University of Coimbra
Faculty of Sciences and Technology
Department of Electrical and Computer Engineering

# STUDY ON NON-PARAMETRIC METHODS FOR FAST PATTERN RECOGNITION WITH EMPHASIS ON NEURAL NETWORKS AND CASCADE CLASSIFIERS.

Oswaldo Ludwig

November 2011

University of Coimbra
Faculty of Sciences and Technology
Department of Electrical and Computer Engineering

# STUDY ON NON-PARAMETRIC METHODS FOR FAST PATTERN RECOGNITION WITH EMPHASIS ON NEURAL NETWORKS AND CASCADE CLASSIFIERS.

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING
OF COIMBRA UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Oswaldo Ludwig

UNDER SUPERVISION OF
Professor Urbano J. Carreira Nunes (supervisor)
Professor Rui Alexandre de Matos Araújo (co-supervisor)

# Resumo

Esta tese concentra-se em reconhecimento de padrões, com particular ênfase para o conflito de escolha entre capacidade de generalização e custo computacional, a fim de fornecer suporte para aplicações em tempo real. Neste contexto são apresentadas contribuições metodológicas e analíticas para a abordagem de dois tipos de *datasets*: balanceados e desbalanceados. Um *dataset* é denominado balanceado quando há um número aproximadamente igual de observações entre as classes, enquanto *datasets* que têm números desiguais de observações entre as classes são denominados desbalanceados, tal como ocorre no caso de detecção de objetos baseada em imagem. Para *datasets* balanceados é adoptado o perceptrão multicamada (MLP) como classificador, uma vez que tal modelo é um aproximador universal, ou seja MLPs podem aproximar qualquer conjunto de dados. Portanto, ao invés de propor novos modelos de classificadores, esta tese concentra-se no desenvolvimento e análise de novos métodos de treinamento para MLP, de forma a melhorar a sua capacidade de generalização através do estudo de quatro abordagens diferentes: maximização da margem de classificação, redundância, regularização, e transdução. A idéia é explorar novos métodos de treino para MLP com vista a obter classificadores não-lineares mais rápidos que o usual SVM com kernel não-linear, mas com capacidade de generalização similar. Devido à sua função de decisão, o SVM com kernel não-linear exige um esforço computacional elevado quando o número de vetores de suporte é grande. No contexto dos *datasets* desbalanceados, adotou-se classificadores em cascata, já que tal modelo pode ser visto como uma árvore de decisão degenerativa que realiza rejeições em cascata, mantendo o tempo de processamento adequado para aplicações em tempo real. Tendo em conta que conjuntos de classificadores são susceptíveis a ter alta dimensão VC, que pode levar ao *over-fitting* dos dados de treino, foram deduzidos limites para a capacidade de generalização dos classificadores em cascata, a fim de suportar a aplicação do princípio da minimização do risco estrutural (SRM). Esta tese também apresenta contribuições na seleção de características e dados de treinamento, devido à forte influência que o pre-processamento dos dados tem sobre o reconhecimento de padrões. Os métodos propostos nesta tese foram validados em vários *datasets* do

banco de dados da UCI. Alguns resultados experimentais já podem ser consultados em três revistas da ISI, outros foram submetidos a duas revistas e ainda estão em processo de revisão. No entanto, o estudo de caso desta tese é limitado à detecção e classificação de peões.

# Abstract

This thesis focuses on pattern recognition, with particular emphasis on the trade off between generalization capability and computational cost, in order to provide support for on-the-fly applications. Within this context, two types of datasets are analyzed: balanced and unbalanced. A dataset is categorized as balanced when there are approximately equal numbers of observations in the classes, while unbalanced datasets have unequal numbers of observations in the classes, such as occurs in case of image-based object detection. For balanced datasets it is adopted the multilayer perceptron (MLP) as classifier, since such model is a universal approximator, i.e. MLPs can fit any dataset. Therefore, rather than proposing new classifier models, this thesis focuses on developing and analysing new training methods for MLP, in order to improve its generalization capability by exploiting four different approaches: maximization of the classification margin, redundancy, regularization, and transduction. The idea is to exploit new training methods for MLP aiming at an nonlinear classifier faster than the usual SVM with nonlinear kernel, but with similar generalization capability. Note that, due to its decision function, the SVM with nonlinear kernel requires a high computational effort when the number of support vectors is big. For unbalanced datasets it is adopted the cascade classifier scheme, since such model can be seen as a degenerate decision tree that performs sequential rejection, keeping the processing time suitable for on-the-fly applications. Taking into account that classifier ensembles are likely to have high VC dimension, which may lead to over-fitting the training data, it were derived generalization bounds for cascade classifiers, in order to support the application of structural risk minimization (SRM) principle. This thesis also presents contributions on feature and data selection, due to the strong influence that data pre-processing has on pattern recognition. The methods proposed in this thesis were validated through experiments on several UCI benchmark datasets. Some experimental results can be found in three ISI journals, others has been already submitted to two ISI journals, and are under review. However, the case study of this thesis is limited to pedestrian detection and classification.

# Acknowledgment

I would like to express my gratitude to my advisor, Prof. Dr. Urbano Nunes, for his support, encouragement, and guidance through out my Ph.D studies; to my co-advisor, Prof. Dr. Rui Araujo, for his support and for his review of this thesis; to Profa. Dra. Bernardete Ribeiro for her review on the statistical machine learning contents; and to my colleague Cristiano Premebida for his friendly support in the experiments on pedestrian detection.

I also wish to express my gratitude for the Institute of Systems and Robotics, University of Coimbra (ISR-UC) for its logistic support, and for Fundação para a Ciência e a Tecnologia de Portugal (FCT), for its support to this thesis, under grant SFRH/BD/44163/2008, and its support to our research group, under project grants PTDC/SEN-TRA/099413/2008 and PTDC/EEA-AUT/113818/2009.

I dedicate this thesis to my parents, who gave me life and education, to my wife for her unconditional support during this endeavor, and to my daughter, who gave me happiness, teaching me what is really important.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introdution

IN the field of pattern recognition there are four main problems to work on: a suitable composition for the training dataset, feature extraction and/or selection, the classifier model, and the training method. This thesis addresses these four items, with particular emphasis on the last problem. Namely, this work proposes new algorithms for feature and data selection, analyzes cascade classifier ensembles, and proposes new training methods for Multilayer Perceptrons (MLP) and cascade classifiers, starting from the mathematical foundations to real engineering applications, passing through methodological contributions.

## 1.1 Thesis scope

Pattern recognition can be categorized according to the classification model or learning procedure. Regarding the classification model, there are parametric and non-parametric approaches, while the learning procedure can be supervised, unsupervised, or semi-supervised. This thesis deals with non-parametric methods trained through supervised and semi-supervised methods. Parametric pattern recognition requires the knowledge of the probability density functions or the assumption of a priori distribution, whose parameters must be estimated before the application of some inference technique, such as Bayesian decision theory. On the other hand, non-parametric pattern recognition makes no assumptions regarding the distributions of the features. However, the bayesian framework is not applicable in the case of non-parametric methods which demands analysis tools based on statistical learning theory [Vap98], in order to place bounds on the performance of the studied methods. Therefore, this thesis applies the framework of statistical learning theory to support the efectiveness of the proposed methods, which are also empirically evaluated.

## 1.2 Objectives and approach

The aim of this thesis is to offer a fast and accurate nonlinear classification for both balanced and unbalanced datasets. A dataset is categorized as balanced when there are approximately equal numbers of observations in the classes, while unbalanced datasets have unequal numbers of observations for each class. Image-based object detectors, such as pedestrian detection, have to face strongly unbalanced datasets, since each image frame is scanned by a slide window detector at different scales and positions, generating several negative bounding-boxes per each pedestrian cut-out. For such kind of unbalanced problem, it was addopted the cascade classifier scheme, since such model can be seen as a degenerate decision tree which performs a sequential rejection, by combining classifiers in a cascade structure, keeping the processing time suitable for on-the-fly applications.

Taking into account that classifier ensembles, such as cascade classifiers, are likely to have high VC-dimension, that may lead to over-fitting the training data, their generalization bounds are derived, in order to support the application of structural risk minimization (SRM) principle.

In case of balanced datasets, it is adopted MLP as classifier, since such model is one of the most known universal approximators [HSW90], i.e. MLPs can fit any dataset. Therefore, rather than propose new classifier models, this thesis focuses on the development and analysis of four new training methods for MLP, in such a way to improve its generalization capability. The idea is to offer nonlinear classifiers faster than the usual nonlinear support vector machine (SVM), with similar generalization capability. Notice that, due to its recursive decision function, the SVM with nonlinear kernel demands high computational effort when the number of support vectors is high.

Excepting a training method based on the well investigated maximal-margin principle [Vap98], all the MLP training methods proposed in this thesis are theoretically investigated based on the statistical learning theory or by associating with the maximal margin principle.

Regarding the empirical analysis, the cascade classifier was evaluated on pedestrian detection, while the new training methods for MLP were evaluated on pedestrian classification.

## 1.3 Contributions

One of the problems that occur during supervised training is called overfitting. The error on the training dataset is driven to a small value, however the error is large when new data are presented to the classifier. It occurs because the classifier memorizes the training examples, i.e., the classifier does not learn to generalize to new situations. To deal with this kind of problem this thesis proposes and analyses new methods for improving generalization capability whitout decreasing the classification speed. Namely, this thesis includes theoretical, methodological, and empirical contributions in this field, published or submitted for publication in ISI journals, which are summarized in Table 1.1. The case study of this thesis is limited to pedestrian classification and detection; however the methods proposed in this thesis were validated through experiments on several UCI benchmark datasets. Some experimental results can be found in three ISI journals, others has been already submitted to two ISI journals, and are under review. In this thesis two datasets were applied: the Daimler Pedestrian Classification benchmark [MG06] and a Dataset[1] previouly collected in the scope of the research project Perception Methods for an Intelligent Transportation System using On-Board Sensing [LPNR11].

