV), the test error on a large test set and the training time in minutes.

Dataset	Forest	Face	Pen	Image
1 *SVM*	27.6% (966)	20.86% (344)	4.78% (205)	2.22% (616)
MSVMs	26.73% (1905)	18.72% (327)	4.52% (373)	2.04% (630)
KASVMs	25.14% (3995)	7.94% (361)	3.31% (386)	1.63% (650)

Table 5.3: Comparison of 3 classification techniques on several large scale datasets. A single *SVM* trained on the complete train set, mixtures of *SVM* trained by random sampling and mixtures of *SVM* obtained by KA-ITS. Error rates are reported for each classifier, as well as the total number of support vectors between parentheses.

In the following experiments, we compared various algorithms on other large scale datasets (i.e Pen and Image [2]), including the face dataset used in section 5.4. Three methods were considered: A single *SVM* (1 *SVM*), a combination of 3 *SVM* trained on random subsets (*MSVMs*) and 3 *SVM* combined by Kernel Adatron ITS (*KASVMs*). Classification results are reported in table 5.3. For each experiment in table 5.3, the number between parentheses represents the total number of support vectors involved in the classifier. In each case, the classification rate is improved using *KASVMs*.

5.7 MSVMs, Small Sample Case

In the previous sections, we presented several alternatives for training *MSVMs* such that the *ITS* of the ensemble is maximized. *MSVMs* presented in the previous sections are proposed for large scale problems. However, we will show in this section that *MSVMs* can also be used in small sample size problems but with a different objective. In fact the goal is not to reduce training complexity but to take benefit from cross validation techniques to improve generalization, following the idea of Freund et al [43]. As described in section 2.6, model selection in small sample size problems consists in a cross-validation on the training set. The parameter vector θ^* giving best cross validation estimate is used for training the final classifier on the whole training set. We propose here to keep the K best parameters,

best in the sense that the resulting ensemble maximizes the *ITS*. In fact during the cross-validation process used for model selection, *ITS* is measured between the cross-validation folds. At the end, the combination returning the highest *ITS* is kept. This technique follows the statistical motivation for combining classifiers as mentioned in section 3.2.

5.7.1 Experiments

In this section, we perform experiment to show-up the behavior of *MSVMs* architectures in small sample cases. First of all, simple experiments show that *MSVMs* as described above are not efficient in small samples cases for two main points. On the first hand, the gain in complexity observed in large scale problems disappears as single *SVM* does not suffer from large complexity due to large sample set. On the other hand, splitting the small subset into smaller disjoint sets introduces the risk of training very underfitted classifiers as the number of training patterns clearly becomes insufficient. The inversion of performance of *MSVMs* with the number of training patterns is represented in figure 5.5. For increasing number of training samples, we trained several time a *SVM* and 3 parallel *SVM*. A face/non-face dataset is considered as example. The inversion phenomenon appears around 60 training samples. By noticing that the dimensionality of the input patterns is $d = 18$, it shows that the *MSVMs* become useless for very small sample cases.

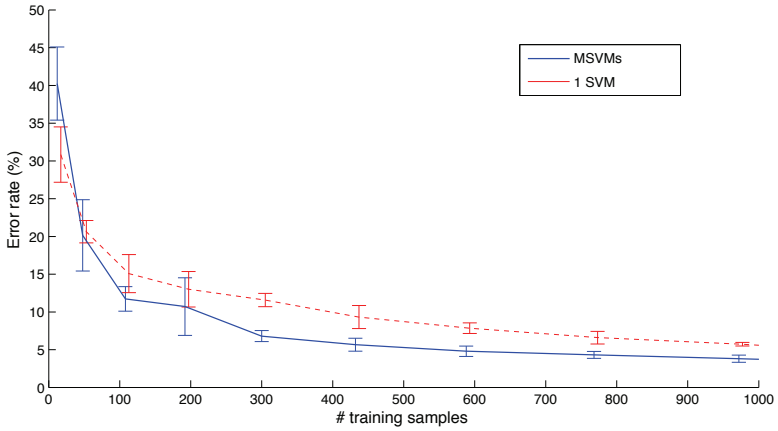


Figure 5.5: Comparison between *MSVMs* and Single *SVM* on face dataset. *MSVMs* as defined in section 5.2 are not efficient in very low sample cases.

The following experimental setup has been performed in order to evaluate multiple *SVM* in small sample cases. We consider the large scale datasets used in the previous section in order to keep a large test set for evaluating the performances of the cross-validation model selection. Once again, classes are merged into 2 classes in order to face binary classification tasks. If not defined explicitly in the UCI [2], we first randomly divide the data into one training and one testing set. For studying the classification performances as a function of the number of training patterns available, we extract training subsets of increasing cardinality. We repeat the subsampling process several times. For each training set, we perform cross-validation for *SVM* parameter selection. The results for each dataset

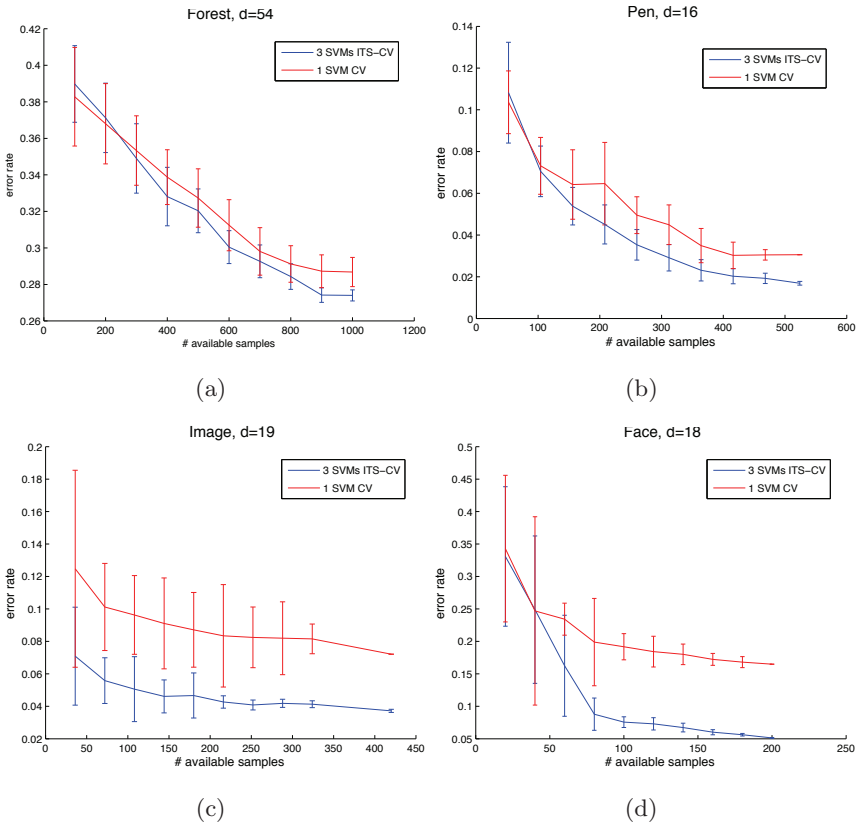


Figure 5.6: *MSVMs* in small sample cases.

are depicted in figure 5.6. According to the these results, the *ITS*-based combination becomes undeniably better than one single cross-validated *SVM* in terms of classification performance. However, we need to notice that the complexity is increased as for each possible combination of the model parameters, we need to compute the resulting *ITS*. The complete brute force search can be avoided by running one pre-selection by normal cross-validation and searching for the best combination of parameters in the neighborhood of the best parameters obtained from standard cross-validation.

5.8 Conclusions

Training a *SVM* can become very complex in large scale problems. This chapter discusses several alternatives for reducing the training complexity while, if possible, improving classification results. We study multiple Support Vector Machines (*SVM*) from two perspective: the combination rule and the spitting strategy used for obtaining subsets of the training data. In particular, we propose an algorithm based on Kernel-Adatron to jointly train parallel *SVM* such that the information theoretic score is maximized. Finally we also show how similar multiple *SVM* techniques can be employed successfully in small scale problems.

Part II

Applications

An Overview of Frontal Face Detection

6

6.1 Introduction

Automatic face detection has been one of the most active applications of computer vision and pattern recognition in the past years. It is commonly used as a main application to demonstrate performances of new object detection systems. Moreover, an increasing number of popular applications needs fast and accurate face detectors: face recognition, facial expression recognition, video conferencing, image indexing, etc. The performances of face detectors generally represents a limiting factor for the quality of the whole system [133].

The general problem of face detection can be described as automatically locating human faces in images or video sequences. The complexity of the task is due to many different sources of variation. Here is a short summary of the factor that may influence face appearance:

- Person-specific characteristics: facial appearance can change significantly according to the identity of the person, its gender, its age or the colour of its skin;
- Face is a deformable object: it can be deformed accordingly to the emotional state of the person (happiness, sadness, angry, fear, etc.), the actions of the person (speaking, reading, laughing, sleeping, crying). It can also be occluded by potential objects such as glasses or beard.
- External factors: head is a 3-D object in a scene. Therefore, the geometrical relationships between the head and the acquisition equipment will determine the head pose (if the face is frontal, profile, upright or rotated, etc.). Moreover, the type and quality of the sensor used to acquire the face image is also determinant. Finally, one of the

most important sources of variability is how sensitive face appearance is to lighting conditions.

All these sources of variability in face appearance demonstrate the complexity of the face detection task. A complete face detector should be robust to all these variations. However, in practice, many of these variations are constrained by the application. In particular, most of the systems that require a further face processing usually imply frontal or almost frontal faces. For example, in a face verification scenario, the person claiming an identity is supposed to be cooperative, which means that it can be asked to looks in a specific direction. The most important advances in face detection research have been carried out considering frontal or almost frontal face detection. Multi-view face detectors (i.e. detectors robust to head pose changes) are usually extrapolations of algorithms used for frontal purposes.

In this chapter, we give an overview of the most significant techniques for frontal face detection. Very complete face detection surveys are proposed by Hjelmas and Low [61] and Yang et al. [177]. A large variety of methods have been proposed for more than ten years, going from intuitive anthropometric models to very complex classification techniques based on recent advances in pattern recognition. Face detection techniques can basically be split into two groups: feature-based methods and image-based methods. Feature-based methods use explicit prior knowledge of faces (e.g. presence of eyes, nose and mouth, skin color models or geometrical structure of the face), while image-based techniques consider the problem as a binary pattern recognition problem (face versus non-face). Note that most of the techniques presented hereafter can be seen as hybrid methods between feature-based and image-based techniques, and there are many other possible ways to categorize all the techniques.

6.2 Feature-Based Methods

Feature-based techniques can be summed up into two main groups: features analysis and appearance models.

6.2.1 Feature Analysis

This section presents methods that use prior knowledge about facial features. Low-level analysis deals with direct segmentation of visual features while high-level analysis uses algorithms for automatic feature extraction.

Low-level analysis

One of the most primitive features used in computer vision is edge analysis. Historically, probably the first face detection system [139] tried to extract facial features based on their edges. They used pre-defined line-drawings around each facial feature. Face candidates were then selected using a simple correlation technique. This precursor work was later improved by imposing various constraints on the contours (e.g. curvature constraints [23]

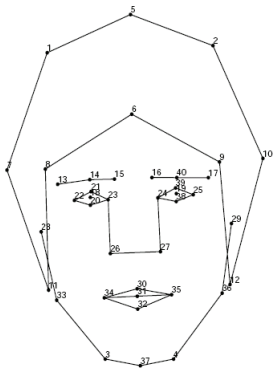


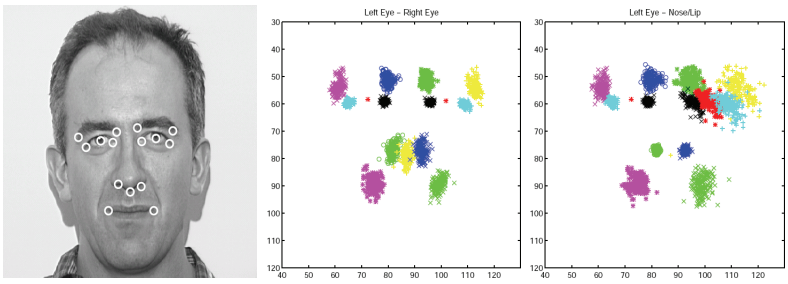
Figure 6.1: An example of pre-defined line-drawing model [24].

or feature labeling techniques [51]). An example of line-drawing model used in [24] is shown in figure 6.1.

Grey-level information is another basic feature that was early used for detecting faces. Faces present very specific contrast changes. Eyes regions are typically darker than the bridge of the nose. The eyebrows and the lips are also generally darker regions than cheeks and forehead. In order to take advantage of these gray-level intensity changes, the first step is to enhance the contrast between these regions, in order to improve the segmentation of the features. One of the most relevant technique using this strategy is the work of Yang and Huang [175]. They use a multi-resolution analysis to find the face candidates. Their study is based in the fact that humans are able to detect faces, even at low resolution. First, face candidates are selected by looking at uniform regions in low-resolution images. Then for each candidate, higher resolution scales are inspected in order to check the presence of the facial features.

Next logical step is to use skin color information which is a very natural and intuitive way of modeling faces. Many works tried to exploit this feature, mainly because of the low computation costs of this technique. Clearly, face color changes significantly between individuals and depending on illumination conditions. The first challenge is to find a good color space representation, as robust as possible to these changes. The most widespread color space is RGB. However, most of the skin colors changes are due to variations of luminance [176]. Therefore, a normalized version of RGB is usually preferred. Each channel is normalized by the sum of the three channels. Skin colors can thus be clustered efficiently in the obtained histogram. Many other color spaces have been proposed. The HSI color space used in [91] present the advantage of introducing a large variance between the skin and the facial components such as eyebrows or lips. It consequently becomes easier to segment the facial features. For a given color space, an efficient modeling of the skin-color regions is needed. This is usually done by simple thresholding but statistical modeling have proved to be particularlily efficient [109]. One of the drawbacks of this technique is that it requires an additional training process. In [72], Kakumanu et al. give a complete survey covering the most efficient skin color modeling techniques.

Finally, several other low-level features have been proposed. For example, Reisfeld et



(a) (b) (c)

Figure 6.2: The shape statistic technique [16]. (a) Facial features (b)Uncertainty in shape variables when the two eyes are references, (c) Uncertainty when left eye/nose/lips are references.

al. [129] proposed to take advantage of the symmetry of the face to efficiently discard background regions.

High-level analysis

The problem with all the low-level features presented above is that they directly rely on prior knowledge about face structure, but many background regions may also contain similar features. That is why high-level analysis have been carried out. The first kind of high-level analysis is called feature searching. It introduces confidence measures to the presence of prominent facial features. The work of De Silva et al. [152] is a good example of feature searching. They first look for top-of-the-head regions and then check the potential presence of eyebrows regions, etc. For the analysis of pre-defined facial components, they use various combinations of the low-level features presented in the previous section. The geometrical relationships between components are generally modeled using anthropometric measures.

Constellation techniques have also been proposed by Leung et al. [92]. The most prominent features points are selected in the image. Then, all the possible constellations of points are tested and the most face-like constellation is determined. Graph matching techniques are used for finding the best constellation. The nodes of the graph correspond to features on a face, and the arcs represent the distances between different features. An alternative to graph matching is to use statistical theory of shapes [16]. The shape statistic is a joint probability density function over the feature points. An illustration is given in figure 6.2.

6.2.2 Appearance Models

In the group of appearance model techniques, the presence of a face is determined by correlation measures between a pre-defined face template and the face candidate. The face model can be rigid (template matching), based on active contours or a deformable template.