## 1.4 Thesis outline

This thesis is organized as follows. Chapter 3 approaches data pre-processing, by introducing new algorithms for feature and data selection. Chapter 4 presents new training algorithms for MLP training that are also mathematically analyzed. Chapter 5 introduces a mathematical analysis on cascade classifier, based on the statistical

[1]http://webmail.isr.uc.pt/ cpremebida/dataset

Table 1.1: Contributions

| **Contribution** | **Scope** | **Chapter** | **Publications** |
|---|---|---|---|
| New feature selector based on genetic algorithm and information theory; | methodological | 3.1 | [LNA$^+$09], [LN10] |
| Data selector to compose balanced datasets; | methodological | 3.2 | - |
| MMGDX, a new maximum-margin training method; | methodological | 4.1 | [LN10] |
| New training method based on redundancy; | methodological | 4.2 | - |
| Proof that redundancy improves the generalization capability of multilayer perceptrons; | theoretical | 4.2 | - |
| Analysis on a new regularization technique, named eigenvalue decay; | theoretical | 4.3 | [LN11] |
| New training method based on regularization and genetic algorithm; | methodological | 4.3 | [LN11] |
| New transductive training method for neural networks; | methodological | 4.4 | [LN11] |
| Cascade classifier scheme based on the principle of minimization of structural risk; | methodological | 5.3 | [LPNR11] |
| Bounds on the expected true positive rate, false positive rate, and BER; | theoretical | 5.4 | [LPNR11] |
| Bounds on the expected risk of cascade classifiers; | theoretical | 5.4 | [LPNR11] |
| WERM, a new training method specially developed for cascade classifiers; | methodological | 5.6 | - |
| Experimental study on a new LIDAR/vision-based pedestrian detection dataset; | empirical | 6.3 | [LPNR11] |
| Three computational applications (protected by the BSD license) available to the scientific community through the MATLAB Central. | comput. applic. | 3.1, 4.1, 6.1 | - |

learning framework, in such a way to offer a fast and effective approach for highly unbalanced datasets. In Chapter 6 the methods and algorithms proposed here are validated through applications in pedestrian detection and classification. Finally, Chapter 7 concludes this thesis, giving directions for future works.

# Chapter 2

# State of the art

TH is thesis proposes new training methods for two usual non-linear models: MLP and cascade of linear classifiers, which were chosen due to their suitability to on-the-fly applications, since such models can perform nonlinear classification under low computational effort. Therefore, the idea is to improve the generalization capacity of such models through the training methods.

MLP is one of the most known universal approximators, as stated by Hornik *et al.* [HSW90], who shown that a MLP with one sigmoidal hidden layer and linear output layer can fit any dataset, because the sigmoidal hidden units of MLP compose a basis of linearly independent soft functions. Despite the simplicity of MLP, experiments with real-world benchmark datasets [LN10], [HCH$^+$04], [Sam04], give evidence that an adequate training can lead the MLP to achieve performance which is better than (or at least similar) as other state-of-the-art approaches, such as Bayesian Neural Network [NZ06], novel algorithms based on Kernel Fisher Discriminant analysis [MRW$^+$99], or Support Vector Machines (SVM) with non-linear kernels [Liu04].

This thesis exploits four different approaches in improving the generalization capability of MLP: by classification-margin maximization, by redundancy, by regularization, and by transduction; however, there are other approaches in improving the generalization of NN, such as selective sampling [CAL94] or noise injection [Mat92]. Actualy, generalization is one of the most widely studied problems in machine learning. A mathematical formalism for the generalization problem was proposed in [Val84]; however, the study of distribution-free learnability, more specifically, non-parametric methods for pattern recognition, started earlier with the pioneer work [VC71], which studied the uniform convergence of relative frequencies of events to their probabilities as function of the hypothesis space complexity and the cardinality of the training dataset. Then the work [BEHW89] introduced the VC-dimension, a simple combinatorial parameter of the class of concepts to be learned that measures the capacity of the set of hypothesis, in order to deal with infinite hypothesis spaces. Such parameter enable to place bounds on the expected classification risk, even in case of infinite hypothesis spaces [Vap98]. The interest in statistical learning theory led to an increasing effort in view to establish bounds on the VC-dimension of non-discrete concept classes in the context of multilayer perceptrons. In this sence, it is important to highlight the work [BH89], which investigated the optimal neural architecture in terms of generalization capability, by studying the order of growth of the VC-dimension of the neural network hypothesis space. This research line also yielded the work [KM95], which places polynomial bounds on the complexity of the neural network hypothesis space. In [CAL94] the authors propose to improve generalization by selective sampling, in order to differentiate a region of uncertainty from the bulk of the domain. A new interesting research line on the generalization capability of neural networks was introduced in [YNW$^+$07], which proposed a local generalization error model that gives upper bounds on the expected classification error of unlabeled examples within a neighborhood of the training examples, differently from the usual approaches that provide generalization bounds intended for the entire input space. The proposed error model is based on a stochastic sensitivity measure, which is an interesting alternative to the most known approaches on statistical learning, such as the bias-variance tradeoff and the VC-style analysis, which is based on the classifier-space complexity that is difficult to estimate, excepting in the case of linear models. However, Yeung *et al.* [YNW$^+$07] do not present a comparative study between the proposed generalization bounds and the usual ones, which could justify some disadvantages of the proposed analysis, such as the restrictive assumption that the unlabeled examples lie within the neighborhood of the training examples, as well as the assumption of uniform distribution for the input pertubations. In summary, that study is still an open field, in the sense that it requires a proof that the proposed bounds are tighter than the usual ones. Moreover, the comparison of the proposed analysis with the analysis on the transductive learning introduced in [Vap82] and developed in [Vap98] reveals an advantage of the last approach, since it does not assumes that the unlabeled examples lie within the neighborhood of the training examples.

During almost three decades statistical learning theory was a purely theoretical analysis of the problem of function estimation from a given collection of data [Vap99]. However, in 1992 statistical learning theory became not only a tool for the theoretical analysis but also a tool for creating practical algorithms for non-parametric pattern recognition. Namely, the maximum margin principle, which underlies the SVM, was introduced in [BGV92], where it was proposed a generic training algorithm that maximizes the margin between the training patterns and the decision boundary. The works [DGC07] and [Abe05] extended the maximum margin principle to NN training. In [DGC07] a decision tree based on linear programming is applied to maximize the margins, while in [Abe05] an MLP is trained layer by layer based on the CARVE algorithm [YD98]. Motivated by the success of large margin methods in supervised learning, some authors extended large margin methods to unsupervised learning [ZBS07]. In this thesis the maximum margin principle is applied to a new training method that jointly optimizes both MLP layers in a single process, back-propagating the gradient of an maximum-margin based objective function, through the output and hidden layers, in order to create a hidden-layer space that enables a higher margin for the output-layer separating hyperplane. The proposed maximum margin based objective function aims to stretch out the margin to its limit by applying an objective function based on Lp-norm, in order to take into account the idea of support vectors, however, overcoming the complexity involved in solving a constrained optimization problem, usual in SVM training.

As regards redundancy, studies on animal physiology have shown evidences of neurophysiological nervous redundancy in biological systems [KDMC83], such as the monkey visual cortex, see [SL86]. This physiological evidence has been complemented by some empirical studies on redundancy in artificial neural networks. This thesis complements some founding works in redundancy applied to NN, such as [IP90], by presenting a theoretical analysis, in the statistical learning framework, that shows the positive effects of redundancy on the upper-bound on the expected classification risk. Moreover, it is proposed a redundant artificial neural network that can be understood as an ensemble of small neural networks, which are trained independently and aggregated into an usual MLP with two hidden layers.

Another commonly used technique in improving the generalization is regularization, since it prevents the learning algorithm from overfitting the training data. There are three usual regularization techniques for NN: early stopping [TG99], curvature-driven smoothing [Bis96], and weight decay [Jin04]. In the early stopping criterion the available data are divided into three subsets. The first subset is the training dataset, which is used for updating the network weights and biases. The second subset is used as a validation dataset and the third subset is used to evaluate the final accuracy. The error on the validation dataset is monitored during the training process. After some number of iterations the NN begins to overfit the data and the error on the validation dataset begins to rise. When the validation error increases during a specified number of iterations, the algorithm stops the training section and applies the weights and biases at the minimum of the validation error to the NN. In [AMM+97] it was proposed an asymptotic theory for neural network overtraining, in order to study the gain in the generalization error when performing early stopping and cross-validation stopping. This work also provides a theoretical framework to estimate the optimal split between training and validation examples. Curvature-driven smoothing includes smoothness requirements on the cost function of learning algorithms, which depend on the derivatives of the network mapping. Weight decay is implemented by including additional terms in the cost function of learning algorithms, which penalize overly high values of weights and biases, in order to control the classifier complexity, which forces the NN response to be smoother and less likely to overfit. The work [Bar98] gave theoretical support for the usual weight decay training method based on statistical learning theory. This thesis introduces a new regularization scheme, named eigenvalue decay. This approach aims at improving the classification margin, as will be showed. This regularization scheme led to the development of a new training method for NN based on the same principles of SVM.

The work [Vap82] has introduced the transductive setting in the context of statistical learning. Transductive learning is based on the idea that prior knowledge carried by the unlabeled testing dataset can be learned by an algorithm, potentially leading to superior performance. Transduction is a concept closely related to semi-supervised learning. However, differently from inductive inference, no general decision rule is inferred. In the case of transduction the inferred decision rule aims only at the labels of the unlabeled testing data. This thesis introduces a transductive NN, which is similar to the transductive SVM [Vap98], in the sense that our method also exploits the geometric structure in the feature vectors of the test examples, by taking into account the principle of low density separation, which states that the decision boundary should lie in a low-density region of the feature space.

In [GBD92] the generalization problem is formulated in the framework of the bias-variance dilemma, which decomposes the expected classification error into a bias term and a variance term. Assuming an infinite supply of independent training datasets, the bias term measures how closely the learning-algorithm average guess matches the target (averaged over all training datasets). The variance term measures how much the learning-algorithm guess bounces-around for the different training datasets, that is, it measures how consistent the classifier decisions are. Such analysis shows that a classifier space with a high capacity, i.e. high VC-dimension, is likely to have low bias, but large variance. On the other hand, a classifier space with a low capacity usualy have a low variance but a large bias. The works [KD95] and [KW96] extended the analysis introduced in [GBD92] in such a way to deal with the usual zero-one loss functions. The introduction of the the bias-variance dilemma supported the idea that by composing an ensemble of multiple classifiers it is possible to reduce the variance term without affecting the bias term, some works, such as [Kun02] and [IYM03], have provided empirical evidence that an ensemble of classifiers is often better than single classifiers. One of the most known classifier enssemble is the AdaBoost [FS95], a method derived from the multiplicative weight-update technique proposed in [LW94]. Both works, [LW94] and [FS95], have provided theoretical analyzes for their algorithms; however, a more comprehensive analysis on classifier ensembles is presented in [SFBL98], which introduces bounds on the expected risk of classifier compositions. However, the main theorem of [SFBL98], Theorem 2, is based on a restrictive assumption that was introduced only in the proof, when it should be included in the theorem statement. Namely, the number of component-classifiers is assumed as function of the component-classifier space VC-dimension[1]. This thesis also deals with classifier ensembles; however, it focuses only on the cascade classifier, because such classifier ensemble is especially important for machine vision applications, such as object detection, since it is possible to combine successively more complex classifiers in a cascade structure, focusing the attention on promising regions of the image, saving processing time. Apart of the usual approaches, we highlight the seminal work of Viola and Jones [VJ01], where they have proposed a boosted cascade classifier scheme, which can be viewed as an object specific focus-of-attention mechanism that discards

[1]See the last line of the proof of Theorem 2 of [SFBL98], where it is stated: "setting $N = \lceil (4/\theta) \ln(m/d) \rceil$ completes the proof". Notice that, in the context of that work, $d$ denotes the VC-dimension of each component-classifier space and $N$ denotes the number of component-classifiers.

regions which are unlikely to contain the objects of interest. Another interesting contribution was reported in [BP02], that proposed a cascade classifier scheme based on a specific formulation of the expectation-maximization (EM) algorithm, which allows the unsupervised estimation of both the class-conditional density functions and the prior joint probabilities of classes. The proposed technique also allows to include in the estimation process additional prior information.