Template Matching

In template matching, the face model is usually made of a combination of several local models, representing each facial feature (nose, mouth, eyes, etc.). The correlation can be computed based on low-level feature extraction [139] or geometrical considerations [23]. The popularity of these correlation-based techniques comes from their trivial implementation. However, they are not very robust to illumination changes or head pose variations. To overcome this problem, Sinha [154] proposed to use a small set of spatial image invariants to describe the space of face patterns. The key remark of this work is that , even if the brightness of the main face components (eyes, cheeks, forehead) are all affected by variations in illumination, their relative brightness remain almost unchanged. The face model can be defined as a collection of pairwise ratios of brightness.

Active Appearance Models

A natural improvement over the pre-defined rigid template is to use deformable templates. The first solution is to use active contours [73] (or snakes). Many solutions were proposed for using snakes to detect head contour [54, 87]. The snake is first initialized around the head boundary. The evolution of the snake is achieved by minimizing a global energy function $E = E_{internal} + E_{external}$. The internal energy term depends on the intrinsic properties of the snake, it means if it extends or shrinks. A typical $E_{internal}$ is to consider an elastic energy, which is directly proportional to the distance between control points of the countour. The external term depends on the image features that are considered. A simple solution is to make the external term sensitive to the gradient of the image [54]. It can also include skin color information [174] in order to attract the contour to the face region. The energy minimization is achieved by steepest gradient ascent or similar techniques.

Yuille et al [180] proposed to introduce a prior knowledge about the shape in order to improve the segmentation. The external energy is decomposed into four terms: valley, edge, peak and image brightness: $E = E_{internal} + E_v + E_e + E_p + E_b$. The main drawback of these techniques is that the evolution of the contour is very sensitive to the initialization. The computation costs are also pretty high because of the sequential optimization.

Finally another kind of active model was proposed by Cootes et al. in [21]: point distributed models (PDM). The countour of the PDM is discretized into a set of label points. First, a training set is used for modeling the variations of these points (see figure 6.3). The variations of the training set is modelled using Principal Component Analysis. Lanitis et al [90] showed that using this technique with face images produces a very compact parametrization. This overcomes one of the main drawbacks of the active appearance techniques: the computation costs. This also proved to be particularly efficient to partial face occlusions.

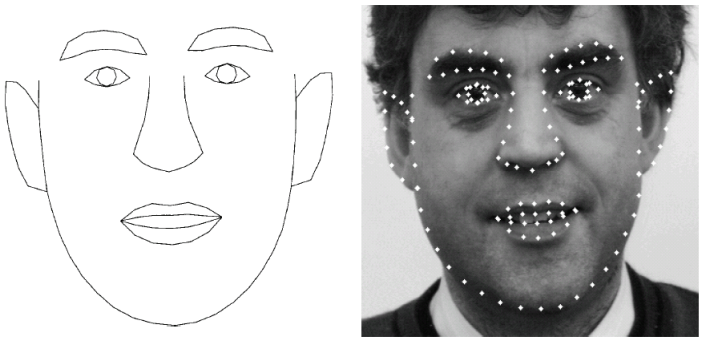


Figure 6.3: Point Distributed Models [90]. A mean shape model and locations of model points.

6.3 Example-Based Methods

All the face models presented hereabove were based on prior knowledge about face structure. The key features that were considered include: presence of specific facial features (eyes, nose, mouth, etc.), shape of face contour and specific face characteristics (e.g. color of the skin). This section reviews the techniques that consider face detection as a binary pattern recognition task. A common point in all the techniques that will be reviewed in this section is the strategy employed for searching faces in an image. In order to detect faces at any position, the input image is scanned with a sliding window. At each position, the window is classified as either face or non face. The method can be applied at different scales (and possibly different orientations) for detecting faces of various sizes (and orientations). Finally, after the whole search space has been explored, an arbitration technique may be employed for merging multiple detections. An efficient exploration of the search space is required for obtaining a fast face detector. Therefore, various methods have been proposed for speeding up this search. Usually, feature-based techniques are employed as pre-processing strategies for reducing the search space. A popular technique is to restrict the scanning process to regions containing skin-color pixels. However, the core component of such systems is the classification of each window as face or non face. Among the most efficient techniques, we will review the linear subspace methods and discriminant-based techniques.

6.3.1 Linear Subspace

The space of face images can be viewed as a subspace of the overall image space. Let us consider an image $\mathbf{x}$ with h rows and w columns. $\mathbf{x}$ is a n-dimensional vector with $n = w \times h$. The principle of the linear subspace methods is to project $\mathbf{x}$ onto a lower dimensionality space where face and non face examples can be efficiently separated. The two techniques commonly used are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

Eigenfaces

In 1987, Sirovich and Kirby [155] first proposed to employ *PCA* decomposition for efficient face representation. Face images can be represented by linear combination of a few eigenvectors corresponding to the largest eigenvalues. The technique is commonly referred as eigenfaces. For example, let us consider a 20×15 pixels face candidate. We can keep 85% of total variation by considering only between 15 and 20 principal components [102, 121]. This work was then used for joint face detection and recognition in [161]. All the training face images are projected onto the lower dimensionality space. They form the so-called face-space. The projection of new face images is supposed to be closer to the face-space than non face images. The presence of a face is thus evaluated from the Distance From Feature Space (*DFFS*), which corresponds to the reconstruction error (the Euclidean distance between the original vector and its projection).

This technique has been later further developed within a probabilistic framework [114]. They developed a maximum likelihood detector which takes into account both face-space and its orthogonal component. Samal et Iyengar [140] proposed to use face silhouettes instead of direct gray value pixel intensities. They generate eigensilhouettes by using several low-level features like edge detection, thresholding, etc.

Fisherfaces

The principal components used hereabove are obtained from a training set made of face images. Though this is particularly efficient for face image representation, it is not optimal in a classification scheme as it does not take into account non face training samples. An alternative is to use Linear Discriminant Analysis LDA. The principle is to find a projection that minimizes the within class variance (S_W) while maximizing the between class variance (S_B). The optimal projection is obtained by Fisher Criterion:

$$W_{fisher}^T = argmax \frac{|W^T S_B W|}{|W^T S_W W|}, \tag{6.1}$$

where W_{fisher}^T is the optimal projection matrix.

This technique was compared to *PCA* for face recognition in [5]. A simple example with 2 dimensional data projected on 1 dimensional line is given in figure 6.4. The advantage of LDA over *PCA* clearly appears. The projection is build in order to facilitate the classification between the classes.

Distribution-based

Several studies tried to use *PCA* or LDA as feature extraction step in order to improve image representation, and then focussing on better estimation of the face and non-face class distributions. Sung and Poggio [158] proposed one of the first reference face detection algorithm. They modeled each class in the eigenspace using 6 local clusters for each class. They obtained 12 multi-dimensional Gaussians. Finally a Muti-Layer Perceptron was trained to

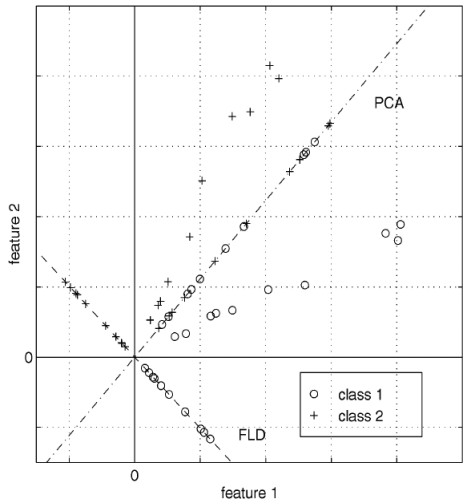


Figure 6.4: Comparison between *PCA* and *LDA* in a two dimensional problem. [5].

classify the input vectors. They also used an iterative bootstrapping procedure to generate a relevant set of non face images. At each iteration, the non face examples misclassified by the current classifier were added to the training set. The detector proposed by Schneiderman and Kanade [145] also models the probability distribution of the face class, but they employ a naive Bayes classifier. These techniques can be seen as a transition between linear subspace methods and the discriminant methods presented in the next section.

6.3.2 Discriminant Methods

In the last years, most of the work in frontal face detection has focused on discriminative techniques. They benefit from recent advances in the field of pattern recognition. Among the most popular techniques, we have neural network classifiers, support vector machines or boosting techniques.

Neural Networks

One of the most representative techniques for the class of neural network approaches is the work of Rowley et al. in [136]. It comprises two modules: a classification module which hypothesizes the presence of a face and a module for arbitrating multiple detections. The classification is done by an ensemble of neural networks. Each network was trained to catch spatial relationships of pixels.

A similar work was carried out by Féraud et al. [39]. Their associative neural networks were based on Constrained Generative Models (CGM). CGM perform a non-linear *PCA* to model the distance of a given input image to the facespace.

Figure 6.5: Component-based face detection [59]. a) Component templates b) Slight translation of the components c) Slight out-of-plane rotation of the head.

Support Vector Machines

Osuna et al. [111] proposed an efficient method for training *SVM* in face detection application. They used two huge sets of 10,000,000 test patterns to tune their models. In [59], Heisele et al. trained several *SVM* on local face regions into a component-based face detector. They obtained a total of 14 face components. The interest of this kind of techniques is their robustness to slight pose changes, as depicted in figure 6.5.

Sparse Network of Winnows

Yang et al. [178] introduced a Sparse Netwok of Winnows (SNoW) learning architecture. It is composed of a sparse network of linear functions in a feature space which is incrementally learned . They use boolean features that encode the positions and intensity of pixels. A comparative study between SNoW and SVMs is given in [179].

6.3.3 Boosting Methods

In 2001, Viola and Jones [165, 166] proposed an hybrid system that can be considered as the first real-time frontal face detector. Since then, most of the work in face detection has been concentrated on improving their system. The success of their work can be explained by three main contributions:

- They use AdaBoost for efficiently selecting and combining simple classifiers;
- They propose simple rectangular filters for building the weak classifiers. These filters can be computed very efficiently using the so-called integral image representation.
- The classifiers are implemented in a cascade structure, in order to decrease the processing time while improving the detection capabilities.

Many works then proposed to improve the system of Viola and Jones by introducing new filters or trying various variants of AdaBoost.

Filters for Building Weak Classifiers

The first detector based on Adaboost was proposed by Pavlovic and Garg [112]. They used directly gray intensity of pixels to build weak classifiers.

Viola and Jones introduced the rectangular Haar-like filters that can be computed very efficiently at any position and scale using the so-called integral image trick (see [25]). The

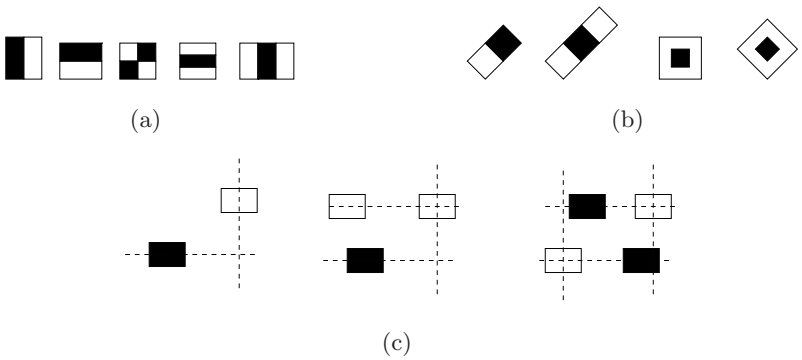


Figure 6.6: Three variants of the Haar-like filters used in Boosting techniques. a) Haar-like filters [165], b) Extended set with rotated filters [94], c) Extended set with non adjacent regions [93]

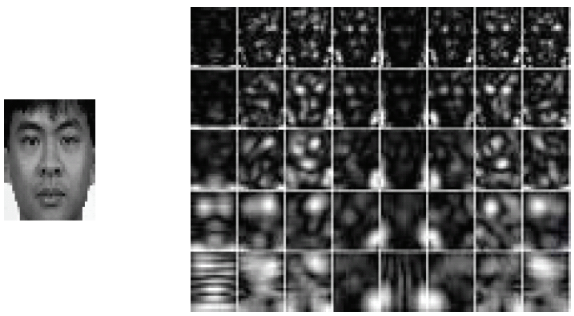


Figure 6.7: Gabor filters for frontal face detection [19].

rectangular templates are shown in figure 6.6(a). Then, Lienhart et al. [94] proposed to enrich the set of rectangular filters by allowing rotations by ± 45 degrees (see figure 6.6(b)). Li et al. [93] considered non-adjacent rectangles which present the advantage of proposing features that are not only local. These filters are depicted in figure 6.6(c).

In [19] Chen et al. proposed to use more discriminative Gabor filters in the last stages of the cascade. The collection of filters is described by various angles and frequency parameters. The reconstructed faces are called *Gaborfaces*. Examples are shown in figure 6.7.

Finally, another type of filters was recently successfully used in a similar boosting strategy: Local Binary Patterns (LBP) [108]. A LBP is a non-parametric 3×3 kernel which evaluates local textures in an image. It compares the grayscale values of a pixels and its 8 neighbors. The label 0 is set to all the neighbors with lower intensity than the considered pixel, and 1 otherwise. The labels are then encoded into an 8-bit word. The encoding of the LBP filter is shown in figure 6.8.

These filters have been extensively used in face verification purposes and have been recently used for face detection in [132]. LBP have very low computational costs and present the advantage of being more robust to local illumination changes than Haar filters.

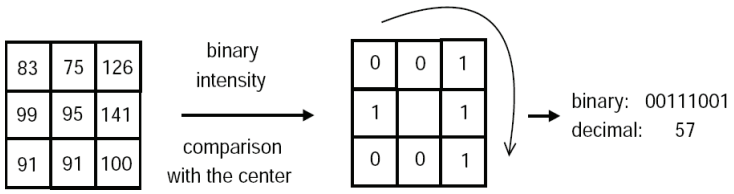


Figure 6.8: Local Binary Patterns proposed by Rodriguez et al. [132]

6.3.4 Variants of AdaBoost

Many different variants of Adaboost have been proposed to improve the discrete AdaBoost proposed in [165]. First, Viola and Jones [165] used simple decision stumps for building weak classifiers from Haar-like filters. Lienhart et al [94] proposed to use CART trees which allow to learn dependencies between the features. The training process becomes slightly more complex but less weak classifiers are needed for obtaining the same test error.

Li and Zhang [93] proposed a new algorithm called FloatBoost. They combine Adaboost with a feature selection strategy called Floating search. Contrary to AdaBoost, the worst filters are removed from the collection for the next iteration.

A generalized version of AdaBoost called RealAdaboost (or RealBoost is some papers) was applied to multi-view face detection in [173]. The output of the weak classifiers is no more limited to binary decisions (see section 3.5.4 for details).

Architecture of the System

Let us recall that the input image is scanned using a sliding window to check the presence of faces at any position and scale. However, only few windows effectively contain faces. This fact motivates the use of a cascade of classifier proposed in [165]. They first applied a very simple classifier made of 2 weak classifiers. The candidate windows classified as face are then given to a slightly more complex classifier. Finally only a few remaining candidates are given to the last stages which contain much more weak classifiers. It clearly decreases the processing speed by quickly discarding easy to classify background regions. It also presents another advantage. Each stage is trained on examples that have been misclassified by the previous stages in the cascade. This technique similar to bootstrapping improves the classification skills of the detector. The main drawback of this strategy is that many hyperparameters need to be manually tuned: the number of stages in the cascade, the number of weak classifiers selected in each stage, the thresholds of each stage, etc.

Luo [95] proposed a systematic method for tuning the thresholds of the stage after the training process. He showed improvements over the cascade of Viola [165] but it did not solve how to optimally design the cascade.

Wu et al. [173] proposed a nested cascade structure, where the first weak classifier of each stage is the complete previous stage. This is a possible way for taking into account confidence levels of each stage.