Similarly to other classifier ensembles, cascades of linear classifiers are likely to have high VC dimension, which may lead to over-fitting the training data, according to the principles introduced in [Vap98] that established a trade-off between the empirical risk and the expected risk, which can be optimized by controling the classifier space complexity, i.e. its VC dimension. Actually, [Vap98] states that the larger the VC dimension, the larger the probability of success at fitting the training data; however, the larger the probability that the expected risk will deviate from the empirical risk. Therefore, in order to improve the generalization capacity of cascades of linear classifiers, this thesis proposes the SRM-cascade, which is based on the structural risk minimization (SRM) principle [Vap98], an inductive principle for model selection that balances the model complexity, i.e. the VC dimension of the ensemble model, against its success at fitting the training data, which corresponds to finding the simplest model in terms of VC dimension and best in terms of empirical error. Our preliminary works on SRM-cascade [LPNA11] applied the SRM principle independently on each cascade stage. Such method can be improved by estimating the strutural risk of the entire ensemble. However, despite the methodological [GB00] and experimental [SK07] contributions given in previous works, there is still a lack of theoretical analysis on the generalization capability of cascade classifiers; differently from the bagging strategy, which was theoretically analyzed by previous works such as [SFBL98] or [KWD03]. Therefore, this thesis contributes with a theoretical analysis, based on statistical learning theory, providing bounds on the false positive rate (FP) and true positive rate (TP), in such a way as to compose the upper-bound on the expected classification risk for the entire cascade ensemble.

This thesis focuses specifically on cascade of linear SVMs, despite the kernel mapping technique, which was introduced by [CV95] in order to construct nonlinear SVMs by mapping input vectors to a high-dimensional feature space, where a linear decision surface is constructed. However, in [Bur98] the authors show how SVM with polynomial and Gaussian radial basis function kernels can have very large, and even infinite, VC dimension, which can affect the generalization performance. Moreover, in the context of some on-the-fly applications, the use of SVM with nonlinear kernels may lead to a prohibitive computational cost, since its decision function requires a recursive calculation that demands a large amount of time when the number of support vectors is big. Moreover, linear classifiers offer the possibility of controling their VC dimension, enabling the application of SRM principle. Taking into account that the stages of cascade classifiers must be adjusted so that the TP is close to one, it is also proposed a new training method for linear classifiers that enables the control of the relationship between TP and FP during the training, in order to avoid the usual threshold adjustment, which often over-penalizes the classifier accuracy.

As a case study, the proposed training methods are applied to pedestrian classification [MG06] and pedestrian detection [LPNR11], aiming at improving the detection rate in outdoor environment. Autonomous ground vehicles navigating in environments with static and moving objects around should be provided with perception systems capable to detect and classify the objects of surroundings, in order to avoid collisions and to mitigate situations of risk during the navigation. In this context, protection systems for pedestrian safety is an emerging area of Advanced Driver Assistance Systems (ADAS) which achieved a notable development in the last decade. It is a still growing research field, evidenced by recent projects, challenges [DAR03],[ELR06], and recent publications [DWSP09],[EG09]. For instance, in the last years, two significant surveys in pedestrian detection and protection systems, in the context of Intelligent Vehicles (IV) and Intelligent Transportation Systems (ITS), were published in [GT07],[GLSG10]. In our experiments, the detection system receives information from both LIDAR and monocular camera, in order to bring redundancy and complementary characteristics that can improve the system reliability and its level of inference. This redundant approach is a typical solution adopted by IV-ITS research community, as can be seen in [DFR07], where geometrical information from the LIDAR data is used, not only to classify the objects as vehicles or non-vehicles, but also to provide range information that is usefull to deal with the problem of object scale variations in the images. In [HCR$^+$07] regions of interest, detected by the LIDAR, are subdivided in five areas, where different trained SVMs are employed to classify vehicles. In [MSRD06] an Adaboost, using Haar-like features, also classifies regions of interest detected by the a single-layer LIDAR, while in

[SSO06] the authors apply a convolutional neural network to classify regions of interest provided by a multi-layer LIDAR. The work [SS08] applies HOG/SVM based on monocular color images and multi-layer LIDAR data to detect pedestrians, while in [FC07] a context-based approach is used to differentiate between static and dynamic on-road objects, which are detected by the LIDAR and distinguished based on their dynamic behavior and some geometrical-features constraints.

# Chapter 3

# Data pre-processing

THIS chapter introduces two data pre-processing algorithms. The first algorithm is a feature selector, while the second one is a data selector.

Feature selection is a usual pre-processing procedure which aims at speeding up learning process, improving model interpretability, while decreasing the classifier complexity (in Vapnik sense), in order to enhance the generalization capability, which depends on the interrelationship between the sample size, the number of features, and the classifier space complexity. The interrelationship between the sample size and the classifier space complexity was modeled in [Vap98]. On the other hand, the interrelationship between the number of features and the classifier space complexity depends on the addopted classifier model, e.g. in case of linear classifiers, such as the classifiers that compose the cascade ensemble applied in this thesis, the VC dimension is directly proportional to the number of features [BEHW89], while in case of MLPs, the VC dimension grows with the square of the number of weights [KS97]; therefore, since the number of weights grows linearly with the number of inputs, i.e. the number of features, it is possible to state that the MLP space complexity grows with the square of the number of features, which may lead the MLP to overfit the training data. Moreover, it has been often observed in practice that an overly high number of features may degrade the performance of a classifier if the number of training samples that are used to design the classifier is small relative to the number of features. Feature selection algorithms can be dichotomized into two categories: feature ranking and subset selection. Feature ranking ranks the features by a metric and selects the features which achieve a threshold. Subset selection optimizes an objective function aiming at finding the optimal subset of features. The algorithm proposed in this chapter falls on the second category, being a development of the well known feature ranking algorithm proposed in [PLD05]. The feature selector proposed in this chapter optimizes an objective function based on information theory by means of evolutionary computation.

## 3.1 Feature selector by GA

The algorithm presented in this section was proposed in our previous work [LN10] and is available for download at Matlabcentral[1].

The selection of a suitable set of features can be achieved by maximizing the mutual information $I(y; x_1, \ldots, x_N)$ between the target $y$ (i.e. the class label) and the set of features $\{x_i, \ldots, x_N\}$ which compose the input vector $X$. However, this procedure demands a high computational effort, due to the required calculation of joint-entropy values, such as $H(x_i, \ldots, x_N, y)$, which requires the estimation of the joint-distribution $p(x_i, \ldots, x_N, y)$. In order to decrease such computational cost, the present work applies a classifier selection criterion based on the principle of minimal-redundancy-maximal-relevance (mRMR) [PLD05], which maximizes the mutual information $I(y; x_1, \ldots, x_N)$ indirectly[2], by jointly minimizing a measure of redundancy, which is averaged on all the features, and maximizing a measure of discriminant power, which is also averaged on all the features, as explained below.

Let $V$ be the relevance of a set of $N$ features, i.e., the mean value of the mutual information $I(x_i; y)$ between each feature and the label,

$$V = \frac{1}{N} \sum_{i=1}^{N} I(x_i; y) \tag{3.1}$$

and $P$ be the redundancy, i.e., the mean value of the mutual information $I(x_i; x_j)$ among features,

[1]http://www.mathworks.com/matlabcentral/fileexchange/29553-feature-selector-based-on-genetic-algorithms-and-information-theory

[2]in [PLD05] there is a proof of the equivalence between directly maximizing the mutual information and the the mRMR approach.

$$P = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} I(x_i; x_j) \tag{3.2}$$

Therefore, the application of the mRMR principle corresponds to searching a set of features to satisfy the maximization problem

$$\max_{x_{i_1} \dots x_{iN}} \Phi \tag{3.3}$$

subject to:

$$i_k \neq i_z \texttt{ for } k \neq z \tag{3.4}$$

where $\Phi = V - P$. The constraint (3.4) was introduced in order to avoid repeated features. This constraint is cheked in steps 22-25 of Algorithm 1. The extreme value of (3.3) is attained when the features $x_i$ are totally correlated to the target output $y$ and mutually exclusive between them.

From (3.1) and (3.2) it is possible to conclude that mRMR only requires low-cost calculations, i.e., values of mutual information that are calculated over couples of variables. However, the combinatorial optimization problem described by (3.3) and (3.4) has

$$0.5 \frac{M!}{(M-N)!} \tag{3.5}$$

possible solutions, where $M$ is the number of available features. In order to decrease the computational effort necessary to checking all the possibilities, the work [PLD05] proposed an incremental search method that achieves a near-optimal solution. For instance, due to the incremental approach, the method proposed in [PLD05] always selects the feature which has the largest relevance, i.e. $I(y; x)$, independently from its redundancy, i.e. $I(x; x_1, \dots, x_N)$. To be more specific, the most relevant feature is always the first feature which is selected. However, depending on the redundancy, it is possible the existence of an optimal set of features that does not contain the most relevant feature. Therefore, as a development of [PLD05], we propose to perform the combinatorial optimization of $I(y; x_1, \dots, x_N)$ by means of Genetic Algorithm (GA), similarly to the indirect approach described in our previous works [LNA$^+$09], [LN10]. However, the crossover operation was modified in order to deal with combinatorial optimization problems, namely, each gene of a new individual is taken from one of the parents in a random process. The Algorithm 1 details the feature selection which requires statistical information supplied by Algorithm 2. The proposed crossover operation is detailed in steps 10-27 of Algorithm 1, while the mRMR fitness function is represented by steps 5 and 6.

## 3.2 Training data selector

The interest in training data selection increased with the applications of SVM with non-linear kernels in real world problems, since such algorithms are based on a mathematical programming problem that requires a large computational effort to be solved, even for moderately sized datasets. Moreover, due to its decision function, the SVM with nonlinear kernel also requires a high computational effort to classify new data, when trained on large datasets, since the number of support vectors is related to the number of training examples.

Regarding the previous works on data selection, it is important to highlight [BdPB00] which employed k-means clustering to select particularly salient data points from the training dataset. In [AI01] the Mahalanobis distance is applied to estimate boundary points. The work [LM01] introduced the reduced SVM setting, which chooses a subset of training examples by using random sampling, while [SC03] proposed a method that selects patterns near the decision boundary based on neighborhood properties.

Image-based object detection using multi-scale sliding window, such as the pedestrian detection algorithm applied in our experiments, may generate thousands of negative cropped images for each positive occurence. Therefore, to avoid bias problems and unfeasible computational requirements in such large unbalanced datasets, we propose an algorithm which composes balanced training datasets from unbalanced ones. Namely, the proposed algorithm selects the hard negative examples, i.e. negative samples which are likely to be near the decision boundary, in order to preserve the information which is relevant to compute the classifier separating hypersurface. The idea is to provide support to the training of neural networks, not only in case of image-based object detection, but also in case of multi-class classification, where the training dataset is usually highly unbalanced, which leads training methods based on mean squared error (MSE) to totally ignoring the minority class.

The data resampling algorithm proposed in this section selects negative training data based on the confidence measure suggested in [AGHR05], which is the number of negative training examples that are contained in the largest hypersphere centered at a negative example without covering a positive example, i.e. considering a sphere around a negative example, $x_{(neg,i)}$, that is as large as possible without covering a positive example, then the Confidence Measure of $x_{(neg,i)}$, here named $CM_i$, is calculated by counting the number of negative examples that fall inside this sphere. The smaller the value $CM_i$ the more likely $x_{(neg,i)}$ will be close to a decision boundary. Therefore, the process of sample selection, i.e. negative data selection, is solved by ranking the negative samples through an exhaustive search in their surrounding, as summarized in Algorithm 3, which has time complexity $O\left(n_n\left(n_n + n_p\right)\right)$.

**Algorithm 1** feature selector by GA

```
Input: $I(x_n;x_m)$ and $I(x_n;y_{train})$, $n = (1,\dots,M)$ and $m = (1,\dots,M)$: statistical data from
    Algorithm 2;
    $N$: desired number of features for the ensemble;
    $a$: selective pressure;
    $max_{gener}$: maximum number of generations;
    $N_{pop}$: population size
Output: $\{i\}$: set of indexes of the selected features
 1: generate a set with $N_{pop}$ chromosomes $\{Cr\}$ for the initial population, each chromosome is a
    vector $Cr = [i_1 \dots i_N]$ containing $N$ features indexes $i$ randomly generated without repeated
    elements;
 2: for $generation = 1 : max_{gener}$ do
 3:   evaluating the population:
 4:   for $ind = 1 : N_{pop}$ do
 5:     calculate $V$ and $P$ for the individual $Cr_{ind}$, according to (3.1) and (3.2), by means of
        the previously calculated mutual information values for all the elements (i.e. indexes) of
        chromosome $Cr_{ind}$;
 6:     $\Phi_{ind} \leftarrow (V - P)$: storing the fitness of each individual $ind$;
 7:   end for
 8:   rank the individuals according to their fitness $\Phi_{ind}$;
 9:   store the genes of the best individual in $\{i\}$;
10:   performing the crossover:
11:   $k \leftarrow 0$;
12:   for $ind = 1 : N_{pop}$ do
13:     $k \leftarrow k + 1$;
14:     randomly selecting the indexes of parents by using the asymmetric distribution
        proposed in [LNA+09]:
15:     $\vartheta_j$, $j = 1,2 \leftarrow$ random number $\in [0,1]$ with uniform distribution;
16:     $parent_j$, $j = 1,2 \leftarrow round\left(N_{pop}\frac{e^{a\vartheta_j}-1}{e^a-1}\right)$ [LNA+09];
17:     create the set $\{i_{abs}\}$, containing all the indexes which are absentees in both parents;
18:     assembling the chromosome $Cr_k^{son}$ :
19:     for $n = 1 : N$ do
20:       randomly select a parent (i.e. between $parent_1$ and $parent_2$) to give the nth gene for the
          kth individual of the new generation:
21:       $Cr_{(k,n)}^{son} \leftarrow Cr_{(parent_{1or2},n)}$;
22:       considering the constraint (3.4):
23:       if there is duplicity of indexes in $Cr_k^{son}$ then
24:         pick up and remove from $\{i_{abs}\}$ a new index for $Cr_{(k,n)}^{son}$;
25:       end if
26:     end for
27:   end for
28: end for
```

**Algorithm 2** Mutual information calculation

```
Input: $\{x_1, \ldots, x_M\}$, $\{y_{train}\}$: dataset containing vectors with all the available features and their
    respective target outputs, regarding the training dataset
Output: $I(x_n; x_m)$ and $I(x_n; y_{train})$, $n = (1, \ldots M)$ and $m = (1, \ldots, M)$: statistical data
 1: calculate entropy $H(x_n)$ of each feature $n$ (details about the distribution generation in [PLD05]);
 2: calculate entropy $H(y_{train})$ of the target output;
 3: calculate the joint entropy $H(x_n, x_m)$ of each feature pair $(n, m)$;
 4: calculate the joint entropy $H(x_n, y_{train})$ of each feature $n$ and the target output;
 5: calculate the mutual information $I(x_n; y_{train}) = H(x_n) + H(y_{train}) - H(x_n, y_{train})$ between
    each feature $n$ and the target output;
 6: calculate the mutual information $I(x_n; x_m) = H(x_n) + H(x_m) - H(x_n, x_m)$ between elements
    of each feature pair $(n, m)$;
```

**Algorithm 3** training data selector

```
Input: $S = \{x_{(neg,1)}, \ldots, x_{(neg,n_n)}\} \cup \{x_{(pos,1)}, \ldots, x_{(pos,n_p)}\}$: set of negative and positive training
    examples, respectively;
Output: $\{x'_{(neg,1)}, \ldots, x'_{(neg,n_p)}\}$: set composed by $n_p$ negative examples, selected from
    $\{x_{(neg,1)}, \ldots, x_{(neg,n_n)}\}$;
 1: calculating $CM$ of all the $n_n$ negative examples:
 2: for $i = 1 : n_n$ do
 3:   searching the closest positive example, to determine the radius, $d_{min}$, of the hy-
      persphere centered at $x_{(neg,i)}$:
 4:   $d_{min} \leftarrow \infty$: setting the largest possible value for $d_{min}$;
 5:   for $j = 1 : n_p$ do
 6:     $d \leftarrow \left\|x_{(neg,i)} - x_{(pos,j)}\right\|$;
 7:     if $d < d_{min}$ then
 8:       $d_{min} \leftarrow d$;
 9:     end if
10:   end for
11:   counting the number of negative examples that fall inside the hypersphere whose
      radius is $d_{min}$:
12:   $CM_i = 0$;
13:   for $k = 1 : n_n$ do
14:     $d \leftarrow \left\|x_{(neg,i)} - x_{(neg,k)}\right\|$;
15:     if $d < d_{min}$ then
16:       $CM_i = CM_i + 1$;
17:     end if
18:   end for
19: end for
20: sort the examples according to their value of $CM$, from the smaller to the larger, composing the
    rearanged set $\{x'_{(neg,1)}, \ldots, x'_{(neg,n_n)}\}$;
21: pick up the first $n_p$ examples of $\{x'_{(neg,1)}, \ldots, x'_{(neg,n_n)}\}$.
```

# Chapter 4

# Improving the generalization capacity of MLP

SIMILARLY to other more complex neural models, such as Simultaneous Recurrent Neural networks (SRN) [IKW08], an MLP with one sigmoidal hidden layer and linear output layer is a universal approximator, because the sigmoidal hidden units of MLP compose a basis of linearly independent soft functions [HSW90]. Taking into account the simplicity of MLP, this thesis adopts this model for pattern recognition applications.

Regarding the classification speed, the MLP is faster than nonlinear SVM. Notice that, in the context of some on-the-fly applications, the use of SVM with nonlinear kernels may lead to a prohibitive computational cost, since its decision function requires a recursive calculation that demands a large amount of time when the number of support vectors is high. Therefore, this chapter proposes new training methods for MLP, in order to offer a fast nonlinear classification with enhanced generalization capacity. Particularly, four different approaches are exploited to improve the generalization capacity of MLPs:

- maximization of the classification-margin;
- redundancy;
- regularization;
- transduction.

## 4.1 Improving generalization by maximizing the classification margin

This chapter introduces a novel maximum-margin training method for MLP, based on the gradient descent with adaptive learning rate algorithm (GDX) and so named Maximum-Margin GDX (MMGDX)[1], which directly increases the margin of the MLP output-layer hyperplane. The proposed method jointly optimizes both MLP layers in a single process, back-propagating the gradient of an MM-based objective function, through the output and hidden layers, in order to create a hidden-layer space that enables a higher margin for the output-layer hyperplane, avoiding the testing of many arbitrary kernels, as occurs in case of SVM training. The proposed MM-based objective function aims to stretch out the margin to its limit. It is also proposed an objective function based on Lp-norm in order to take into account the idea of support vectors, however, overcoming the complexity involved in solving a constrained optimization problem, usual in SVM training. The training method proposed in this chapter has time and space complexities $O(N)$ while usual SVM training methods have time complexity $O(N^3)$ and space complexity $O(N^2)$, where $N$ is the training-dataset size.

MM-based training algorithms for neural networks (NN) are often inspired on SVM-based training algorithms, such as [DGC07] and [Abe05]. In [DGC07] a decision tree based on linear programming is applied to maximize the margins, while in [Abe05] an MLP is trained layer by layer based on the CARVE algorithm [YD98]. Motivated by the success of large margin methods in supervised learning, some authors extended large margin methods to unsupervised learning [ZBS07]. Besides early stopping criterion, our work also explores MM-based training algorithms. However, different from the SVM approach, in this section the concept of margin has an indirect relation with support vectors. Actually in this section, *margin is defined as the orthogonal distance between each pattern and the output-layer hyperplane.* Inspired on SVM-based training algorithms, a simple method that applies Lp-norm in order to take into account the idea of support vectors is proposed.