Finally Brubacker et al. [15] proposed a new technique for adjusting the threshold of

each stage and optimizing the number of weak classifiers in each stage. For this, they used probabilistic models of the overall cascade performance.

6.4 Performance Evaluation of Frontal Face Detectors

6.4.1 General Comparison of the Methods

In the previous sections we presented two main groups of approaches that have been proposed for face detection. Each technique presents advantages and disadvantages and the choice of the optimal algorithm mostly depends on the underlying application. In general, feature-based techniques are more robust to variations in head pose as the appearance of the main facial components (eyes, mouth, nose, etc.) is almost invariant to small rotations of the head. However, the problem of illumination changes is better addressed in example-based algorithms as the variations can be learned automatically during the training phase.

Feature-based techniques do not require intensive model training and are generally used in face localization applications. The task of face localization is to return the precise position of the face and its main components, while assuming that there is one and only one face in the image. Example-based techniques can be very fast and accurate for detecting many low-resolution frontal faces. They are thus generally used in video-surveillance applications or as pre-processing to face verification or further face analysis.

6.4.2 Quantitative Comparisons and Performance Evaluation

In order to give quantitative measures for comparing the techniques, algorithms need to be evaluated on standard datasets. A natural performance evaluation technique seems to simply count the number of correct detections and the number of false alarms. Using various sensibility thresholds, we can simply draw Receiver Operating Characteristic curves (ROC). The shape of ROC curves is usually used for comparing detectors. However, it does not necessarily gives fair comparisons of the systems.

- The output of typical face detectors is represented by window-boxes around the detected positions. But when is a detection considered as correct? If the whole face falls into the bounding box? If both eyes are present in the bounding box? If both eyes are present in the upper part of the bounding box? There is no common rules for stating how accurate the detected windows need to be in order to be counted as correct detections.

- Researchers need to pay attention to a problem that is common in many pattern recognition applications. We need to be precise if we aim to compare systems or algorithms. In the case of example-based techniques for face detections, comparing two systems means comparing two face detectors already trained. An independent test set is given to each detector and the best is the detector presenting best ROC curves. For comparing two face detector algorithms, we need to trained them with rigorously the same training data and then measure the performances on a separate test set. In

Figure 6.9: An example of image of the CMU test set presented in [136].

practice, people usually collect their own training and images. The comparison with other works is then limited to detector comparisons.

- The other difficulty in performance evaluation of face detection algorithms is that people have various definitions of the object to be detected. The first reference test set was introduced by Rowley et al. in [136], and Sung et al in [158]. The so-called CMU test set contains 130 images with a total of 507 labeled frontal faces. However, among the 507 faces labelled in this dataset, few dozens of them are manually drawn faces. Should they be considered as human faces or false detections? This ambiguity produced an important confusion in comparison of face detection techniques. An example of such ambiguous image is shown in figure figure 6.9. In this image, two faces are labeled. One is a human face, the other is a sketch on the whiteboard.

In order to address these evaluation ambiguities, Jesorsky et al. [69] introduced in 2001 a criterion based on the relative distance between the detected and the ground-truth position of the eye centers.

$$d = \frac{\max(d(E_l, \hat{E}_l), d(E_r, \hat{E}_r))}{d(E_l, E_r)}, \tag{6.2}$$

where E_l, E_r represent respectively the detected positions of the left and right eyes, and $\hat{E}_l, \hat{E}_r$ represent the true positions. A detection is consider as successfull if $d < 0.25$. In 2004, Popovici et al. [122] proposed an unified framework for performance evaluation of face detectors. The evaluation is performed by taking into account several parameters between the detected location and the annotated positions. They propose a scoring function that gives high score (1.0) to perfect detections and 0.0 to completely wrong detections. The scoring function is computed for various criteria: the ratio of the between-eyes distances, the angle between the eyes axis and of course the distance between the annotated and

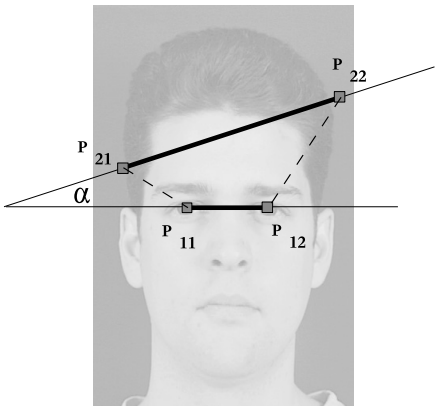


Figure 6.10: The criteria used in the performance evaluation proposed in [122].

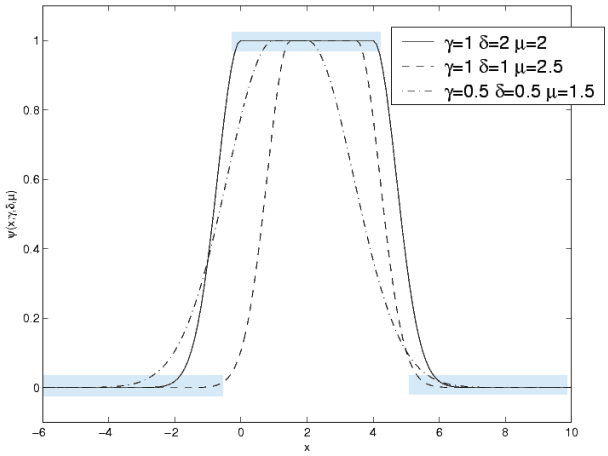


Figure 6.11: Shape of the scoring function [122] with several values of parameters α, δ and μ.

detected eye positions. These criteria are given in figure 6.10.

The scoring function can be parametrized specifically according to each application (only upright faces, face detection or localization,etc.). The shape of the scoring function is suggested in figure 6.11, where α, δ μ are 3 parameters of the scoring function.

6.5 Conclusions

In this chapter we presented the standard techniques employed in most of the recent face detectors. On the first hand we presented the feature-based techniques, robust to slight changes in head pose but very sensitive to changes in lighting conditions. On the other hand, example-based techniques can process very low resolution faces and can deal with relatively important changes in lighting conditions. These latter techniques are usually preferred when a further post-processing is needed (face recognition/verification, facial expression recognition, etc.). We also discussed the complexity of comparing performances

of different classifiers and gave the main evaluation protocols that have been proposed so far. In the next chapter e will present a new face detection system that benefits from the advances in pattern recognition techniques presented in chapter 2. We will also propose complete performance evaluation on some classical datasets using the evaluation protocol [122] presented hereabove.

Frontal Face Detection with Anisotropic Gaussian Filters

7

7.1 Introduction

In this chapter we present a new face detection system based on the work of Viola and Jones [165]. Their system uses cascades of boosted Haar-like features for classifying candidate images between face and non-face. Viola and Jones motivated their choice by underlying that Haar-like filters can capture contrast between face regions. Nevertheless, it turns out that these rectangular filters a not particularly well suited for discriminating between face images and background regions that present similar contrast characteristics than faces. Consequently, these challenging background windows will fall very close to the face class in the feature space. As we discussed in the introduction to pattern recognition in chapter 2, those examples that are close the separating decision function are very important for obtaining robust and sparse classifiers. That is why in this section, we will introduce new filters called anisotropic Gaussian filters that are more discriminative than Haar-like filters, especially for hard-to-classify examples.

The remaining of this chapter is structured as follows. Section 7.2 introduces the new geometrical filters and discusses their ability to model the face patterns. In Section 7.3 we report experiments that support the choice of these new filters, we give comparisons with relevant existing face detectors. We will also show that using these new filters do not affect significantly the processing speed of the detection system. Finally, we will draw some conclusions in section 7.4.

7.2 Boosted Anisotropic Gaussian Features

This section introduces the new geometrical features and discusses their ability to model face patterns compared to Haar-like filters. As in [165], AdaBoost is used to build a linear combination of simple hypotheses, which, combined together form a strong ensemble of classifiers. The weak learner called at each iteration of AdaBoost builds a simple binary hypothesis from the collection of filters. This basically means that AdaBoost performs an implicit feature extraction step coupled with the classification task.

In face detection applications, we usually prefer detecting all the faces even if the number of false alarms is increased compared to equal error rate. This can be easily implemented by considering an asymmetric version of AdaBoost that encourages the correct classification of the positive examples. There are basically two techniques for emphasizing on positive samples:

- Initialize differently the weight distribution D for the two classes: $D(i) = \frac{1}{N_+}$ if $y_i = 1$ and $D(i) = \frac{1}{N_-}$ if $y_i = -1$, with $N_+ < N_-$. However, this solution simply introduces a bias at the initialization step which does not guaranty a high true positive rate for the complete classifier.

- Add a threshold θ to the final decision of AdaBoost $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) + \theta\right)$ such that the desired operating point on the *ROC* curve is achieved on a separate validation set.

7.2.1 Anisotropic Gaussian filters

In this section we propose a new set of local filters that can be used for constructing the weak classifiers. The filters are made of a combination of a Gaussian in one direction and its first derivative in the orthogonal direction. These functions have been introduced by Peotta et al. in [115] for image compression and signal approximation.

The anisotropic filters are defined by the generating function $\phi(x, y) : \mathbb{R}^2 \to \mathbb{R}$ given by:

$$\phi(x, y) = x \exp(-|x| - y^2). \tag{7.1}$$

This generative function is depicted in figure 7.1. It efficiently captures contour singularities with a smooth low resolution function in the direction of the contour (y axis in figure 7.1) and it approximates the edge transition in the orthogonal direction (x axis in figure 7.1) with the first derivative of the Gaussian.

In order to generate a collection of local filters, the following transformations can be applied to the generating function:

1. Translation by (x_0, y_0):

$$\mathcal{T}_{x_0,y_0}\phi(x, y) = \phi(x - x_0, y - y_0).$$

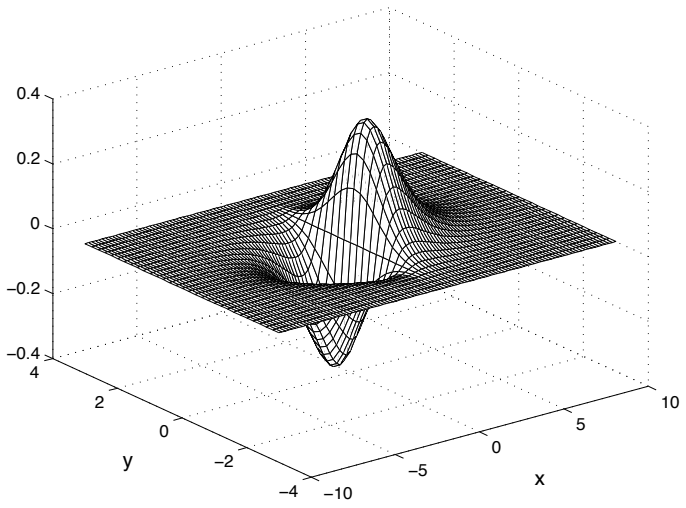


Figure 7.1: Shape of the Anisotropic Gaussian Filters. A Gaussian in one direction and its first derivative in the orthogonal direction.

2. Rotation by θ:

$$\mathcal{R}_\theta \phi(x, y) = \phi(x \cos\theta - y \sin\theta, x \sin\theta + y \cos\theta).$$

3. Bending by r:

$$\mathcal{B}_r \phi(x, y) = \begin{cases} \phi(r - \sqrt{(x-r)^2 + y^2}, r \arctan(\frac{y}{r-x})) & \text{if } x < r \\ \phi(r - |y|, x - r + r\frac{\pi}{2}) & \text{if } x \geq r \end{cases}$$

4. Anisotropic scaling by (s_x, s_y):

$$\mathcal{S}_{s_x, s_y} \phi(x, y) = \phi(\frac{x}{s_x}, \frac{y}{s_y}).$$

Bending is obtained by projecting the y component onto a circle of radius r as shown in figure 7.2.

By combining these four basic transformations, we obtain a large collection of functions that we denote $\mathcal{D}$:

$$\begin{aligned} \psi_i(x, y) &= \psi_{s_x, s_y, \theta, r, x_0, y_0}(x, y) && (7.2) \\ &= \mathcal{T}_{x_0, y_0} \mathcal{R}_\theta \mathcal{B}_r \mathcal{S}_{s_x, s_y} \phi(x, y). && (7.3) \end{aligned}$$

Figure 7.3 shows some of these functions with various bending and rotating parameters.

At each iteration of AdaBoost, a weak learner is called to train simple classifiers from the collection of geometrical filters. As described in section 6.3.3, many techniques can

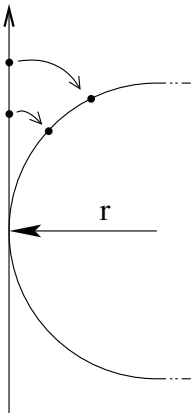


Figure 7.2: Bending by r. The filter is projected onto a circle of radius r such that the y axis is a vertical tangent to that circle.

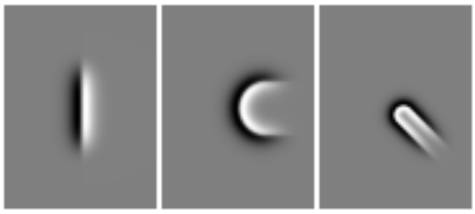


Figure 7.3: Anisotropic Gaussian filters with different rotating and bending parameters.

be used to train these weak classifiers. The simplest weak learner consists in learning a threshold and a parity for each filter response. More sophisticated constructions were proposed by Lienhart et al. in [94]. They proposed CART trees which allow to learn dependencies between the features. However, equivalent results can be obtained using the simple decision stumps by adding few more iterations in AdaBoost. Moreover, AdaBoost only requires classifiers that are slightly better than random guessing so this choice will not affect significantly the convergence of the learning.

We thus consider a simple linear classifier $h_j : \mathbb{R}^d \to \{-1, 1\}$ for each filter configuration by choosing two parameters: a threshold θ_j and a parity p_j as shown in equation (7.4). These parameters are chosen using the Bayes decision rule.

$$h_j(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad p_j f_j(\mathbf{x}) < p_j \theta_j \\ -1 & \text{otherwise} \end{cases} , \tag{7.4}$$

where the feature $f_j(\mathbf{x})$ is the scalar product between the image and the filter ψ_j corresponding to a particular filter configuration $(s_x, s_y, \theta, r, x_0, y_0)$:

$$\forall \psi_j \in \mathcal{D} \quad f_j(\mathbf{x}) = \iint_{X \times Y} \psi_j(x, y) I(x, y)\, dxdy. \tag{7.5}$$

Figure 7.4 shows some functions selected in the first iterations of AdaBoost. It turns out that they are particularly well adapted to capture local contours and are insensitive to

Figure 7.4: Some of the first selected base functions.

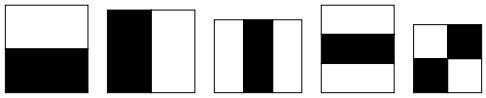


Figure 7.5: Haar-like templates.

changes of the lighting conditions. In comparison, Haar filters [165] model global contrasts that are more sensitive the the direction of the light source.

7.2.2 Gaussian vs. Haar-like

This section shows a comparison between the Haar-like filters (HF) proposed in [165] and the anisotropic Gaussian filters (GF) described above.

The HF are made of 2, 3 or 4 rectangular masks and have 4 parameters: horizontal and vertical scaling and the coordinates of the center. The templates are recalled in figure 7.5. A very important advantage of the HF over GF is that they can be computed extremely efficiently using a so-called integral image representation.

To give a quantitative comparison between HF and GF, two boosted classifiers have been trained on the same training set containing face and non-face images using AdaBoost. The results are evaluated on a large validation set extracted from some reference datasets (see section 7.3).

Figure 7.6 gives a comparison of the intrinsic performances of each feature type. The test error decreases quickly with the number of Adaboost iterations but it stops decreasing after roughly 100 iterations in the case of HF while it keeps decreasing with GF. Intuitively, after several iterations, AdaBoost focuses on the hard-to-classify examples and the simple HF are not discriminant enough to separate the two classes. In practice, after only few iterations of AdaBoost, the weighted training error of the best HF selected becomes close to 0.5. A Receiver Operating Characteristic (ROC) analysis (figure 7.7) clearly show the better discrimination capabilities of the GF compared to HF. The operating points used for drawing the ROC curves are found by changing the threshold of the final decision output by AdaBoost.

Let us now evaluate the time needed for computing each feature. We trained two similar classifiers with 200 HF on one hand and 200 GF on the other hand. By applying these two classifiers on several images, we compared the average computation time for applying a single HF and a single GF. We found that computing a GF takes in average 2.86 more time than a HF. (Note that the Gaussian filters are precomputed in the model such that the expensive computation of the generative function is avoided).

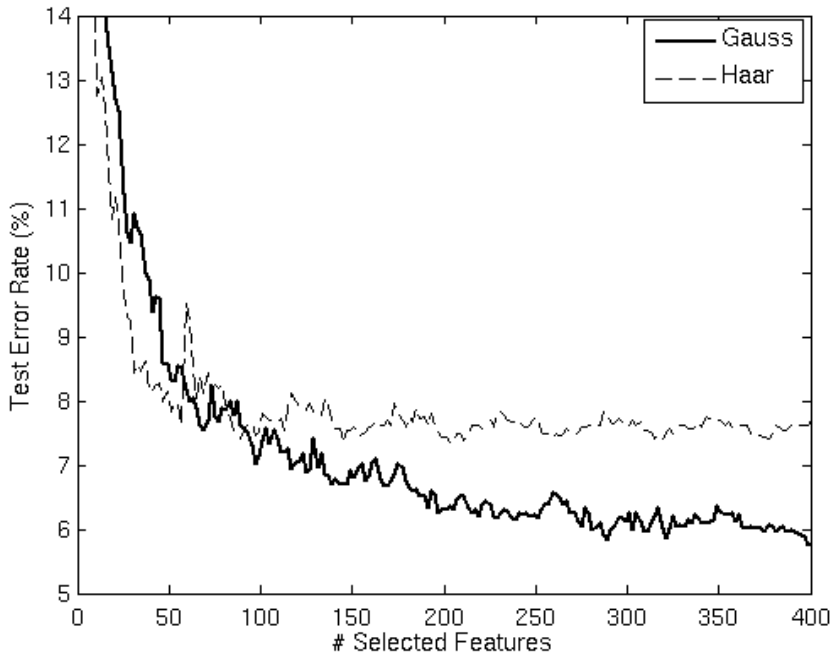


Figure 7.6: Performance of GF and HF–based detectors on a separate test set.

An intersting point to notice here is the shape of the GF that are selected by AdaBoost (figure 7.4). The first functions chosen have generally large scale parameters, they can globally model the face appearance whereas more local features are extracted later in the selection process.

HF are only binary filters, thus they may be able to well capture the contrast between image regions but will be limited for modeling smooth transitions present in facial images. GF are continuous functions more appropriate to model continuous natural images. The flexibility due to the large number of parameters allows to model contour singularities as well as intensity changes in large regions (using large scaling parameters).

7.2.3 Cascade of classifiers

As in [165], we use a cascade of classifiers speeding up the decision process by only applying simple classifiers to candidates easy to discard while keeping the most complicated and time consuming stages for the challenging examples. Moreover, training each stage of the cascade on different subsets of the training set reduces the risk of overtraining the data as described in section 3.5.2 by reducing the influence of potential outliers.

In figure 7.6 we can see that Haar-like features are comparatively efficient for building the first linear classifiers. Up to roughly 90 iterations, generalization curves of GF and HF are sensibly identical. The rectangular shaped filters are sufficient for discarding the simple examples. In order to take advantage of their computation efficiency, a simple five staged cascade of Haar-like features is added as a pre-processing to our final Gaussian-based classifier. This efficiently reduces the search space such that only few remaining windows

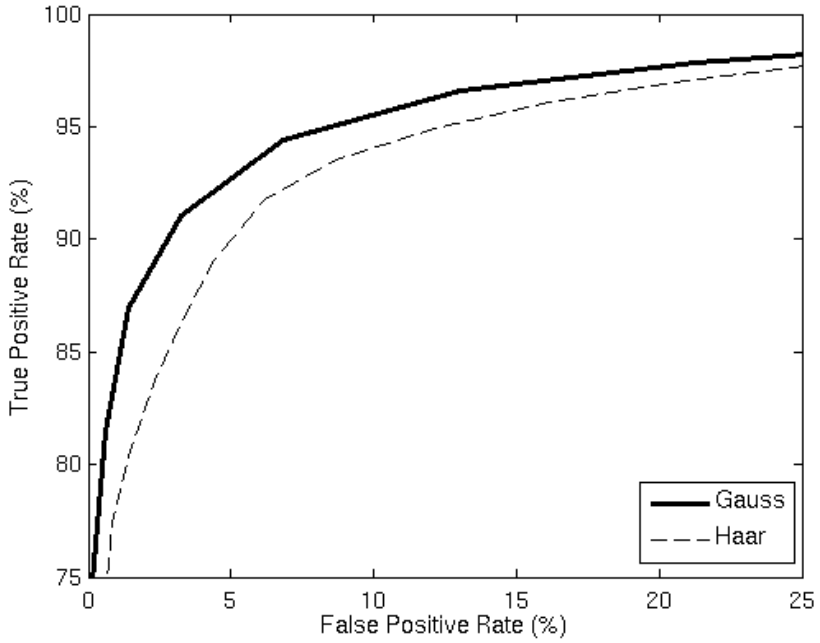


Figure 7.7: *ROC* curves for Gaussian and Haar-like features.

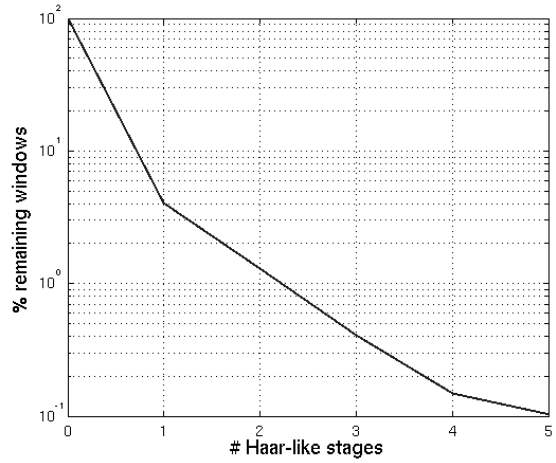


Figure 7.8: Reduction of the search space by a simple cascade of Haar features.

need to be tested with the Gaussian models. Figure 7.8 gives the average percentage of negative windows discarded by each of these 5 first stages on the CMU/MIT test set [136]. For example, after only 2 stages of HF, only 0.4% of negative windows have still not been rejected.

Figure 7.9: Images that are generated from one original image taken from the BioID dataset [47]. These transformations include shifts, slight in-plane rotations, scaling and horizontal flipping.

7.3 Experiments and Results

7.3.1 Structure of the System

In order to test the performances of this system and compare it with other relevant methods the following experiments have been performed.

Scanning Process

First, the size of the scanning window influences directly the quality of the detector [94]. According to previous empirical studies, we used 20×15 pixels window to scan the images. It is then dilated by powers of 1.2 in order to detect faces at any scale. Then, a very simple arbitration method clusters the neighbor positive windows such that only one detection per face is returned. We simply extract the mean window for each cluster (average position and scale).

Datasets

To train the models, face images were collected from some classical face datasets: XM2VTS [100], BioID [47], FERET [116]. In order to improve the robustness of the detector with respect to slight shifts or in-plane rotation, we increased the size of the training set by adding images extracted around the groundtruth positions. For each image, we also perform small linear transformations like slight rotations of flipping. Figure 7.9 gives examples of small transformations that are applied to each image.

The resulting complete face training set contained around 9500 images. The non face dataset was bootstrapped from randomly selected images (without human faces). A total of roughly 500000 non face images were finally used.

We also used a separate validation set made of roughly 10000 faces and 100000 non faces images in order to tune the various hyperparameters. For example the number of filters selected in each classifier is determined according to the desired detection rates on this validation set.

Choice of the HF and GF Filters

The collection of HF and GF needed for training the classifiers has to be defined. The task is to define the transformation parameters of the filters. These transformations would then

condition the size of the set $\mathcal{D}$ of filters and, consequently, determine the training complexity (the cardinality of $\mathcal{D}$ can be seen as the dimensionality of the feature space). The set of HF that we used to train the cascade contained 37 520 filters (all possible combinations in a 20×15 pixels window). In the case of GF, considering all the possible combinations of the 6 filter parameters would produce a huge set of filters. We thus decided to subsample the complete set of filters in order to keep a manageable set for the training process. For this, we noticed that slight variations of the 2 scaling parameters and the rotation parameter do not change significantly the response of the filter. The other 3 parameters are more unstable in the sense that small changes may have large effects on the response of the filter. The parameter space has been subsampled according to these previous remarks. The final set of GF that we used for training the classifiers contains 202 200 features. This set of GF gives a good trade-off between training complexity and parameter flexibility.

Evaluation Protocol

An ambiguous point in face detection algorithms is the way the performances are measured. Papers usually provide the detection rate and the false positive rate to show the quality of their system, however they often consider different measures for those rates. It appears to be very difficult to objectively compare different published results. In this work, the problem is addressed using the evaluation protocol proposed by Popovici et al. in [122]. The evaluation is performed by taking into account several parameters between the detected location and the ground-truth annotated positions. The scoring function measures the ratio of the between-eyes distances, the angle between the eyes axis and of course the distance between the annotated and detected eye positions. This method gives a more objective scoring of the detection performances. See [122] for details on how to use the scoring function.

7.3.2 Evaluation on BANCA Database

The system has been tested on two distinct datasets. On the one hand we considered the BANCA database [4] which was built for training and testing multi-modal verification systems. The face images were acquired using various cameras and under several scenarios (controlled, degraded and adverse). Some examples of detection results of the so-called adverse scenario are shown in figure 7.10. In this work, we used 12 480 images from the so-called French and English datasets as we dispose of precise groundtruth annotations for these ones.

Table 7.1 gives a comparison of four variants of the system tested on the BANCA database. We first tested 5 stages of HF to evaluate the efficiency of the few first features selected. Then a complete cascade of 12 stages of HF is compared to the same structure using GF. Finally we tested a GF cascade preprocessed by the 5 first stages of a HF cascade. First of all it confirms that a cascade trained with GF gives much better detection rates than using HF. It is then interesting to notice that a very simple pre-processing by a 5 staged cascade of HF followed by a Gaussian based model does not affect significantly the

Figure 7.10: Results on images of BANCA [4] in the complex adverse scenario.

Table 7.1: Comparisons of various methods tested on the BANCA [4] database. Results are reported for the French and English parts following the evaluation protocol described in [122]. Detections with a global score larger than 0.95 are considered as correct.

Classifier	% of detections with score > 0.95 (following [122])
5 stages Boosted HF	52.08
12 stages Boosted HF	86.78
12 stages Boosted GF	91.02
5 st. Boosted HF + 12 st. Boosted GF	90.74

performances compared to GF alone. We thus can benefit from the computation efficiency of the HF without decreasing the detection rates and then take advantages of the GF discrimination to improve the classifier accuracy.

The evaluation protocol [122] allows to measure the main characteristics of our detector in terms of precision of localization. Each individual criterion in figure 7.11 shows that when a face is correctly detected, the bounding box returned by the detector is really precise both in scale and shift. However, there is a bit more imprecision with respect to the shift score. This can be explained by two main factors:

- The arbitration criterion that we use for merging the multiple detections around each face very simple;
- The large variability that we voluntarily introduced in the face training set might cause some imprecision in the detected locations.

Let us recall here that the purpose of this system is to detect faces in images which is different to face localization. Face localization usually suppose the presence of one and only one face in the input image. The task is then to return the precise positions of this face.

7.3.3 Evaluation on the CMU/MIT Test Set

We now consider a more challenging database commonly used to evaluate performances of face detectors especially on very low resolution faces. The CMU/MIT Test set [136] was first introduced by Rowley in [136] for testing a Neural Networks based detector. The first version of this test set contained 23 images with a total of 155 very low resolution faces (it

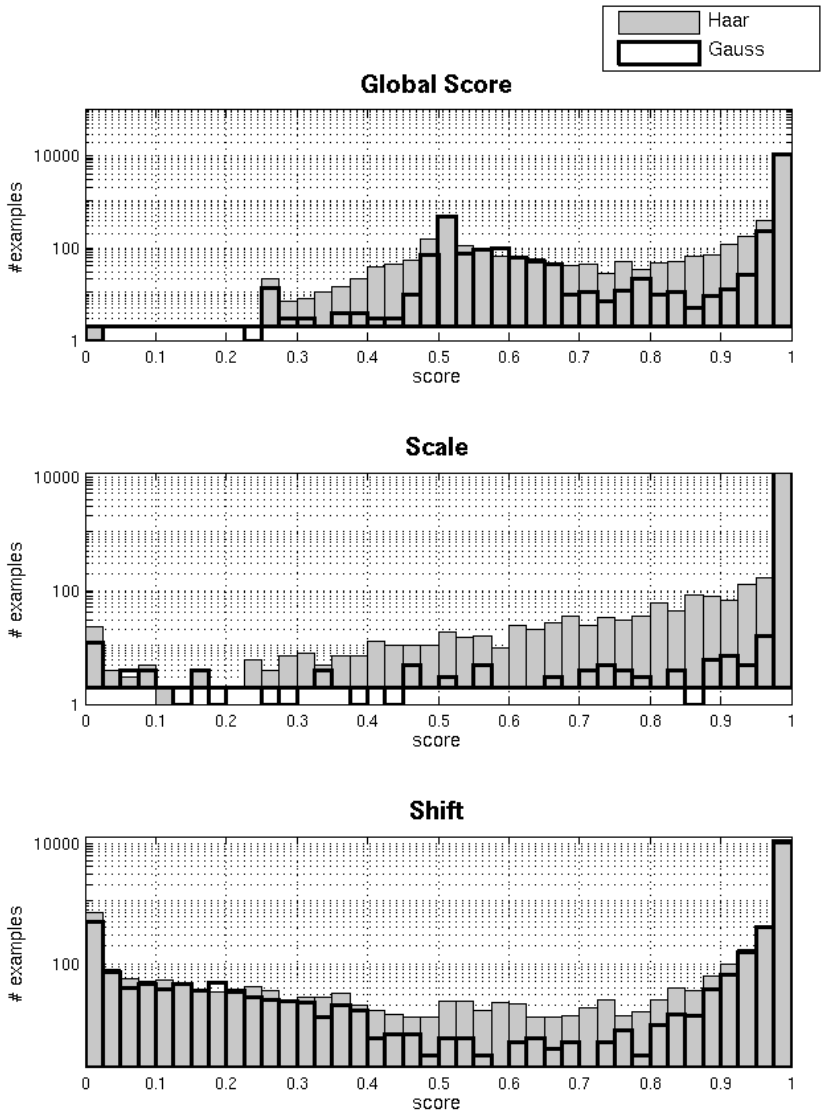


Figure 7.11: Detection scores using the evaluation protocol [122] including the two individual scores (shift and scale) and the global score. Note that a logarithmic scale is used.