[1]MMGDX was made available for download at Matlabcentral, http://www.mathworks.com/matlabcentral/fileexchange/28749-mmgdx-a-maximum-margin-training-method-for-neural-networks

Table 4.1: Some variations of back-propagation algorithm

| **Method** | **Comments** |
|---|---|
| Levenberg Marquardt (LM) [Mor78] | This second order method is the fastest usual training algorithm for networks of moderate size. Despite of the large amount of memory needed, it has a memory reduction feature for use when the training set is large; |
| Levenberg Marquardt with Variable Projection (LMVP) [KL08] | The variable projection method reduces the dimension of the learning problem, and then the LM is applied to optimize the NN model by using a Jacobian matrix computed by a modified back-propagation algorithm; |
| Powell Beale conjugate gradient (CGB) [Pow77] | Less storage requirements than LM, also presents faster convergence; |
| BFGS quasi-Newton (BFG) | Requires storage of approximate Hessian matrix and has more computation in each iteration than the conjugate gradient algorithm, usually converges in fewer iterations; |
| Adaptive learning rate (GDX) [DB98] | Faster training than basic gradient descent, however, it can only be used in batch mode training; |
| Bayesian regularization (BR) [Wil95] | Bayesian regularization minimizes a linear combination of squared errors and weights in order to obtain a network with good generalization qualities. |

### 4.1.1 Gradient descent with adaptive learning rate

Besides some global search methods that have been applied in MLP training, such as genetic algorithms [Jin04], simulated annealing [SDJ99], or hybrid methods [LGL06], there are many variations of the back-propagation algorithm due to different approaches of the gradient descent algorithm, such as the methods which are commented in Table 4.1. The MATLAB Neural Network Toolbox offers some training algorithms, among them we highlight the Gradient Descent with momentum term and adaptive learning rate (GDX) [DB98], due to its application in this work.

The usual objective function of GDX is the MSE of

$$r_i = (y_i - \hat{y}_i) \tag{4.1}$$

where $y_i$ is the target output, $\hat{y}_i$ is the output estimated by the MLP for the input $x_i$ belonging to the training dataset, and $r_i$ is the error. Back-propagation is used to calculate the derivatives of the MSE functional with respect to the weight and bias. Each variable is adjusted according to the gradient descent with momentum term. For each epoch, if MSE decreases towards the goal, then the learning rate is increased by a given factor $\eta$. If MSE increases by more than a given threshold $\gamma$, the learning rate is decreased by a given factor $\mu$, and the updating of synaptic weights that increased the MSE is discarded.

### 4.1.2 Maximum-Margin GDX

This sub-section introduces the MMGDX, a new MM-based training algorithm where both MLP layers are jointly optimized in a single process. In fact, an MM-based objective function $J$ is back-propagated through the output and hidden layers in such a way as to create a hidden output especially oriented towards obtaining a larger margin for the output-layer separating-hyperplane. This methodology is different from other previous approaches, such as [Abe05] where the MLP is trained layer by layer. The unconstrained optimization problem

$$\min_{W_1,b_1,W_2} J \tag{4.2}$$

is applied to a MLP with one sigmoidal hidden layer and linear output layer, according to the following model:

$$\begin{aligned} yh &= \varphi\left(W_1 \cdot x + b_1\right) \\ \hat{y} &= W_2^T yh + b_2 \end{aligned} \tag{4.3}$$

where $yh$ is the output vector of the hidden layer, $W_l$ ($l$=1, 2) is the synaptic weights matrix of the layer $l$, $b_1$ is the bias vector of layer 1, $x$ is the input vector, and $\varphi\left(\cdot\right)$ is the sigmoid function. In MMGDX, the output layer of model (4.3) has bias $b_2 = 0$, because after the training section the ROC curve information is taken into account to adjust the classifier threshold, which acts as bias. The separating-hyperplane of model (4.3) is given by

$$W_2^T yh^{limit} = 0 \tag{4.4}$$

where $yh^{limit}$ is a point belonging to the hyperplane. Considering $yh^{proj}$ as the projection of point $yh$ on the separating-hyperplane (4.4) and $d$ as the distance between the separating-hyperplane (4.4) and $yh$, yields:

$$yh - yh^{proj} = d\frac{W_2}{||W_2||} \tag{4.5}$$

Multiplying both sides of (4.5) by $W_2^T$ yields:

$$W_2^T yh - W_2^T yh^{proj} = d\frac{W_2^T W_2}{||W_2||} \tag{4.6}$$

As $yh^{proj}$ belongs to hyperplane (4.4), substituting (4.4) and the second line of (4.3) in (4.6), yields:

$$d = \frac{\hat{y}}{||W_2||} \tag{4.7}$$

As the sigmoid activation function bounds the hidden neuron output in the interval $[0, 1]$, the norm of vector $yh$ has its maximum value equal to $\sqrt{n}$, where $n$ is the number of hidden neurons. Taking into account that the norm of $\frac{W_2}{||W_2||}$ is one, we can deduce that

$$-\sqrt{n} \leq \frac{W_2^T}{||W_2||} yh \leq \sqrt{n} \tag{4.8}$$

i.e. the distance $d$ (4.7) is bounded in the interval $[-\sqrt{n}, \sqrt{n}]$. Therefore, as the target output $y_i$ (where $i$ denotes the training example index) assumes the values -1 or 1, we propose the error function

$$e_i = \left( y_i\sqrt{n} - \frac{\hat{y}_i}{||W_2||} \right) \tag{4.9}$$

in order to force the MLP to stretch out the value of $d_i$ (in this work defined as the classification margin of example $i$) to its limit, creating a hidden output space where the distance between patterns of different classes is as larger as possible. Different from our approach, the traditional back-propagation methods usually adopt the error (4.1), which permits the undesirable increase of the output matrix $W_2$ in order to achieve the target output $y_i$ without taking into account the distance $d_i$, as can be inferred from (4.7).

### 4.1.3 MMGDX based on MSE

Our first approach of MMGDX, here referred as MMGDX-A, has a maximal margin objective function based on MSE:

$$J = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{4.10}$$

where $N$ is the total number of training examples and $e_i$ is defined in (4.9). The weights update is based on the gradient descent with momentum term and adaptive learning rate, therefore this method was named MMGDX. Backpropagation is used to calculate the derivatives of the objective function (4.10) as follows

$$\frac{\partial J}{\partial W_1^{Ln}} = \frac{-2W_2^n}{N\left\|W_2\right\|}\sum_{i=1}^{N} e_i \varphi^{'}\left(v_i^n\right) x_i \tag{4.11}$$

$$\frac{\partial J}{\partial b_1^n} = \frac{-2W_2^n}{N\left\|W_2\right\|}\sum_{i=1}^{N} e_i \varphi^{'}\left(v_i^n\right) \tag{4.12}$$

$$\frac{\partial J}{\partial W_2} = \frac{-2}{N}\sum_{i=1}^{N} e_i \left(\frac{yh_i}{\left\|W_2\right\|} - \frac{W_2^T yh_i W_2}{\left(\left\|W_2\right\|\right)^3}\right) \tag{4.13}$$

where $W_1^{Ln}$ is the $n^{th}$ row of matrix $W_1$, $W_2^n$ is the $n^{th}$ element of vector $W_2$, $b_1^n$ is the $n^{th}$ position of vector $b_1$, $\varphi^{'}(\cdot)$ is the derivative of the sigmoid function, $v_i^n = W_1^{Ln} x_i + b_1^n$ is the activation function of neuron $n$, and $yh_i$ is the hidden layer output vector in response to example $x_i$.

The weights of layer $(l = 1, 2)$ are updated as follows:

$$W_{1,k+1}^{Ln} = W_{1,k}^{Ln} - \alpha \left.\frac{\partial J}{\partial W_1^{Ln}}\right|_{k} - \beta \left.\frac{\partial J}{\partial W_1^{Ln}}\right|_{k-1} \tag{4.14}$$

$$W_{2,k+1} = W_{2,k} - \alpha \left.\frac{\partial J}{\partial W_2}\right|_{k} - \beta \left.\frac{\partial J}{\partial W_2}\right|_{k-1} \tag{4.15}$$

$$b_{1,k+1}^n = b_{1,k}^n - \alpha \left.\frac{\partial J}{\partial b_1^n}\right|_{k} - \beta \left.\frac{\partial J}{\partial b_1^n}\right|_{k-1} \tag{4.16}$$

where $k$ is the iteration, $\alpha$ is the learning rate, and $\beta$ is the momentum term. In short, each variable is adjusted according to the gradient descent with momentum term. For each epoch, if the MM-based objective function $J$ decreases towards the goal, then the learning rate $\alpha$ is increased by a given factor $\eta$. If $J$ increases by more than a given threshold $\gamma$, the learning rate is decreased by a given factor $\mu$ and the updating of synaptic weights and biases that increased $J$ in the current iteration are undone. During the training, the value of area under ROC curve (AUC) is calculated at each $\tau$ epochs, over the validation dataset. If AUC increases, then a register of network parameters is updated. In the final, the registered network parameters, which correspond to the highest AUC, are adopted. The training section stops after $\xi$ failed attempts in improving the AUC. Algorithm 4 details the proposed method. Notice that, at each iteration the non-recursive equations (4.14)−(4.16) are calculated, as well as the recursive equations (4.10)−(4.13) that demand a number of iterations directly proportional to the total number of training examples $N$. Therefore, the MMGDX has time complexity $O(N)$. Similarly to other first-order optimization methods, the MMGDX does not need to store second-order derivatives to compose Jacobian matrix, therefore it has space complexity $O(N)$.

### 4.1.4 MMGDX based on Lp-norm

The second approach of MMGDX, here denoted as MMGDX-B, has the objective function $J$ based on Lp-norm:

$$J = \|E\|^p \tag{4.17}$$

where $\|\cdot\|^p$ is the Lp-norm, $E = [e_1, e_2, \ldots, e_N]$ is the error vector, and $e_i$ is defined in (4.9). The main idea is to calculate the functional $J$ focusing specially on the support vector margins, inspired on the SVM soft-margin training algorithm. The Lp-norm is a trick to avoid the constrained optimization problem usual in SVM-like approaches. Notice that, larger errors $e_i$ are related to support vectors (i.e. the patterns with small distance $d$ from the separating-hyperplane), therefore, if the Lp-norm is applied, the larger is $p$ the larger is the contribution of the larger errors in the calculation of the objective function $J$. In fact, if $p \to \infty$ only the pattern with smallest distance from the separating hyperplane will be considered in the calculation of the objective function $J$. In short, the Lp-norm enables the implementation of a training algorithm with some similarity to the soft-margin SVM (i.e. with larger importance for the