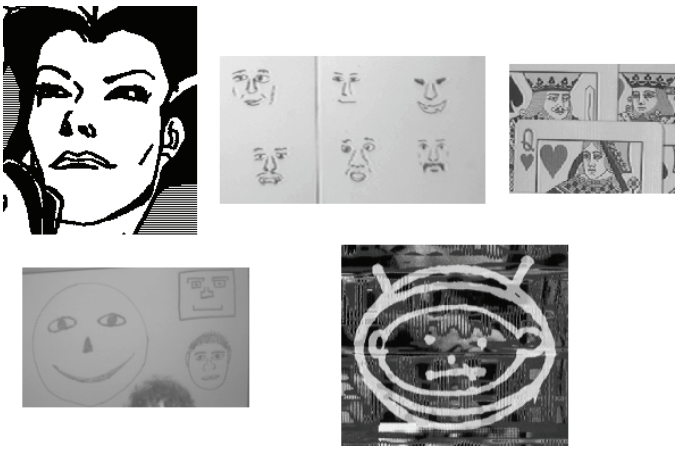


Figure 7.12: Examples of images that are considered as faces in some publications and as non faces is others.

Table 7.2: Performances on the CMU/MIT test set [136]. 3 datasets configurations are considered: Dataset 1: 155 faces, dataset2: 483 faces, Dataset 3: 507 faces. It shows the Detection rate (D.R.) and number of false alarms (F.A) for each method.

Methods	Dataset 1		Dataset 2		Dataset 3	
	D.R.(%)	F.A.	D.R.(%)	F.A.	D.R.(%)	F.A.
Rowley [136]	87.1	15	92.5	862	90.5	570
Sung Poggio [158]	81.9	13	—	—	—	—
Shneiderman [145]	—	—	93.0	88	94.4	65
Viola Jones [165]	—	—	—	—	91.4	50
12 stages HF	**83.7**	**20**	**88.6**	**95**	**88.3**	**50**
5 st. HF + 12 st.GF	**88.3**	**17**	**92.8**	**88**	**91.7**	**50**

is referred as Dataset 1 in table 7.2). The complete set contains 130 images with 507 faces (Dataset 3 in table 7.2). However, some of these annotated faces are manually drawn or sketches and they are counted as false detections in some publications.

To address this ambiguity, some papers only consider 123 images with 483 faces (Dataset 2 in table 7.2). Figure 7.12 gives some examples of such faces that are annotated in Dataset 3 but not in Dataset 2.

The three versions of the dataset are tested in this work to avoid any confusion. Figure 7.13 shows some detection results on images of this database.

Table 7.2 gives comparisons with the state-of-the-art methods on these datasets. It gives global performances of two versions of the system: a cascade of HF and a cascade of GF preprocessed by 5 stages of HF. The results in this table have to be taken cautiously as they are affected by many factors: the scanning parameters (scaling factor, window shifting step, etc,...), the technique chosen for merging overlapping windows, the number of training patterns and so forth. In particular, the way the non-face training examples are generated has a great impact on the decision functions. Finally some systems used

Figure 7.13: Face detection results on some images of the MIT/CMU testset [136].

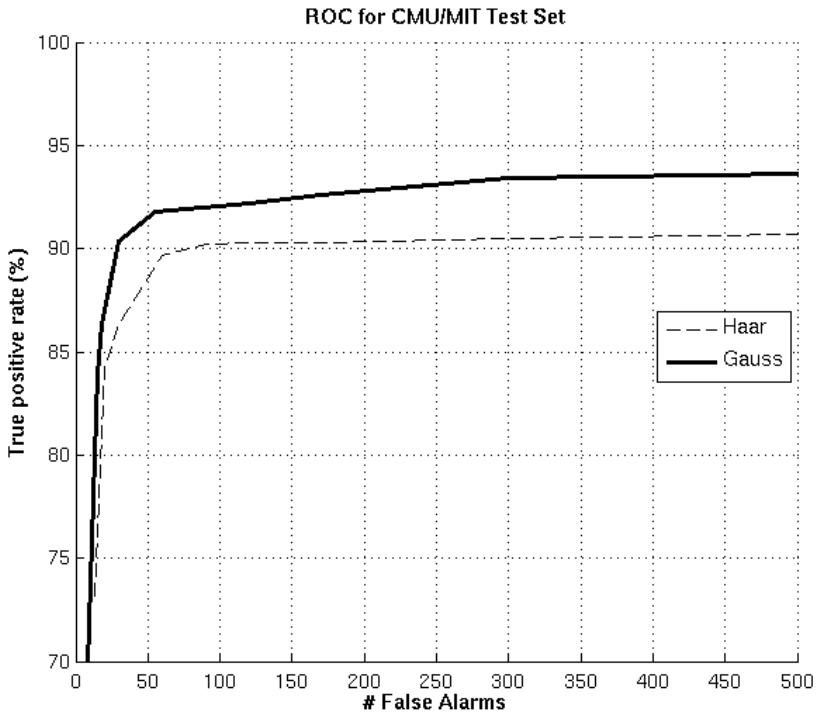


Figure 7.14: *ROC* analysis for comparing the algorithms on the MIT/CMU testset [136].

additional post-processing to improve the results. For example, Viola et al. in [165] used a voting strategy between several cascades to reduce the false positive rate. This explains why our cascade of HF has slightly lower performances than the implementation in [165]. However, the use of Gaussian filters gives roughly similar results with [165] without any post-processing, especially since we used only 1260 features in our cascade instead of 6061 in [165]. Shneiderman et al. in [145] report good performances. However they use several intensity corrections and a complex wavelets-based network that lead to extremely heavy computation. This will be discussed more deeply in chapter 8.

The purpose of this chapter is to show the advantages of using the Gaussian filters compared to Haar-like filters. Therefore, a Receiver Operating Characteristic curve (ROC) is given in figure 7.14. The two models were trained using the same data and the evaluation is performed with strictly identical parameters. It shows that the use of the new filters brings an important improvement in term of detection capabilities on real world data. For instance, considering a detection rate of 90 % for both classifiers, there are roughly twice less false alarms with GF.

7.4 Conclusions

This chapter presented new Gaussian-based filters which lead to high detection performances. These new local discriminant features are combined by boosting to model efficiently the face class and images are preprocessed by Haar features easier to compute in order to

speed up the detection. The complete system has been tested on classical datasets and compared with other relevant methods. In a future work we will introduce the combination of several parallel classifiers in order to reduce the false positive rate.

Classifier Combination for Frontal Face Detection

8

8.1 Introduction

In this chapter we present how simple classifier combination techniques can be efficiently used in a real face detection system. Classifiers used in this chapter are built from Haar-like filters and anisotropic Gaussian filters presented in the previous chapter. We will see how boosted classifiers combined by fixed probabilistic rules can be use in face class modelling applications. It will clearly show that decomposing a learning task into lower complexity problems can reduce training complexity and improve detection results. The complete face detection system contains three levels of classifier combination techniques:

1. *AdaBoost:* base classifiers are built from Haar-like or Gaussian filters using AdaBoost. The goal is to build a strong ensemble from a collection of weak classifiers based on local discriminant filters (GF ans HF).

2. *Cascades:* We still consider Cascades of boosted classifiers. They are sequential combinations of classifiers, mainly for improving the processing speed.

3. *Mixture of boosted classifiers:* Finally we consider a parallel combination scheme for combining several cascades of Gaussian-based classifiers. The goal of this stage is to simplify training process and improve generalization by reducing influence of noise and potential outliers in the training set.

The two first combination strategies (AdaBoot and cascades) have been discussed widely in the previous chapters. We will now concentrate on the third point which concerns mixtures of boosted classifiers. The remaining of this chapter is structured as follows. Section 8.2 presents the mixtures of boosted classifiers and how they are combined together

to take the final decision. Section 8.3 reports some results as well as comparisons with relevant existing face detectors. Finally, we draw some conclusions in section 8.4.

8.2 Mixtures of Boosted Classifiers

8.2.1 Motivations

This section introduces a structure that will improve the classification skills of the face detection system presented in chapter 7. As already mentioned in chapter 6, one of challenges of face detection resides in the fact that a very large set of face and non-face examples must be collected, for two main reasons:

- Face class has a large variance as face is a deformable object whose appearance changes according to various factors such as lighting conditions, changes in head pose, identity of the person, age, etc.
- We need to add more variation in the face training set in order to detect faces with slight in-plane rotations. Images with slight changes in scale and shift must also be taken into account depending on the scanning parameters used.

Moreover, a large number of features is also needed to obtain a sufficiently low false positive rates. AdaBoost minimizes an exponential loss function (see equation (3.11)) such that after several iterations, many features have to be added for only slightly reducing the false positive rate. Consequently, some weak classifiers potentially very efficient as experts on subspaces of the training data might behave worse on the whole training set.

These motivations suggest to use a multi-classifier structure built in parallel. Instead of training a single boosted cascade on the complete training set, we propose to build several cascades on subsets of the original dataset. A similar technique was developed and discussed in chapter 5 where Support Vector Machines were used for the parallel classifiers. This is similar to Bagging introduced in [10], except that the size of the bootstrap samples is smaller than the initial sample set.

This approach presents several advantages over a single classifier trained on the complete training set. On the one hand, each classifier is trained on a subset of the training dataset so that it can be seen as an expert that focuses on its own domain. This will thus decrease the influence of potential outliers in the complete training set. More specifically, as the power of AdaBoost resides in the fact that it focuses on the hard to classify examples, the parallelization technique reduces the weight of the noisy examples or potential outliers. This last point also reduces the risk of overfitting as mentioned above.

On the other hand, this parallelization technique also allows to decrease the classifier complexity. The complexity of training AdaBoost varies linearly with the number of samples. Splitting the data and training several AdaBoosted classifiers on the subsets will thus not affect the training complexity as compared to a single AdaBoosted classifier. However, as it will be shown in section 8.3, less features are needed for achieving equivalent classification rates compared to a single classifier trained on the complete training set.

8.2.2 Splitting Strategy

We could imagine two strategies for splitting the dataset into several subsets: either random sampling if we want to estimate several times the decision boundary or clustering if we want to build experts on subsets of the face class. In our case, no information is available about the distribution of the face class, our ultimate goal is to simplify the problem while trying to improve the classification skills, that is why simple random sampling has been chosen for creating the training subsets. Another reason why the clustering would not be appropriate comes from the variations introduced in the training set. This variability is obtained by slightly rotating, shifting and scaling the original face images. The clustering would eventually cluster examples resulting from similar transformations and thus the combination would certainly fail.

8.2.3 Posterior Probability Estimation

The decisions of the parallel classifiers are then combined using simple probability rules. For this, we need to obtain estimates of posterior probabilities of the boosted classifiers. There are basically two approaches for estimating posteriors. On the first hand, let us recall that AdaBoost can be interpreted as a soft margin classifier [126]. Using a similar technique as for *SVM* in section 5.3 [118], we can fit a simple continuous model to the margin distribution output by Adaboost (e.g. a sigmoid function like in equation (5.3)). On the second hand, a more natural way of estimating posterior probabilities from AdaBoost is to consider the probabilistic interpretation of boosting [68] presented in section 3.5.6. We recall the main idea here. AdaBoost can be viewed as a simple additive model by considering the criterion $J(H) = E[\exp(-yH(\mathbf{x}))]$, where $H(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$ is the output of AdaBoost. Friedman proved in [68] that the function minimizing J is the symmetric logistic transform of $p(y = 1|\mathbf{x})$. The posterior probabilities $P(y = 1|\mathbf{x})$ and $P(y = -1|\mathbf{x})$ are then given by:

$$p(y = 1|\mathbf{x}) = \frac{\exp(H(\mathbf{x}))}{\exp(-H(\mathbf{x})) + \exp(H(\mathbf{x}))}, \tag{8.1}$$

$$p(y = -1|\mathbf{x}) = \frac{\exp(-H(\mathbf{x}))}{\exp(-H(\mathbf{x})) + \exp(H(\mathbf{x}))}. \tag{8.2}$$

These posterior probabilities are directly used for combining the decisions of the parallel classifiers are the confidence level. We consider the fixed probability rules defined in section 3.3.1. Previous studies [34, 76, 101, 120] noticed that the choice of the rule has not a large influence on the overall performances. In this work we only consider the summation rule defined in equation (3.5) for combining the decisions of the multiple boosted classifiers. The choice of this rule is influenced by the splitting method that is used. The sum rule averages the decisions of the individual classifiers such that it sets a good trade off for discarding false alarms while preserving the correct detection of faces. For example the product rule is known to be a severe rule which risks to strongly penalize the true positive rate.

A reason why simple probability rules are preferred for combining the expertise of each

individual classifier is the stability of the parallel classifiers. The boosted cascades are stable in the sense that small changes in the training set lead to small changes in the classifier output [83]. More sophisticated combination techniques like Boosting need unstable classifiers to improve the overall performance.

8.2.4 Discussion

This parallelization technique presents some advantages against the cascade structure. A cascade of classifiers is a sequential combination of classifiers such that an example is rejected if it is classified as negative at any stage of the cascade. It can be seen as a mixture of classifiers but considering a product probability rule for combining the decisions. In fact if we consider the parallel classifiers to be conditionally independent (which can be assumed in this study as we use random sampling for generating the subsets), if one of the classifiers considers an example as negative with probability close to 1, the probability that the final decision is negative will be high. The only difference would be from the complexity point of view as we would have to test all the classifiers whereas the cascade would directly stop the processing chain.

One advantage of our parallel approach over the cascade is that if a positive example is classified as negative by a given classifier, it can be reassigned to the positive class by the overall system where in the cascade case it would be rejected. This would especially happen in the last stages of the cascade as the remaining candidate examples are the most challenging to classify. It is clear that the mixture approach will not reduce the testing time as we roughly use the same features number as in a single layer classifier. However we do not need to optimize the testing time as a simple pre-processing with a cascade based on Haar-like features will discard a large majority of candidate windows. Thus only few remaining critical windows will be tested by the mixture of boosted classifiers.

8.3 Experiments and Results

8.3.1 Structure of the System

The following experimental setup has been carried out in order to evaluate the performances of the new system, in particular, the improvements brought by the paralellization technique. As in the previous chapter introducing the new anisotropic filters, we scan images with a 20×15 pixels window and use a scaling factor of 1.2 for detecting faces at any scale. For details on the training sets used and the choice of the hyperparameters of the various classifiers, see section 7.3. The same pre-processing based on a cascade of HF is used in order to speed up the detection. The structure of the complete system is shown in figure 8.1. The final mixture contains 5 parallel classifiers which represents a good trade-off between gain in robustness and both training and testing complexity.

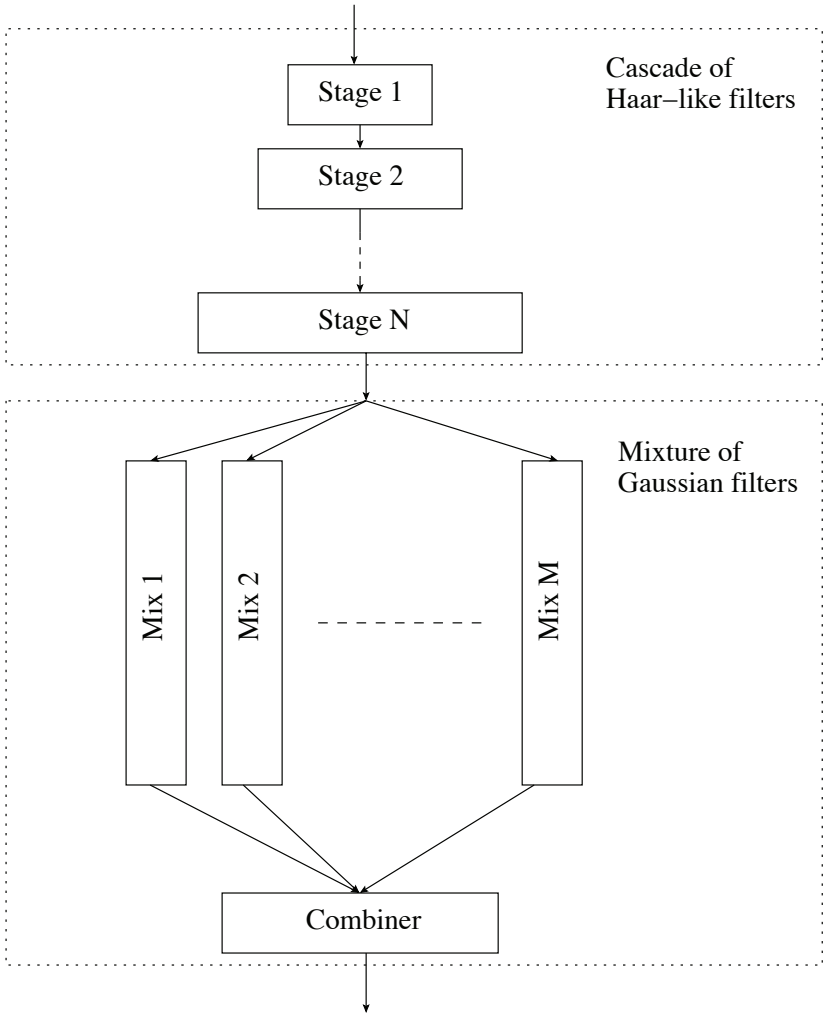


Figure 8.1: Structure of the complate face detector. It comprises a cascade of 5 HF stages followed by a mixture of 5 GF cascades.

Table 8.1: Comparisons of various methods tested on the BANCA [4] database. Results are reported for the French and English parts following the evaluation protocol described in [122]. Detections with a global score larger than 0.95 are considered as correct.

Classifier	% of detections with score > 0.95 (following [122])
5 stages Boosted HF	52.08
5 st. Boosted HF + 12 st. GF	90.74
5 st. Boosted HF + Mix Boosted GF	96.37

8.3.2 BANCA Database

This section gives results on BANCA dataset [4] using the evaluation protocol defined in [122]. Table 8.1 gives a comparison of different classifiers tested on the 12 480 images. A first classifier is made of 5 stages of a cascade of Haar features. It discards a large majority of negative windows but is not sufficient for being used alone. It is therefore used as pre-processing to speed up the process. The thresholds are tuned in order to have a very few positive windows rejected on the validation set (even if the false positive rate is increased). Then a classifier trained using GF using the complete training set is added. It is similar to the GF-based cascade presented in chapter 7. It significantly improves the classification rates. Finally the parallelization strategy has been tested for training the classifiers with GF, we improve the performances by roughly 6% compared to one single GF classifier. The single Gaussian-based classifier was trained using the same data than the complete mixture. However, the total number of features used in the mixture is much lower than in the complete cascade. One could imagine that the parallelization technique would increase the complexity of the system, but the mixture model is in fact sparser than its single counterpart.

8.3.3 CMU/MIT Test Set

This section gives experimental results on the CMU/MIT Test set [136]. As in section 7.3, we consider several configurations of the test set depending on the images that are considered in various papers. Dataset 1 in table 8.2 contains 23 images with a total of 155 faces, Dataset 2 in table 8.2 contains 123 images with 483 faces and Dataset 3 in table 8.2 contains 130 images with 507 faces. Figure 8.3 shows some detection results on images of this database. It compares the detector made of Gaussian filters with and without parallel combination.

We report in table 8.2 comparative results with other standard techniques on these CMU datasets. Performances of two different techniques are reported. On the first hand, a single cascade of GF and then a mixture of GF based classifiers. It comprises parallel 5 classifiers each made of roughly 100 filters. The number of classifiers in the mixture was chosen in order to obtain a good trade-off between false positive rate and detection speed. Each tested window is pre-processed using histogram equalization and simple illumination correction.

This study shows the improvements due to the parallelization technique. The faces

Table 8.2: Performances on the CMU/MIT test set [136]. 3 datasets configurations are considered: Dataset 1: 155 faces, dataset2: 483 faces, Dataset 3: 507 faces. It shows the Detection rate (D.R.) and number of false alarms (F.A) for each method.

Methods	Dataset 1		Dataset 2		Dataset 3	
	D.R.(%)	F.A.	D.R.(%)	F.A.	D.R.(%)	F.A.
Rowley [136]	87.1	15	92.5	862	90.5	570
Sung Poggio [158]	81.9	13	—	—	—	—
Shneiderman [145]	—	—	93.0	88	94.4	65
Viola Jones [165]	—	—	—	—	91.4	50
5 st. HF +12 st. GF	**88.3**	**17**	**92.2**	**88**	**91.7**	**50**
5 st. HF + Mix.	**89.2**	**15**	**92.1**	**68**	**93.9**	**60**

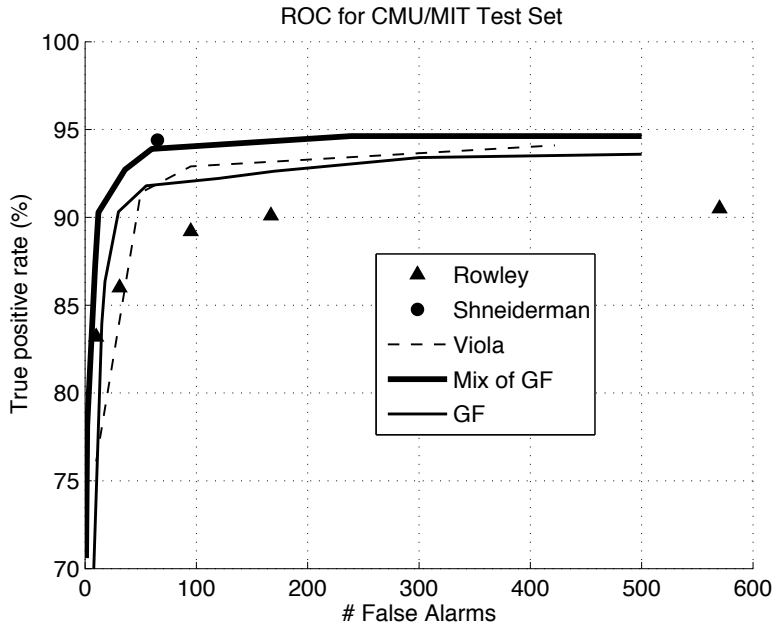


Figure 8.2: *ROC* analysis for comparing the algorithms on the MIT/CMU test set [136].

Table 8.3: Detection speed in frames per seconds (fps) of 4 detectors. The measure is an average over the 1500 frames of a sequence of 320×240 pixels images.

Detector	fps
12 stages HF	28.63
5 st. HF **+ 12 st.** GF	27.32
5 st. HF **+ 1 st.** GF	27.18
5 st. HF **+ Mix of** GF	27.22

that are drawings or sketches detected in Dataset 3 are counted as false detections in Dataset 2, it explains why there are more false detections in the smallest version of the test set. Table 8.2 only gives a single operating regime (i.e. single point on the The Receiver Operating Characteristic (ROC) curve). We thus also include complete ROC curves in figure 8.2. It shows that we gain several percents of detection rates compared to single GF-based classifier, and this at any operating point on the ROC.

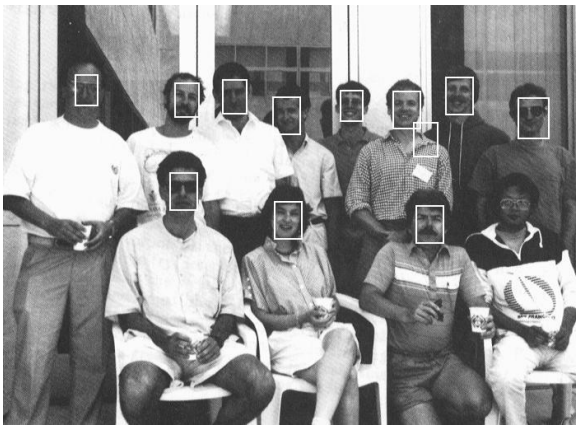
The Mixture of GF technique also compares favorably to state of the art. We recall here that the results in this table have to be taken cautiously as they are affected by many factors (e.g scanning procedure, technique for clustering overlapping windows, etc.) Objective comparison of detectors, while a desirable goal, is almost impossible in practice without access to the programs used by authors to build the models and perform the detection. However, one can have a rough idea about the relative time performance of various methods by looking at the complexity of the detection task, which in case of Shneiderman [145] comprises several intensity correction steps followed by a complex wavelets–based network. That is why we consider that the proposed method being faster and better suited for a low-latency system. We give in next section numerical comparisons of processing speed.

8.3.4 Processing Speed

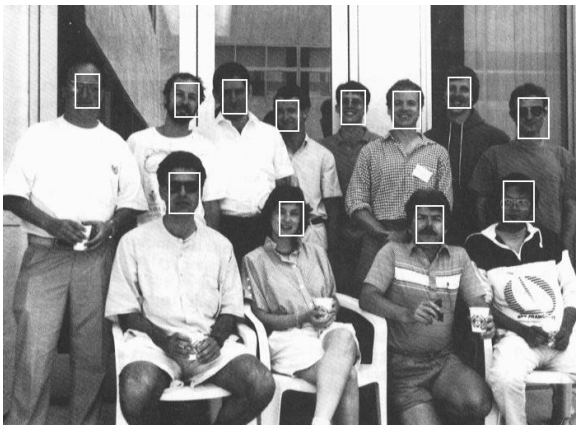
As noted in section 7.2.2, computing the response of a Gaussian filter was roughly 3 times more expensive than applying a Haar filter. Moreover the parallelization technique may also increase the processing time as all the candidate windows are tested by each classifier in the mixture (in opposition to the cascade architecture). However we show in this section that the overall detection speed in not significantly reduced by these two contributions.

Let us consider 4 detectors, all pre-processed by a cascade made of 5 stages of HF: a cascade of 7 other stages of HF, a cascade of 12 stages of GF, 1 stage of 500 GF, a mixture of GF. We apply these 4 detectors on a sequence of 1500 images with 320×240 pixels, each frame containing one or several faces. We then report in table 8.3 the average detection speed (number of frames per seconds).

Detectors with GF are only slightly slower than the one only based on HF. Moreover the speed of the GF-based detectors does not depend on the structure of the system after the pre-processing step. Using a cascade of GF, a single stage of GF or a mixture of GF lead to roughly equivalent detectors in terms of computation complexity. In fact the 5 stages of HF discard a large majority of non-face windows so that the computation of the



(a) 1 stage of GF



(b) Mixture of GF

Figure 8.3: Comparison between 1 cascade of GF (a) and a Mixture of GF (b), both pre-processed by 5 stages of HF. The image is taken from the CMU/MIT test set [136].

GF does not affect much the overall detection speed.

8.4 Conclusions

This chapter presents a complete frontal face detection system that implements a simple classifier combination technique. The detector uses a combination of several boosted classifiers which leads to high detection performances and can be applied in real-time. Each classifier is trained using local discriminant features based on anisotropic Gaussian filters. The complete training set is randomly subsampled and separate classifiers are trained on each subsets. They are then combined probabilistically. It has been shown that the mixture of boosted classifiers decreases significantly the false positive rate without affecting the true positive rate. The complete system has been tested on reference datasets and compared favorably to state-of-the-art.

9 Conclusions and Future Work

In this thesis, we covered various aspects of pattern recognition techniques, from both theoretical and practical perspectives. In particular, we studied classifier combination techniques and showed how they can be applied with success to face detection. We will now give a short summary of the main results described in this work. We also discuss the main limitations of the present approach as well as possible future works.

9.1 Achievements

As mentioned in the introduction of this thesis, the work presented here is twofold. First, we study classifier combination techniques, emphasizing of the notion of diversity between classifiers. In the second part of the thesis, we focus on face detection application, showing how classifier combination techniques can be used efficiently in a real-life problem.

In chapter 4, we introduce an information theoretic framework for studying classifier combination. We show how the main relevant concepts of classifier ensembles can be modeled by information theoretic measures. In particular, we propose to use mutual information to measure the dependency between classifiers in the ensemble and accuracy of each individual classifier. Information theoretic tools help us to demonstrate that a performant ensemble should contain classifiers that are both diverse and individually accurate. However, these two quantities, diversity and average individual accuracy, appear to be contradictory in the sense that we cannot maximize them both together. Two very accurate classifiers cannot be diverse, and inversely, two very diverse classifiers will necessarily have low classification rates. In order to tackle this contradiction, we propose a information theoretic score (ITS) that fixes a trade-off between these two quantities. A first possible use of this score is to consider it as a selection criterion for extracting a good ensemble in a predefined pool of classifiers. We also propose an ensemble creation technique based on

AdaBoost, by taking into account the information theoretic score for selecting iteratively the classifiers.

In chapter 5, we propose a complete study of one particular application of classifier ensembles: Multiple Support Vector Machines ($MSVMs$). We show that combining several Support Vector Machines (SVM) trained on subsets of the complete training set can help decreasing training complexity while increasing classification performances, in particular for large scale problems. A $MSVMs$ is defined by two main components:

- The splitting algorithm that splits the original training set into subsets. The most common is simple random sampling but clustering techniques can also be applied.
- The combination rule that will combine the decisions of each parallel SVM.

In this work, we propose several combination rules for merging the parallel decisions. The first possibility is to train a second layer SVM on the margins of the first layer SVM. As the margin can be viewed as a measure of confidence about predictions, more weight is given to the most reliable classifiers. However, this merging technique requires an additional training set for training the second layer SVM. To overcome this problem, we also propose to use simple fixed probability rules. They directly combine decisions of the parallel classifiers from estimates of the posterior probabilities. We then propose algorithms that find better partitioning of the initial training set, by taking into account the ITS. The first algorithm uses Genetic Algorithm for optimizing the score. It leads to very high classification performances compared to the previous combination techniques but is very computationally expensive. The second algorithm trains jointly SVM such that the same ITS level as Genetic Algorithm is reached, but with much lower computation requirements. It based on Kernel Adaton algorithm that learns on-line SVM. We introduce the concept of Ensemble Support Vectors that support the margin of the ensemble. Let us summarize its concept here. The algorithm starts by randomly splitting the complete training set into disjoint subsets. We then iteratively train the parallel SVM by taking into account the predictions of the support vectors of other SVM. The support vectors that are not classified correctly by the ensemble are iteratively added to training set of other classifiers, thus becoming Ensemble Support Vectors. We give complete evaluation and comparison of these various algorithms on several commom large scale datasets.

Finally, we also focus on the small sample case, for which the classical $MSVMs$ structure does not compares favorably with single SVM in terms of classification rates and training time. We propose a variant of $MSVMs$ that construct an ensemble of $MSVMs$ by using cross-validation techniques. Instead of acting on the training data, we generate several classifiers by changing the herparameters (e.g. kernel parameters of the SVM). The obtained ensemble maximizes the ITS.

In the second part of this thesis, we focus on the face detection application. We first introduce in chapter 7 new geometrical filters called anisotropic Gaussian filters, that are more discriminant than the rectangular Haar-like filters. We give comparison with standard techniques in terms of classification rates and processing speed. The new filters appear to

discriminate much better faces and non face examples that are close to the face class. The processing time is not much affected as we employ a short cascade of Haar-like filters to efficiently reduce the search space. Finally in chapter 8, we apply classifier combination paradigm to the face detection system. We propose a parallel mixture of boosted classifier for reducing the false positive rate and decreasing the training time while keeping the testing time unchanged. The complete face detection system is compared on several datasets, showing that it compares favorably to state-of-the-art techniques.