**Algorithm 4** MMGDX

```
Input: ${x_{train}}$, ${y_{train}}$: training dataset
    ${x_{valid}}$, ${y_{valid}}$: validation dataset (for AUC calculation)
    $n$: number of neurons in the hidden layer
    $\tau$: number of iterations between each AUC checking
    $\xi$: stop criterion (maximum number of events $AUC \leq AUC_{max}$)
Output: $W_1, W_2, b_1, b_2$: network parameters
 1: initiate weights according to nguyen-widrow algorithm [NW90];
 2: $i \leftarrow 0; AUC_{max} \leftarrow 0; k \leftarrow 0;$
 3: while $i \leq \xi$ do
 4:   for $epoch = 1 : \tau$ do
 5:     $k \leftarrow k + 1;$
 6:     update weights by means of (4.11), (4.12), (4.13), (4.14), (4.15), and (4.16);
 7:     propagate ${x_{train}}$ through the model (4.3) obtaining ${\hat{y}_{train}}$;
 8:     apply ${\hat{y}_{train}}$ and ${y_{train}}$ in equations (4.9) and (4.10) in order to check $J_k$;
 9:     if $J_k > J_{k-1} + \gamma$ then
10:       $\alpha \leftarrow \mu \cdot \alpha;$
11:     else
12:       if $J_k < J_{k-1}$ then
13:         $\alpha \leftarrow \eta \cdot \alpha;$
14:       end if
15:     end if
16:   end for
17:   propagate ${x_{valid}}$ through the model (4.3) obtaining ${\hat{y}_{valid}}$;
18:   calculate $AUC$ using ${\hat{y}_{valid}}$ and ${y_{valid}}$;
19:   if $AUC > AUC_{max}$ then
20:     $AUC_{max} \leftarrow AUC;$
21:     $W_1^{stored} \leftarrow W_1$; $W_2^{stored} \leftarrow W_2$; $b_1^{stored} \leftarrow b_1$;
22:   else
        $i \leftarrow i + 1;$
23:   end if
24: end while
25: $W_1 \leftarrow W_1^{stored}$; $W_2 \leftarrow W_2^{stored}$; $b_1 \leftarrow b_1^{stored}$
26: adjust threshold (i.e. network parameter $b_2$) by means of ROC curve information
```

support vector margins), applying back-propagation in an unconstrained optimization approach. Backpropagation is used to calculate the derivatives of the objective function (4.17) as follows:

$$\frac{\partial J}{\partial W_1^{Ln}} = -k \frac{W_2^n}{\|W_2\|} \sum_{i=1}^{N} e_i^{(p-1)} \varphi' \left(v_i^n\right) x_i \tag{4.18}$$

$$\frac{\partial J}{\partial b_1^n} = -k \frac{W_2^n}{\|W_2\|} \sum_{i=1}^{N} e_i^{(p-1)} \varphi' \left(v_i^n\right) \tag{4.19}$$

$$\frac{\partial J}{\partial W_2} = -k \sum_{i=1}^{N} e_i^{(p-1)} \left( \frac{yh_i}{\|W_2\|} - \frac{W_2^T yh_i W_2}{\left(\|W_2\|\right)^3} \right) \tag{4.20}$$

where $W_1^{Ln}$ is the $n^{th}$ row of $W_1$, $W_2^n$ is the $n^{th}$ element of vector $W_2$, $b_1^n$ is the $n^{th}$ position of vector $b_1$, $\varphi^{'}(\cdot)$ is the derivative of the sigmoid function, $v_i^n = W_1^{Ln}x_i + b_1^n$ is the activation function of neuron $n$, $yh_i$ is the hidden layer output vector in response to example $x_i$, and

$$k = \left(\sum_{i=1}^{N} e_i^p\right)^{\left(\frac{1-p}{p}\right)} \tag{4.21}$$

The weights update was described in Subsection 4.1.3. A dynamic norm is adopted in order to escape from local minima. Actually, if the optimization algorithm stops at a local minimum, the adopted norm $p$ is replaced by L2-norm during one iteration. If $J$ is improved, the algorithm restores the adopted norm $p$.

In short, our results on benchmark datasets, [LN10] and [LN11], indicate that an adequate training algorithm can lead the MLP to achieve performance which is better than (or at least similar) as other state-of-the-art approaches, such as Bayesian Neural Network [NZ06], algorithms based on Kernel Fisher Discriminant analysis [MRW$^+$99], or SVM with nonlinear kernels [Liu04], which is the most known maximal margin algorithm, for which different kernels have been proposed.

## 4.2 Improving generalization by redundancy

Studies on animal physiology have shown evidences of neurophysiological nervous redundancy in biological systems. Redundancy seems to be a fundamental characteristic of nervous systems, as reported in [KDMC83], which has showed that there are several physiological systems within the motor nervous system that produce the same behavior. This study also has reported that the ensemble of these subsystems enhanced the movement precision. Redundancy was also found in the monkey visual cortex, see [SL86]. This physiological evidence has been complemented by some empirical studies on redundancy in artificial neural networks. The redundant artificial neural network (RNN) can be understood as an ensemble of small neural networks, here named neural subsystems, which are trained independently. After aggregating such subsystems the resultant NN model is a usual MLP with two hidden layers.

Artificial neural networks are still very simple models of the brain, which may lead us to suppose that the NN performance can be improved by adding new physiological characteristics. However, some of these characteristics may be only requirements from brain's physical processes, without any influence on the learning ability. Therefore, before increasing the complexity of the usual models by adding new biological characteristics, it is convenient to show, mathematically, their efficiency in improving the NN performance, since real world problems require large computational effort, which may prevent on-the-fly applications. Taking into account this fact, this chapter complements some founding works, such as [IP90], by presenting a theoretical analysis, in the statistical learning framework, that shows the positive effects of redundancy on the upper-bound on the expected classification risk.

RNN are in line with the current technology; notice that, due to limitations of the current paradigm in speed up the processor clock, the computer technology points to parallel processing. Therefore, RNN can take advantage of such fact, similarly to biologic systems, where redundancy does not increase the overall network processing time, because all redundant units are working in parallel. In fact, it has been estimated that the brain has up to seven layers of redundancy [Gla87].

### 4.2.1 The adopted MLP model

The present study adopts an homogeneous RNN. The term homogeneous denotes a RNN composed by subsystems with the same structure and complexity, in Vapnik sense, i.e. MLPs with the same number of neurons, but different random seeds or even trained by different training methods. The adopted MLP model is given by (4.22).

$$\begin{aligned} yh_1 &= \varphi\left(W_1 \cdot x + b_1\right) \\ yh_2 &= \phi\left(W_2 \cdot yh_1 + b_2\right) \\ \hat{y} &= W_3 \cdot yh_2 + b_3 \end{aligned} \tag{4.22}$$

where $x$ is the input vector, $yh_1$ and $yh_2$ are the output vectors of the first and second hidden layer respectively, $\hat{y}$ is the RNN output, $W_1 = \left[W_{(1,1)}^T, W_{(1,2)}^T, \ldots, W_{(1,N_c)}^T\right]^T$, $b_1 = \left[b_{(1,1)}^T, b_{(1,2)}^T, \ldots, b_{(1,N_c)}^T\right]^T$, $W_{(1,n)}$, and $b_{(1,n)}$ are the weight matrix and bias vector of the first layer of the $n^{th}$ neural subsystem,

$$W_2 = \begin{bmatrix} W_{(2,1)}^T & 0_{1\times n_n} & \cdots & 0_{1\times n_n} \\ 0_{1\times n_n} & W_{(2,2)}^T & \cdots & 0_{1\times n_n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times n_n} & 0_{1\times n_n} & \cdots & W_{(2,N_c)}^T \end{bmatrix}, \tag{4.23}$$

$b_2 = \left[b_{(2,1)}, b_{(2,2)}, \ldots, b_{(2,N_c)}\right]^T$, $W_{(2,n)}$ and $b_{(2,n)}$ are the weight matrix and bias value of the output layer of the $n^{th}$ neural subsystem, $0_{1\times n_n}$ is a vector that has all its components equal to zero, $n_n$ is the number of hidden neurons of each neural subsystem, $W_3 = [1, 1, \ldots, 1]$, $b_3 = 0$, and $\phi(\cdot)$ is a step-like function which is 1 if its argument is larger than zero, and -1 otherwise.

Figure 4.1 illustrates the architecture of the RNN, which has an usual MLP architecture, except for some disabled connections in the second hidden layer, i.e. some synaptic weights are zero, according to (4.23).

### 4.2.2 Training the RNN

The RNN is trained by parts. Namely, each neural subsystem, whose model is given by (4.3), is trained independently with initial weights and biases generated through different random seeds. The subsystems can also be trained by using different training methods and training datasets, similarly to the bagging approach; however, the training datasets must have the same cardinality, in order to conform to the theoretical analysis developed in the next section. After the training of the subsystems, their output neurons must have their linear transfer functions[2] replaced by step-like transfer functions, $\phi$, before integration in the RNN, according to Fig.4.1. The weights of the output layer are fixed, i.e. the RNN output neuron performs a non-trainable fusion. Algorithm 5 details the RNN training.

### 4.2.3 Generalization bounds for RNN

This section presents a mathematical proof of the efficiency of the proposed redundant training method. The main idea is to derive the upper bound on the expected risk

[2]see model (4.3)

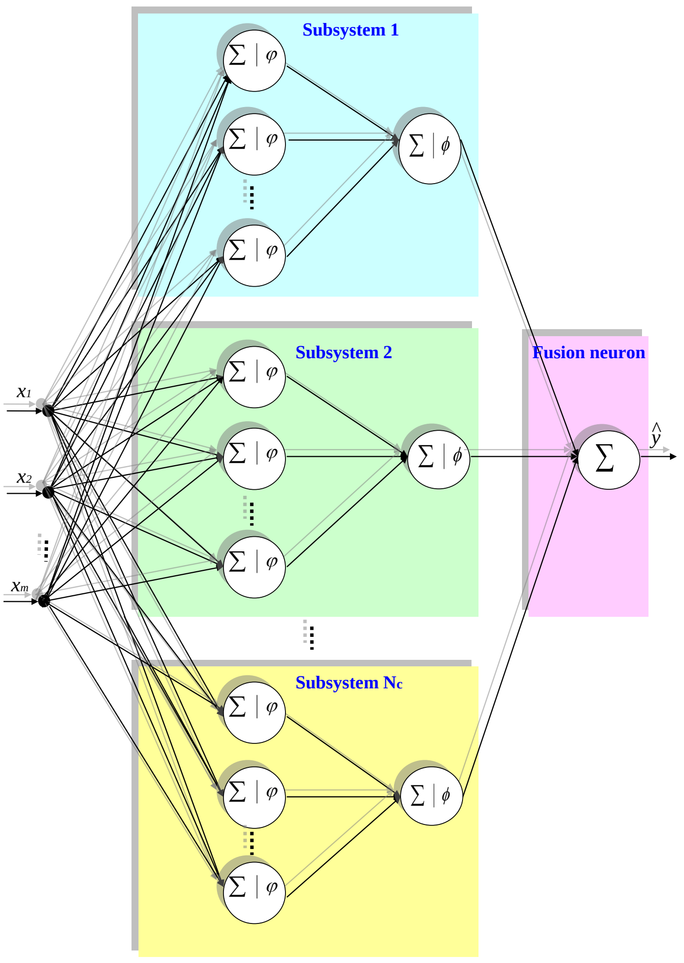


Figure 4.1: Architecture of the proposed RNN.

for RNN as function of the number of neural subsystems, in order to show that the larger the number of neural subsystems, the smaller the upper bound on the expected risk.