9.2 Perspectives

There are basically two main directions future work can take: one the one hand, it would be interesting to investigate how the information theoretic framework can be generalized to other classifier combination techniques. On the second hand, the techniques presented in this thesis on optimal classifier combination using *ITS* could be applied to the face detection application. More specifically, here is a list of possible perspectives:

- First, the *ITS* score defined in this work is independent of the application. It has been designed upon general theoretical considerations, so that it gives the main direction how to obtain a good ensemble. However it does not imply that the obtained ensemble is optimal. It would be interesting to investigate how this score could be adapted specifically depending on the application. In particular, if the classifiers in a pool have very different accuracies, the main hypothesis used for defining *ITS* may not hold as the measure of average individual accuracy may not be relevant measure of the global performance of the individual classifiers.

- The *ITS* has been tested on datasets with up to 10 classes. As it has been discussed in section 4.5.3, the number of classes influences the relationship between average individual accuracy and diversity. This phenomenon could be taken into account by considering a diversity term that includes the number of classes.

- The main limitations of the information theoretic framework, as presented in this thesis, is that it only applies for majority voting combination. Even if it has also been extended to weighted majority voting, it would be interesting to consider extending this framework to other combination rules.

- Multiple *SVM* as presented in chapter 5 produce very efficient ensembles of classifiers in practice. However, a pertinent research direction could be to study more deeply the influence of the the number of members in the ensemble. The number K of classifier in the ensemble has been set generally between 3 and 7 in order to keep a moderate training complexity while having *enough* data in each training subset. Setting the optimal K a priori is always a challenging task. This direction could be explored in order to further improve $MSVMs$.

- Concerning face detection application, several perspectives could be investigated. First, from an algorithmic point of view, new Boosting strategies could be used for training the classifiers: FloatBoost, RealAdaBoost, GentleBoost, or the ITS-AdaBoost presented here. The techniques proposed for *MSVMs* based on *ITS* could also be adapted to AdaBoost such that the mixture of boosted classifiers can reduce even more the number of false alarms.

- The work on frontal face detection can be extended to multi-view face detection by implementing the mixture of boosted classifiers into a tree of boosted classifiers. The idea is to hierarchically build pose-specific classifiers.

Curriculum Vitae

Julien Meynet
Avenue de Morges 17
CH-1004 Lausanne
Phone +41 763310150
Julien.meynet@epfl.ch

Age: 26
Single
French

Education

Swiss Federal Institute of Technology (EPFL), Lausanne 2003–2007
Ph.D. Candidate at the Signal Processing Institute STI/ITS

British Computer Society Summer School July 2005
3rd British Computer Society Summer School on Pattern Recognition, Plymouth, U.K.

Grenoble Institute of Technology (INPG), Grenoble, France 2002–2003
Masters degree in research (DEA) in Signal, Image, Speech and Telecommunications.

Grenoble Institute of Technology (INPG), Grenoble, France 2000–2003
Master in Electrical Engineering and Information Processing.
Last year achieved as an exchange student at the Ecole Polytechnique Federale de Lausanne (EPFL).

CPGE, Lycee Berthollet, Annecy, France. 1998–2000
Intensive two-year university foundation course preparing for the competitive entrance examinations to the 'Grandes Ecoles' (French Elite Institutes).

Professional Experience

Swiss Federal Institute of Technology (EPFL), Lausanne 2003–2007
Research assistant at the Signal Processing Institute.

- Research and development in the field of computer vision and pattern recognition with emphasis on the face detection application
- Participation in the Swiss National Centre for Competence in Research (NCCR) Interactive Multimodal Information Management (IM2)
- Participation in the European Network of Excellence SIMILAR on multimodal signal processing
- Advisor of 3 M.S. students
- Teaching assistant: lectures on Image Processing and Pattern Recognition

SICPA, Annemasse, France July 2002
Internship at SICPA, specialist in inks and varnish of printing works in Annemasse, France, as service engineer.

Language Skills

- **French**: mother tongue
- **English**: fluent
- **German**: good knowledge
- **Spanish**: good knowledge

Computer Skills

- **Programming:** C/C++, Java, Matlab, Perl, Python, HTML, Maple
- **Operating Systems:** MS Windows, Linux/Unix

Personal Interests

- **Sports**: Rock climbing, mountaineering, skiing
- **Music**: Playing guitar, amateur guitar luthier

List of Publications

Journal Papers

- J. Meynet and J.-P. Thiran, **Information Theoretic Combination of Classifiers With Application to Ensembles of SVMs**, submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.
- J. Meynet, V. Popovici and J. Thiran, **Mixtures of Boosted Classifiers for Frontal Face Detection**, Signal, Image and Video Processing, Vol. 1, Nr. 1, pp. 29–38, 2007.
- J. Meynet, V. Popovici and J. Thiran, **Face Detection with Boosted Gaussian Features**, Pattern Recognition, Vol. 40, Nr. 8, pp. 2283-2291, 2007.

Conference Papers

- B. Noris, K. Benmachiche, J. Meynet, J.-P. Thiran and A. Billard, **Analysis of Head Mounted Wireless Camera Videos for Early Diagnosis of Autism**, International Conference on Recognition Systems, 2007.
- J. Meynet and J.-P. Thiran, **Information Theoretic Combination of Classifiers with Application to AdaBoost**, 7th international Workshop on Multiple Classifier Systems (MCS), 2007.
- V. Popovici, J. Meynet and J. Thiran, **Anisotropic Gaussian Filters for Face Class Modeling**, Proceedings of IEEE 2nd International Conference on Intelligent Computer Communication and Processing (ICCP), Vol. 1, pp. 121-127, 2006.
- J. Meynet, V. Popovici, M. Sorci and J. Thiran, **Combining SVMs for Face Class Modeling**, 13th European Signal Processing Conference - EUSIPCO, 2005.
- J. Meynet, V. Popovici and J. Thiran, **Mixture of SVMs for Face Class Modeling**, MLMI'04: Proceedings of the Workshop on Machine Learning for Multimodal Information, Lecture Notes in Computer Science, Vol. 3361, pp. 173-181, 2004.
- J. Meynet, V. Popovici and J. Thiran, **Face Class Modeling Using Mixture of SVMs**, In Proceedings of International Conference on Image Analysis and recognition, ICIAR 2004, Porto, Portugal, Lecture Notes In Computer Science, Vol. 3212, pp. 709-716, 2004.

Technical Reports

- J. Meynet, T. Arsan, J. Cruz Mota and J.-P. Thiran **Fast Multiview Face Tracking with Pose Estimation**, Technical Report, 2007.
- J. Meynet and J. Thiran, **Information Theoretic Combination of Classifiers With Application to Multiple SVMs**, Technical Report, 2006.

- J. Meynet, V. Popovici and J. Thiran, **Face Detection with Mixtures of Boosted Discriminant Features**, Technical Report, 2005.

Bibliography

[1] M. A. Aizerman, E. M. Braverman, L. I. Rozonoér (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control* **25**:821–837.

[2] A. Asuncion, D. Newman (2007). UCI machine learning repository .

[3] R. Avnimelech, N. Intrator (1999). Boosted mixtures of experts: An ensemble learning scheme. *Neural Computation* **11**:475–490.

[4] E. Bailly-Bailliere et al. (2003). The banca database and evaluation protocol. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication (ICAVBPA), Guildford, UK*, pp. 625–638.

[5] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **19**(7):711–720.

[6] J. A. Benediktsson, P. H. Swain (1992). Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics* **22**(4):688–704.

[7] P. Besson et al. (2007). Extraction of audio features specific to speech production for multimodal speaker detection. *IEEE Transactions on Multimedia* .

[8] C. M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.

[9] J. D. Borda (1781). Mémoire sur les elections au scrutin. *Histoire de l'Académie Royale des Sciences* .

[10] L. Breiman (1996). Bagging predictors. *Machine Learning* **24**(2):123–140.

[11] L. Breiman (1998). Arcing classifiers. *The Annals of Statistics* **26**(3):801–849.

[12] L. Breiman (2001). Random forests. *Machine Learning* **45**(1):5–32.

[13] L. Breiman, J. Friedman, R. Olshen, C. Stone (1984). *Classification and regression trees*. Wadsworth International Group.

[14] G. Brown, J. Wyatt, R. Harris, X. Yao (2005). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion* **6**(1):5–20.

[15] S. C. Brubaker, M. D. Mullin, J. M. Rehg (2006). Towards optimal training of cascaded detectors. In *9th European Conference on Computer Vision ECCV, Graz, Austria*, pp. 325–337.

[16] M. Burl, T. Leung, P. Perona (1995). Face localization via shape statistics. In *International Workshop on Automatic Face and Gesture Recognition (AFGR), 1995.*

[17] T. Butz, J.-P. Thiran (2005). From error probability to information theoretic (multi-modal) signal processing. *Signal Processing* **85**(5):875–902.

[18] C.-C. Chang, C.-J. Lin (2001). *LIBSVM: a library for support vector machines.* Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[19] J. Chen et al. (2004). Novel face detection method based on gabor features. In *Advances in Biometric Person Authentication, 5th Chinese Conference on Biometric Recognition, SINOBIOMETRICS 2004, Guangzhou, China,*, pp. 90–99.

[20] R. Collobert, S. Bengio, Y. Bengio (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation* **14**(5).

[21] T. F. Cootes, C. J. Taylor (1992). Active shape models: Smart snakes. In *British Machine Vision Conference (BMVC)*, pp. 267–275.

[22] T. Cover, J. Thomas (1991). *Elements of Information Theory.* John Wiley and Sons, Inc., New York.

[23] I. Craw, H. Ellis, J. R. Lishman (1987). Automatic extraction of facial features. *Pattern Recognition Letters* **5**(2):183–187.

[24] I. Craw, D. Tock, A. Bennett (1992). Finding face features. In *European Conference on Computer Vision (ECCV)*, pp. 92–96.

[25] F. Crow (1984). Summed-area tables for texture mapping. *Computer Graphics - SIGGRAPH* **18**(3):207–212.

[26] P. Cunningham (2000). Overfitting and diversity in classification ensembles based on feature selection. *Technical Report TCD-CS-2000-07, Department of Computer Science, Trinity College Dublin* .

[27] P. Cunningham, J. Carney (2000). Diversity versus quality in classification ensembles based on feature selection. In *European Conference on Machine Learning (ECML)*, pp. 109–116.

[28] P. A. Devijver, J. Kittler (1982). *Pattern recognition: A statistical approach.* Prentice Hall, New York.

[29] T. G. Dietterich (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* **10**(7):1895–1924.

[30] T. G. Dietterich (2000). Ensemble methods in machine learning. *Lecture Notes in Computer Science* **1857**:1–15.

[31] T. G. Dietterich (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* **40**(2):1–19.

[32] R. Duda, P. Hart (1973). *Pattern classification and scene analysis*. John Wiley & Sons, New York.

[33] R. Duin (2002). The combining classifier: To train or not to train? In *International Conference on Pattern Recognition (ICPR)*, pp. 765–770.

[34] R. Duin, D.Tax (2000). Experiments with classifier combining rules. In *Multiple Classifier Systems*, pp. 16–29.

[35] R. Duin et al. (2004). Prtools4, a matlab toolbox for pattern recognition. *Delft University of Technology* .

[36] D. Erdogmus, J. C. Principe (2004). Lower and upper bounds for misclassification probability based on renyi's information. *Journal of VLSI Signal Processing* **37**:305–317.

[37] T. Evgeniou, M. Pontil, A. Elisseeff (2004). Leave one out error, stability, and generalization of voting combinations of classifiers. *Machine Learning* **55**(1):71–97.

[38] R. Fano (1961). *Transmission of Information: A Statistical Theory of Communication*. MIT Press, Wiley, Cambridge.

[39] R. Féraud, O. Bernier, J.-E. Viallet, M. Collobert (2001). A fast and accurate face detector based on neural networks. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **23**(1):42–53.

[40] J. Fisher, III, J. Principe (1998). A methodology for information theoretic feature extraction. In *IEEE International Conference on Neural Networks (IJCNN'98)*, vol. 3, pp. 1712–1716.

[41] Fletcher (1987). *Practical Methods of Optimization*. Wiley.

[42] Y. Freund (1995). Boosting a weak learning algorithm by majority. *Information and Computation* **121**(2):256–285.

[43] Y. Freund, Y. Mansour, R. Schapire (2001). Why averaging classifiers can protect against overfitting. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*.

[44] Y. Freund, R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**:119–139.

[45] J. H. Friedman (1996). On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data mining and Knowledge Discovery* **1**(1):54–77.

[46] T.-T. Frieß, N. Cristianini, C. Campbell (1998). The Kernel-Adatron algorithm: a fast and simple learning procedure for Support Vector machines. In *Procedings of the. 15th International Conference on Machine Learning (ICML)*, pp. 188–196.

[47] R. W. Frischholz, U. Dieckmann (2000). Bioid: A multimodal biometric identification system. *IEEE Computer* **33**(2):64–68.

[48] K. Fukunaga (1990). *Introduction to Statistical Pattern Recognition (Second Edition)*. Academic Press, New York.

[49] S. Geman, E. Bienenstock, R. Doursat (1992). Neural networks and the bias/variance dilemma. *Neural Computation* **4**(1):1–58.

[50] G. Giacinto, F. Roli (2001). Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing* **19**(9):699–707.

[51] V. Govindaraju (1996). Locating human faces in photographs. *International Journal of Computer Vision* **19**(2):129–146.

[52] Y. Grandvalet (2001). Bagging can stabilize without reducing variance. In *International Conference on Artificial Neural Networks - ICANN, Vienna, Austria*, pp. 49–56.

[53] S. Guenter, H. Bunke (2004). Optimization of weights in a multiple classifier. handwritten word recognition system using a genetic algorithm,. *Electronic Letters of Computer Vision and Image Analysis, ELCVIA* **3**(1):25 – 44.

[54] S. R. Gunn, M. S. Nixon (1998). Global and local active contours for head boundary extraction. *International Journal of Computer Vision* **30**(1):43–54.

[55] M. Gurban, A. Valles, J.-P. Thiran (2007). Low-Dimensional Motion Features for Audio-Visual Speech Recognition. In *15th European Signal Processing Conference (EUSIPCO), Poznan, Poland*.

[56] S. T. Hadjitodorov, L. I. Kuncheva, L. P. Todorova (2006). Moderate diversity for better cluster ensembles. *Information Fusion* **7**(3):264–275.

[57] T. Hastie, R. Tibshirani, J. H. Friedman (2001). *The Elements of Statistical Learning*. Springer.

[58] S. Haykin (1999). *Neural Networks: A Comprehensive Introduction*. Prentice Hall.

[59] B. Heisele, T. Serre, M. Pontil, T. Poggio (2001). Component-based face detection. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 657–662.

[60] K. Hild, D. Erdogmus, K. Torkkola, J. C. Principe (2006). Feature extraction using information-theoretic learning. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **28**(9):1385–1392.

[61] E. Hjelmås, B. K. Low (2001). Face detection: a survey. *Computer Vision and Image Understanding* **83**(3):236–274.

[62] H. Hotelling (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* **24**:417–441.

[63] Y. S. Huang, C. Y. Suen (1995). A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **17**(1):90–94.

[64] R. Jacobs, M. Jordan, S. J. Nowlan, G. E. Hinton (1998). Adaptive mixtures of local experts. *Neural Computation* **3**:79–87.

[65] Jain, Mao, Mohiuddin (1996). Artificial neural networks: A tutorial. *IEEE Computer* **29**(3):31–44.

[66] A. Jain, R. Duin, J. Mao (2000). Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **22**(1):4–37.

[67] A. K. Jain, D. Zongker (1997). Feature selection: Evaluation, application and a small sample performance. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **19**:153–158.

[68] T. H. Jerome Friedman, R. Tibshirani (2000). Additive logistic regression, a statistical view of boosting. *The Annals of Statistics* **28**(2):337–374.