Considering a training dataset $S$ with $l$ pairs $(x_1, y_1), \ldots, (x_l, y_l)$, where $x \in U$ represents the input vectors and $y$ denotes the targets, which are considered to be drawn randomly and independently according to an unknown joint distribution $F(x, y) = F(y|x) F(x)$, we define the learning procedure as the process of choosing an appropriate function $f(x, \alpha^*)$ [Vap98], in the sense of the adopted objective function of the training algorithm, from a set of functions $f(x, \alpha)$, $\alpha \in \Lambda$ which can contain a finite number of elements (e.g., in the case of decision trees) or an infinite number of elements (e.g. syntactic classifiers, such as RNN, which have a set $\Lambda$ of adjustable parameters that can assume any real value).

**Algorithm 5** RNN training

```
Input: $\{x_{train}\}, \{y_{train}\}$: training dataset
    $n_n$: desired number of hidden neurons for the subsystems
    $N_c$: desired number of neural subsystems
Output: trained RNN model
 1: for $n = 1 : N_c$ do
 2:     apply $\{x_{train}\}$ and $\{y_{train}\}$ to train the $n^{th}$ neural subsystem, obtaining the adjusted parameters $W_{(1,n)}$, $W_{(2,n)}$, $b_{(1,n)}$, $b_{(2,n)}$;
 3: end for
 4: $W_1 \leftarrow \left[W^T_{(1,1)}, W^T_{(1,2)}, \dots, W^T_{(1,N_c)}\right]^T$;
 5: $b_1 \leftarrow \left[b^T_{(1,1)}, b^T_{(1,2)}, \dots, b^T_{(1,N_c)}\right]^T$;
 6: $W_2 \leftarrow \begin{bmatrix} W^T_{(2,1)} & 0_{1\times n_n} & \cdots & 0_{1\times n_n} \\ 0_{1\times n_n} & W^T_{(2,2)} & \cdots & 0_{1\times n_n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times n_n} & 0_{1\times n_n} & \cdots & W^T_{(2,N_c)} \end{bmatrix}$;
 7: $b_2 \leftarrow \left[b_{(2,1)}, b_{(2,2)}, \dots, b_{(2,N_c)}\right]^T$;
 8: $W_3 \leftarrow [1, 1, \dots, 1]$;
 9: $b_3 \leftarrow 0$;
10: apply the adjusted parameters $W_1$, $W_2$, $W_3$, $b_1$, $b_2$, and $b_3$ in (4.22) to compose the trained RNN model;
```

Taking into account the following loss function

$$L\left(f(x,\alpha), y\right) = \begin{cases} 0 \text{ if } f(x,\alpha)=y \\ 1 \text{ if } f(x,\alpha)\neq y \end{cases}, \tag{4.24}$$

the first step of our study is to determine the probability that the expected risk $R(\alpha) = \int L\left(f(x,\alpha), y\right) dF(x,y)$ will deviate from the empirical risk $R_{emp}(\alpha) = \frac{1}{l}\sum_{i=1}^{l} L\left(f(x_i,\alpha), y_i\right)$ for the case of a single neural subsystem. Since the set $\Lambda$ contains infinite real valued parameters, it is required a measure of the classifier complexity. There are several complexity measures[3], in this work it is adopted the VC-dimension [BEHW89], defined below.

**Definition 4.1.** [BEHW89] *The VC-dimension, $h$, of a set of binary functions $f(x,\alpha)$, $\alpha \in \Lambda$ is defined as the maximal number of vectors $x_1 \dots\ x_h$ which can be shattered, i.e. dichotomized in all $2^h$ ways, by using functions in the set.*

The upper-bound on the expected risk for a single neural subsystem can be estimated by applying directly the famous learning bound derived by Vapnik and Chervonenkis, which is stated in the following theorem:

[3]such as covering numbers, annealed entropy, VC entropy, Rademacher, and Gaussian complexity.

**Theorem 4.1.** [Vap98] *Let $h$ denote the VC-dimension of the set of functions $f(x,\alpha)$, $\alpha \in \Lambda$. For all $\alpha$, all $l > h$, and all $\sigma > 0$ the inequality bounding the expected risk*

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h\left(\ln \frac{2l}{h} + 1\right) - \ln \frac{\sigma}{4}}{l}} \tag{4.25}$$

*holds with probability of at least $1-\sigma$ over the random draw of the training samples.*

To complete this analysis, it is necessary to estimate the upper-bound on the expected risk for the whole RNN, given by Corollary 4.1., which requires the following lemma:

**Lemma 4.1.** *Let us assume the cumulative binomial distribution:*

$$P(X \leq x) = \sum_{k=0}^{\lfloor x \rfloor} \binom{n}{k} (p)^k (1-p)^{(n-k)} \tag{4.26}$$

*where $n$ is the number of trials, $p \in [0,1]$ is the success probability in each trial, $X$ is a random variable, and $\lfloor \cdot \rfloor$ is the floor operator. Then, for $k > np$, the Hoeffding inequality [Hoe63] yields the upper bound:*

$$P(X \leq x) \leq \frac{1}{2} \exp\left(-\frac{2}{n}(np - x)^2\right) \tag{4.27}$$

**Corollary 4.1.** *Let us assume an homogeneous RNN composed by $N_c$ neural subsystems. Suppose the neural subsystem space has VC-dimension $h_{sub}$ and let $R_{emp}^{sub}$ be the biggest value of empirical risk over all the neural subsystems. By assuming that the distribution of the error among the neural subsystems is independent, it is possible to derive the following upper bound on the expected risk for the RNN*

$$R_{RNN} \leq \frac{1}{2} \exp\left(-2N_c\left(0.5 - R_{emp}^{sub}(\alpha) - \sqrt{\frac{h_{sub}\left(\ln \frac{2l}{h_{sub}} + 1\right) - \ln \frac{\sigma}{4}}{l}}\right)^2\right) \tag{4.28}$$

*which holds with probability of at least $1-\sigma$ over the random draw of the training samples.*

**Proof**. Due to the step-like function, $\phi(\cdot)$, the output of the second hidden layer, $yh_2$, is composed by the label outputs of all the neural subsystems. Therefore, the linear output-neuron performs a majority-vote fusion over all the neural subsystems. In this case, an error occurs if, at least, $N_c/2$ neural subsystems fail, i.e. $R_{RNN} = P\left(s \leq N_c/2\right)$, where $s$ is the number of neural subsystems which have success. This probability is given by the following equation

$$R_{RNN} = P\left(s \leq N_c/2\right) \leq \sum_{k=0}^{N_c/2} \tbinom{N_c}{k} \left(1-R_{sub}\right)^k \left(R_{sub}\right)^{(N_c-k)} \tag{4.29}$$

where $R_{sub}$ is the upper bound on the expected risk for the worst neural subsystem, given by Theorem 4.1, according to the following inequation:

$$R_{sub} \leq R_{emp}^{sub}(\alpha) + \sqrt{\frac{h_{sub}\left(\ln\frac{2l}{h_{sub}}+1\right) - \ln\frac{\sigma}{4}}{l}} \tag{4.30}$$

However, (4.29) has factorials and a summation which do not enable a good analysis. Therefore, by substituting $p = 1 - R_{sub}$, $x = N_c/2$, and $n = N_c$ into (4.27) we obtain the following upper bound on (4.29):

$$R_{RNN} = P\left(s \leq N_c/2\right) \leq \frac{1}{2}\exp\left(-2N_c\left(0.5-R_{sub}\right)^2\right) \tag{4.31}$$

Substituting (4.30) into (4.31) completes the proof.■

By observing the quadratic form in the argument of the exponential function in (4.28) it is possible to state that the larger the number of neural subsystems, $N_c$, the smaller the upper bound on the expected risk, $R_{RNN}$. This analysis supports the efficiency of our redundant training method in improving the generalization capability of MLPs. However, such theoretical result is based on the assumption that the distribution of the error among the neural subsystems is independent. Therefore, *the diversity of behaviour among the neural subsystems is a key subject, which can be approached by adopting different training methods for each neural subsystem*, as will be done in the experimental chapter of this thesis.

## 4.3 Improving generalization by regularization

This section deals with regularization, a commonly used technique for preventing the learning algorithm from overfitting the training data. There are three usual regularization techniques for NN: early stopping [TG99], curvature-driven smoothing [Bis96], and weight decay [Jin04]. In the early stopping criterion the available data are divided into three subsets. The first subset is the training dataset, which is used for updating the network weights and biases. The second subset is used as a validation dataset and the third subset is used to evaluate the final accuracy. The error on the validation dataset is monitored during the training process. After some number of iterations the NN begins to overfit the data and the error on the validation dataset begins to rise. When the validation error increases during a specified number of iterations, the algorithm stops the training section and adopts the weights and biases corresponding to the minimum of the validation error. Curvature-driven smoothing includes smoothness requirements on the cost function of learning algorithms, which depend on the derivatives of the network mapping. Weight decay is implemented by including additional terms in the cost function of learning algorithms, which penalize overly high values of weights and biases, in order to control the classifier complexity, which forces the NN response to be smoother and less likely to overfit.

This thesis introduces a novel regularization scheme, named eigenvalue decay, that includes an additional term in the cost function of the learning algorithm, which penalize overly high values of the biggest and the smallest eigenvalues of $W_1W_1^T$, where $W_1$ is the synaptic weight matrix of the first layer of model (4.3). This approach aims at improving the classification margin, as will be shown. This regularization scheme led to the development of a new training method based on the same principles of SVM, and so named Support Vectors NN (SVNN). Moreover, it is given an insight on how the most usual cost function, i.e. mean squared error (MSE), can hinder the margin improvement. This section starts by proposing the eigenvalue decay technique, then the relationship between such regularization scheme and the classification margin is analyzed. Finally, a novel algorithm for maximum margin training, based on regularization and evolutionary computing, is proposed.

### 4.3.1 Eigenvalue decay

The most usual objective function is the MSE:

$$E = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.32}$$

where $N$ is the cardinality of the training dataset, $y_i$ is the target output, $\hat{y}_i$ is the output estimated by the MLP for the input $x_i$ belonging to the training dataset. However, in case of the usual weight decay method [DB98], additional terms which penalize overly high values of weights and biases are included. Therefore, the generic form for the objective function is:

$$E^* = E + \kappa_1 \sum_{w_i \in W_1} w_i^2 + \kappa_2 \sum_{w_j \in W_2} w_j^2 + \kappa_3 \sum_{b_{(1,k)} \in b_1} b_{(1,k)}^2 + \kappa_4 b_2^2 \tag{4.33}$$

where $W_1$, $W_2$, $b_1$, and $b_2$ are the MLP parameters, according to (4.3), and $\kappa_i > 0$, i=(1...4) are regularization hyperparameters. Such method was theoretically analyzed by Bartlett [Bar98], which concludes that the bounds on the expected risk of MLPs depends on the magnitude of the parameters rather than the number of parameters. In the same work [Bar98] it is shown that the misclassification probability can be bounded in terms of the empirical risk, the number of training examples, and a scale-sensitive version of the VC-dimension, known as the fat-shattering dimension[4]. Then, upper bounds on the fat-shattering dimension for MLPs are derived in terms of the magnitudes of the network parameters, independently from the number of parameters[5].