[69] O. Jesorsky, K. J. Kirchberg, R. W. Frischholz (2001). Robust face detection using the hausdorff distance. In *International Conference on Audio- and Video-Based Biometric Person Authentication (ICAVBPA)*, pp. 90–95.

[70] M. I. Jordan, R. A. Jacobs (1994). Hierarchical mixtures of experts and the em algorithms. *Neural Computation* **6**:181–214.

[71] M. I. Jordan, L. Xu (1995). Convergence results for the em approach to mixtures of experts architectures. *Neural Networks* **8**(9):1409–1431.

[72] P. Kakumanu, S. Makrogiannis, N. Bourbakis (2007). A survey of skin-color modeling and detection methods. *Pattern Recognition* **40**(3):1106–1122.

[73] M. Kass, A. Witkin, D. Terzopoulos (1988). Snakes: Active contour models. *International Journal of Computer Vision* **1**(4):321–331.

[74] J. Kittler (1986). Feature selection and extraction. In *Handbook of Pattern Recognition and Image Processing*, pp. 59–83.

[75] J. Kittler, K. Fu, L. Pau (1982). *Pattern Recognition Theory and Applications*. Kluwer.

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

[77] R. Kohavi (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1137–1145.

[78] R. Kohavi, D. Wolpert (1996). Bias plus variance decomposition for zero-one loss functions. In *International Conference on Machine Learning (ICML)*, pp. 275–283.

[79] Krogh, Vedelsby (1995). Neural network ensembles, cross validation, and active learning. *Seventh Conference on Neural Information Processing Systems (NIPS)* pp. 234–38.

[80] L. Kuncheva, C. Whitaker (2002). Using diversity with three variants of boosting: Aggressive, conservative, and inverse. In *International Workshop on Multiple Classifier Systems (MCS)*, pp. 81–90.

[81] L. Kuncheva, C. Whitaker, C. Shipp, R. Duin (2003). Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications* **6**:22–31.

[82] L. I. Kuncheva (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley.

[83] L. I. Kuncheva (2004). *Combining Pattern Classifiers Methods and Algorithms*. John Wiley, New York„ New York, NY, USA.

[84] L. I. Kuncheva, L. C. Jain (2000). Designing classifier fusion systems by genetic algorithms. *IEEE Trans. Evolutionary Computation* **4**(4):327–348.

[85] L. I. Kuncheva, C. J. Whitaker (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning* **51**(2):181–207.

[86] J. T. Kwok (1998). Support vector mixture for classification and regression problems. In *International Conference on Pattern Recognition (ICPR)*, vol. 1, pp. 255–258.

[87] K.-M. Lam, H. Yan (1996). Locating and extracting the eye in human face images. *Pattern Recognition* **29**(5):771–779.

[88] L. Lam, C. Y. Suen (1995). Optimal combinations of pattern classifiers. *Pattern Recognition Letters* **16**(9):945–954.

[89] L. Lam, S. Suen (1997). Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics* **27**:553–568.

[90] A. Lanitis, C. J. Taylor, T. F. Cootes (1997). Automatic interpretation and coding of face images using flexible models. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **19**(7):743–756.

[91] C. H. Lee, J. S. Kim, K. H. Park (1996). Automatic human face location in a complex background using motion and color information. *Pattern Recognition* **29**(11):1877–1889.

[92] T. Leung, M. Burl, P. Perona (1995). Finding faces in cluttered scenes using labelled random graph matching. In *International Conference on Computer Vision (ICCV)*, pp. 637–644.

[93] S. Z. Li, Z. Zhang (2004). Floatboost learning and statistical face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **26**(9):1112–1123.

[94] R. Lienhart, A. Kuranov, V. Pisarevsky (2003). Empirical analysis of detection cascades of boosted classifiers for rapid object detection,. In *German Pattern Recognition Symposium*, pp. 297–304.

[95] H. Luo (2005). Optimization design of cascaded classifiers. In *International Conference on Pattern Recognition (CVPR)*, pp. 480–485.

[96] J. MacQueen (1967). Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Mathematics Statistics and Probability*, pp. 281–297.

[97] R. Meir, G. Rätsch (2002). An introduction to boosting and leveraging. In *Machine Learning Summer School*, pp. 118–183.

[98] P. Melville, R. J. Mooney (2005). Creating diversity in ensembles using artificial data. *Information Fusion* **6**(1):99–111.

[99] J. Mercer (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of Royal Society London* **A209**:415–446.

[100] K. Messer et al. (1999). Xm2vtsdb: The extended m2vts database. In *Second International Conference on Audio and Video-based Biometric Person Authentication (ICAVBPA)*, pp. 72–77.

[101] J. Meynet, V. Popovici, M. Sorci, J. Thiran (2005). Combining SVMs for Face Class Modeling. In *13th European Signal Processing Conference - EUSIPCO*.

[102] J. Meynet, V. Popovici, J. Thiran (2004). Face Class Modeling Using Mixture of SVMs. In *In Proceedings of International Conference on Image Analysis and recognition, ICIAR 2004, Porto, Portugal,*, pp. 709–716.

[103] L. C. Molina, L. Belanche, À. Nebot (2002). Feature selection algorithms: A survey and experimental evaluation. In *International Conference on Data Mining (ICDM)*, pp. 306–313.

[104] K.-R. Müller et al. (2001). An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks (TNN)* **12**:181–201.

[105] Narasimhamurthy (2005). Theoretical bounds of majority voting performance for a binary classification problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)* **27**(12):1988– 1995.

[106] P. M. Narendra, K. Fukunaga (1977). A branch and bound algorithm for feature subset selection. *IEEE Trans. Computers* **26**(9):917–922.

[107] A. B. Novikoff (1962). On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata* **12**:615–622.

[108] T. Ojala, M. Pietikäinen, D. Harwood (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition* **29**(1):51–59.

[109] N. Oliver, A. P. Pentland, F. Berard (2000). LAFTER: a real-time face and lips tracker with facial expression recognition. *Pattern Recognition* **33**(8):1369–1382.

[110] D. Opitz, R. Maclin (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* **11**:169–198.

[111] E. Osuna, R. Freund, F. Girosi (1997). Training support vector machines: an application to face detection. In *International Conference on Pattern Recognition (CVPR)*, pp. 130–136.

[112] V. Pavlovic, A. Garg (2001). Efficient detection of objects and attributes using boosting. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[113] F. Peng, R. A. Jacobs, M. A. Tanner (1995). Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *Journal of the American Statistical Association* **91**(435):953–960.

[114] A. P. Pentland, B. Moghaddam (1996). Probabilistic visual learning for object representation. In *Early Visual Learning*, pp. 99–130.

[115] L. Peotta, L. Granai, P. Vandergheynst (2003). Very low bit rate image coding using redundant dictionaries. In *Applications in Signal and Image Processing (SPIE)*, pp. 228–239.

[116] P. J. Phillips, H. Wechsler, J. Huang, P. J. Rauss (1998). The FERET database and evaluation procedure for face-recognition algorithms. vol. 16, pp. 295–306.

[117] J. Platt (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning* pp. 185–208.

[118] J. Platt (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classiers, eds., MIT Press*, pp. 61–74.

[119] T. Poggio, F. Girosi (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science* **247**:978–982.

[120] V. Popovici (2004). *kernel-based classifiers with application to face detection.* Ph.D. thesis, EPFL lausanne.

[121] V. Popovici, J. Thiran (2003). Face Class Modeling in Eigenfaces Space. In *3rd International Workshop on Pattern Recognition in Information Systems (PRIS), Anger, France*, pp. 38–45.

[122] V. Popovici, J.-P. Thiran, Y. Rodriguez, S. Marcel (2004). On performance evaluation of face detection and localization algorithms. In *International Conference on Pattern Recognition (ICPR)*, pp. 313–317.

[123] J. Principe, D. Xu, J. Fisher (2000). Learning from examples with information theoretic criteria. *J. VLSI Signal Processing Systems* **26**:61–77.

[124] R. Quinlan (1986). Induction of decision trees. *Machine Learning* **1**(1):81–106.

[125] G. Rätsch, T. Onoda, K.-R. Mueller (2001). Soft margins for adaboost. *Machine Learning* **42**(3):287.

[126] G. Rätsch, M. K. Warmuth (2005). Efficient margin maximizing with boosting. *Journal of Machine Learning Research* **6**:2131–2152.

[127] S. Raudys, F. Roli (2003). The behavior knowledge space fusion method: Analysis of generalization error and strategies for performance improvement. In *International Workshop on Multiple Classifier Systems (MCS)*, pp. 55–64.

[128] S. J. Raudys, V. Pikelis (1980). On dimensionality, sample size, classification error, and complexity of classification algorithms in pattern recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **2**(3):243–252.

[129] D. Reisfeld, H. Wolfson, Y. Yeshurun (1995). Context-free attentional operators: The generalized symmetry transform. *International Journal of Computer Vision* **14**(2):119–130.

[130] J. Richiardi, A. Drygajlo (2007). Reliability-based voting schemes using modality-independent features in multi-classifier biometric authentication. In *International Workshop on Multiple Classifier Systems (MCS)*, pp. 377–386.

[131] J. J. Rodríguez, L. I. Kuncheva, C. J. Alonso (2006). Rotation forest: A new classifier ensemble method. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **28**(10):1619–1630.

[132] Y. Rodriguez (2006). *Face detection and verification using local binary patterns*. Ph.D. thesis, EPFL Lausanne.

[133] Y. Rodriguez, F. Cardinaux, S. Bengio, J. Mariéthoz (2004). Estimating the quality of face localization for face verification. In *International Conference on Image Processing, ICIP*, vol. 01.

[134] F. Roli, S. Raudys, G. L. Marcialis (2002). An experimental comparison of fixed and trained fusion rules for crisp classifier outputs. In *International Workshop on Multiple Classifier Systems (MCS)*, pp. 232–241.

[135] F. Rosenblatt (1962). *Principles of Neurodynamics*. Spartan, New York.

[136] H. A. Rowley, S. Baluja, T. Kanade (1998). Neural network-based face detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)* **20**(1):23–38.

[137] C. Rudin, I. Daubechies, R. E. Schapire (2004). The dynamics of adaboost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research* **5**:1557–1595.

[138] D. Ruta, B. Gabrys (2002). A theoretical analysis of the limits of majority voting errors for multiple classifier systems. *Pattern Analysis and Applications* **5**(4):333–350.

[139] T. Sakai, M. Nagao, M. Kidode (1971). Processing of multilevel pictures by computer - the case of photographs of human face. *Systems Computers Controls* **2**(3):445–452.

[140] A. Samal, P. A. Iyengar (1995). Human face detection using silhouettes. *International Journal of Pattern Recognition and Artificial Intelligence* **9**:845–867.

[141] R. Schapire (1990). The strength of weak learnability. *Machine Learning* **5**:197–227.

[142] R. E. Schapire (2002). Advances in boosting. In *UAI'02*, pp. 446–452.

[143] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics* **26**(5):1651–1686.

[144] R. E. Schapire, Y. Singer (1999). Improved boosting using confidence-rated predictions. *Machine Learning* **37**(3):297–336.

[145] H. Schneiderman, T. Kanade (2000). A statistical method for 3D object detection applied to faces and cars. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 746–751.

[146] B. Schoelkopf, A. J. Smola (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA.

[147] L. Shapley, B. Grofman (1984). Optimizing group judgemental accuracy in the presence of interdependencies. *Public Choice* **43**:329–343.

[148] J. Shawe-Taylor, N. Cristianini (2004). *Kernel Methods for Pattern Analysis*. CUP.

[149] C. Shipp, L. Kuncheva (2002). An investigation into how adaboost affects classifier diversity. In *Information Processing and management of Uncertainty in Knowledge-based Systems (IPMU), Annecy, France*.

[150] C. A. Shipp (2004). *A Study of Diversity in Classifier Ensembles*. Ph.D. thesis, Bangor university.

[151] C. A. Shipp, L. I. Kuncheva (2002). Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion* **3**:135–148.

[152] L. C. D. Silva, K. Aizawa, M. Hatori (1995). Detection and tracking of facial features by using a facial feature model and deformable circular template. *Transactions on Information and Systems* **E78-D**(9):1195–1207.

[153] V. Sindhwani et al. (2004). Feature selection in mlps and svms based on maximum output information. *IEEE Trans. On Neural Networks,* **15**:937–949.

[154] P. Sinha (1994). Object recognition via image invariants: A case study. *Investigative Ophtalmology and Visual Science* **35**(4):1735–1740.

[155] L. Sirovich, M. Kirby (1987). Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America* **4**(3):519–524.

[156] D. Skalak (1996). The sources of increased accuracy for two proposed boosting algorithms. In *Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms (AAAI)*.

[157] J. Smith (1988). *Decision Analysis: A Bayesian Approach*. Chapman and Hall.

[158] K. K. Sung, T. Poggio (1998). Example-based learning for view-based human face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* **20**(1):39–51.

[159] D. Tao, X. Tang, X. Li, X. Wu (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence* **28**(7):1088–1099.

[160] A. N. Tikhonov, V. Y. Arsenin (1977). *Solutions of Ill-Posed Problems*. Winston and Sons, Washington, DC.

[161] M. Turk, A. Pentland (1991). Eigenfaces for recognition. *Journal of Cognitive Neuro Science* **3**:71–86.

[162] L. G. Valiant (1984). A theory of the learnable. *Commun. ACM* **27**(11):1134–1142.

[163] V. Vapnik (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York.

[164] V. Vapnik (2000). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

[165] P. Viola, M. Jones (2001). Rapid object detection using a boosted cascade of simple features. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[166] P. A. Viola, M. J. Jones (2004). Robust real-time face detection. *International Journal of Computer Vision* **57**(2):137–154.

[167] A. R. Webb. (2002). *Statistical Pattern Recognition*. John Wiley and Sons Ltd.

[168] D. Whitley (1994). A genetic algorithm tutorial. *Statistics and Computing* **4**:65–85.

[169] T. Windeatt (2005). Diversity measures for multiple classifier system analysis and design. *Information Fusion* **6**:21–36.

[170] T. Windeatt (2006). Accuracy/diversity and ensemble MLP classifier design. *Trans. on Neural Networks* **17**(5):1194–1211.

[171] D. H. Wolpert (1992). Stacked generalization. *Neural Networks* **5**:241–259.

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

[173] B. Wu, H. Ai, C. Huang, S. Lao (2004). Fast rotation invariant multi-view face detection based on real adaboost. In *International Conference on Automatic Face and Gesture Recognition (FGR)*, pp. 79–84.

[174] H. Wu, T. Yokoyama, D. Pramadihanto, M. Yachida (1996). Face and facial feature extraction from color image. In *International Conference on Automatic Face and Gesture Recognition (FGR)*, pp. 345–350.

[175] G. Yand, T. S. Huang (1994). Human face detection in a complex background. *Pattern Recognition* **27**(1):53–63.

[176] J. Yang, A. Waibel (1996). A real-time face tracker. In *Workshop on Applications of Computer Vision*, pp. 142–147.

[177] M. Yang, D. Kriegman, N. Ahuja (2002). Detecting faces in images: a survey. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)* **24**(1):34–58.

[178] M.-H. Yang, D. Roth, N. Ahuja (1999). A SNoW-based face detector. In *Neural Information Processing Systems (NIPS)*, pp. 862–868.

[179] M. H. Yang, D. Roth, N. Ahuja (2002). A tale of two classifiers: SNoW vs. SVM in visual recognition. In *European Conference on Computer Vision*, pp. 685–693.

[180] A. L. Yuille, D. S. Cohen, P. W. Hallinan (1989). Feature extraction from faces using deformable templates. *International Journal of Computer Vision* **8**(2):104–109.

[181] G. Yule (1903). On the association of attributes in statistics. *Biometrika* **2**:121–134.