In the case of eigenvalue decay the proposed objective function is:

$$E^{**} = E + \kappa \left(\lambda_{min} + \lambda_{max}\right) \tag{4.34}$$

where $\lambda_{min}$ and $\lambda_{max}$ are, respectively, the smallest and the biggest eigenvalues of $W_1 W_1^T$.

[4]See Theorem 2 of [Bar98]
[5]See Theorem 13 of [Bar98]

### 4.3.2 Analysis on eigenvalue decay

In this sub-section we show a relationship between eigenvalue decay and the classfication margin, $m_i$. Our analysis requires the following lemma:

**Lemma 4.2.** *Let $K$ denotes the field of real numbers, $K^{n\times n}$ a vector space containing all matrices with $n$ rows and $n$ columns with entries in $K$, $A \in K^{n\times n}$ be a symmetric positive-semidefinite matrix, $\lambda_{min}$ be the smallest eigenvalue of $A$, and $\lambda_{max}$ be the largest eigenvalue of $A$. Therefore, for any $x \in K^n$, the following inequalities hold true:*

$$\lambda_{min} x^T x \leq x^T A x \leq \lambda_{max} x^T x \tag{4.35}$$

**Proof**. The upper bound on $x^T Ax$, i.e. the second inequality of (4.35), is well known; however, this work also requires the lower bound on $x^T Ax$, i.e. the first inequality of (4.35). Therefore, since this proof is quite compact, we will save a small space in this work to present the derivation of both bounds as follows:

Let $V = [v_1 \dots v_n]$ be the square $n \times n$ matrix whose $i^{th}$ column is the eigenvector $v_i$ of $A$, and $\Lambda$ be the diagonal matrix whose $i^{th}$ diagonal element is the eigenvalue $\lambda_i$ of $A$; therefore, the following relations hold:

$$x^T Ax = x^T V V^{-1} A V V^{-1} x = x^T V \Lambda V^{-1} x = x^T V \Lambda V^T x \tag{4.36}$$

Taking into account that $A$ is positive-semidefinite, i.e. $\forall i,\ \lambda_i \geq 0$:

$$x^T V \left(\lambda_{min} I\right) V^T x \leq x^T V \Lambda V^T x \leq x^T V \left(\lambda_{max} I\right) V^T x \tag{4.37}$$

where $I$ is the eye matrix. Note that $x^T V \left(\lambda_{min} I\right) V^T x = \lambda_{min} x^T x$ and $x^T V \left(\lambda_{max} I\right) V^T x = \lambda_{max} x^T x$; therefore, substituting (4.36) into (4.37) yields (4.35). ■

The following theorem gives a lower and an upper bound on the classification margin:

**Theorem 4.2.** *Let $m_i$ be the margin of the training example $i$, i.e. the smallest orthogonal distance between the classifier separating hypersurface and the training example $i$, $\lambda_{max}$ be the biggest eigenvalue of $W_1 W_1^T$, and $\lambda_{min}$ be the smallest eigenvalue of $W_1 W_1^T$; then, for $m_i > 0$, i.e. an example correctly classified, the following inequalities hold true:*

$$\frac{1}{\sqrt{\lambda_{max}}}\mu \leq m_i \leq \frac{1}{\sqrt{\lambda_{min}}}\mu \tag{4.38}$$

where

$$\mu = \min_j \left( y_i \frac{W_2^T \Gamma_j W_1 \left(x_i - x_{proj}^j\right)}{\sqrt{W_2^T \Gamma_j \Gamma_j^T W_2}} \right), \tag{4.39}$$

$\Gamma_j = \begin{bmatrix} \varphi'(v_1) & 0 & \cdots & 0 \\ 0 & \varphi'(v_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varphi'(v_n) \end{bmatrix}$, $[v_1, v_2, \ldots, v_n]^T = W_1 \cdot x_k + b_1$, $\varphi'(v_n) = \left.\frac{\partial \varphi}{\partial v_n}\right|_{x_{proj}^j}$, $x_{proj}^j$ is the $j^{th}$ projection of the $i^{th}$ training example, $x_i$, on the separating hypersurface, as illustrated in Fig.4.2, and $y_i$ is the target label of $x_i$.

**Proof**. The first step in this proof is the calculation of the gradient of the NN output $\hat{y}$ in relation to the input vector $x$ at the projected point, $x_{proj}^j$. From (4.3) we have:

$$\nabla \hat{y}_{(i,j)} = \left.\frac{\partial \hat{y}}{\partial x}\right|_{x_{proj}^j} = W_2^T \Gamma_j W_1 \tag{4.40}$$

The versor

$$\vec{p_j} = \frac{\nabla \hat{y}_{(i,j)}}{\left\|\nabla \hat{y}_{(i,j)}\right\|} \tag{4.41}$$

is normal to the separating surface, giving the direction from $x_i$ to $x_{proj}^j$; therefore

$$x_i - x_{proj}^j = d_{(i,j)} \vec{p_j} \tag{4.42}$$

where $d_{(i,j)}$ is the scalar distance between $x_i$ and $x_{proj}^j$. From (4.42) we have:

$$\nabla \hat{y}_{(i,j)} \left(x_i - x_{proj}^j\right) = d_{(i,j)} \nabla \hat{y}_{(i,j)} \vec{p_j} \tag{4.43}$$

Substituting 4.41 into 4.43 and solving for $d_{(i,j)}$, yields:

$$d_{(i,j)} = \frac{\nabla \hat{y}_{(i,j)} \left(x_i - x_{proj}^j\right)}{\left\|\nabla \hat{y}_{(i,j)}\right\|} \tag{4.44}$$

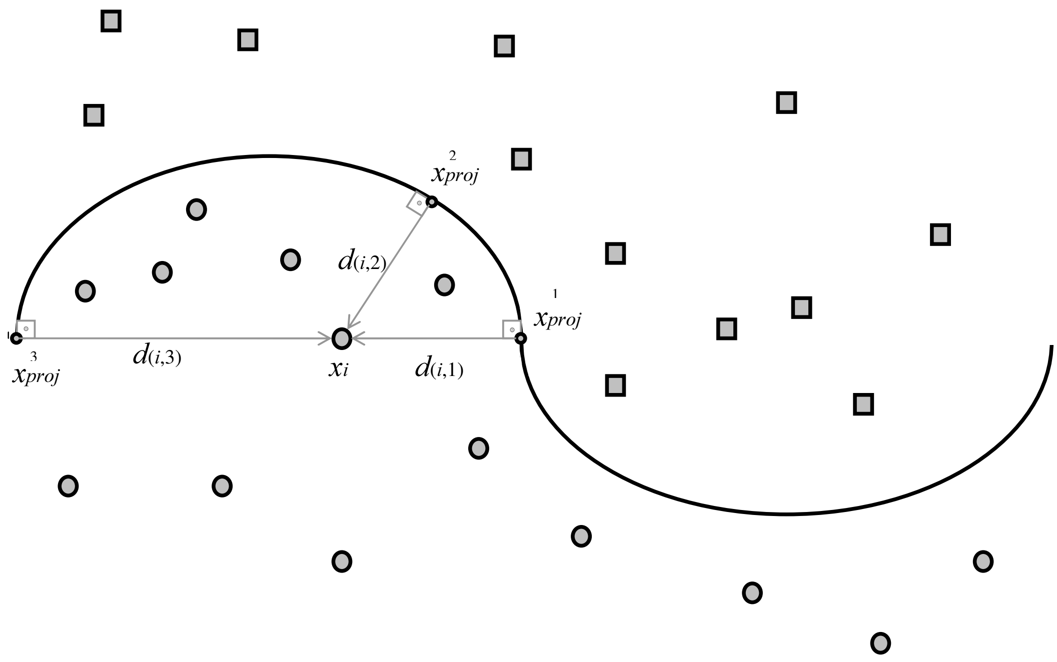


Figure 4.2: A feature space representing a nonlinear separating surface with the projections, $x_{proj}^{j}$, of the $i^{th}$ training example, $x_i$, and examples of orthogonal distances $d_{(i,j)}$.

The absolute value of the classifier margin, $m_i$, is the smallest value of $d_{(i,j)}$ in relation to $j$, that is:

$$|m_i| = \min_j \left( d_{(i,j)} \right) \tag{4.45}$$

The sign of $m_i$ depends on the target class $y_i$, therefore:

$$m_i = \min_j \left( y_i \frac{\nabla \hat{y}_{(i,j)} \left( x_i - x_{proj}^{j} \right)}{\left\| \nabla \hat{y}_{(i,j)} \right\|} \right) \tag{4.46}$$

Substituting (4.40) in (4.46), yields:

$$m_i = \min_j \left( y_i \frac{W_2^T \Gamma_j W_1 \left( x_i - x_{proj}^{j} \right)}{\sqrt{W_2^T \Gamma_j W_1 W_1^T \Gamma_j^T W_2}} \right). \tag{4.47}$$

Note that $W_1 W_1^T$ is a symmetric positive-semidefinite matrix, therefore, from Lemma 4.2., the inequalities:

$$\lambda_{min} W_2^T \Gamma_j \Gamma_j^T W_2 \leq W_2^T \Gamma_j W_1 W_1^T \Gamma_j^T W_2 \leq \lambda_{max} W_2^T \Gamma_j \Gamma_j^T W_2 \tag{4.48}$$

hold true for any $\Gamma_j$ and any $W_2$. From (4.48) and (4.47) it is easy to derive (4.38). ■

The objective function of the eigenvalue decay decreases $\lambda_{max}$ and $\lambda_{min}$, aiming at increasing the lower bound and the upper bound on the classification margin, according to (4.38). Notice that, the numerator of (4.39) depends not only on the magnitudes $\left\|W_2^T \Gamma_j W_1\right\|$ and $\left\|x_i - x_{proj}^j\right\|$ but also on the collinearity between $W_2^T \Gamma_j W_1$ and $\left(x_i - x_{proj}^j\right)$, which can be improved even taking into account the minimization of $\lambda_{max}$ and $\lambda_{min}$.

### 4.3.3 Maximal-margin Training by GA

Some previous works have proposed maximal-margin training algorithms for NN inspired on SVM, such as [DGC07] and [Abe05]. In [DGC07] a decision tree based on linear programming is applied to maximize the margins, while in [Abe05] an MLP is trained layer-by-layer based on the CARVE algorithm [YD98]. However, Theorem 4.2 allows us to propose a maximal-margin training method even more similar to SVM, in the sense that the proposed method also minimizes values related with the parameters of the classifier model, in order to maximize the margin, allowing the minimization of the classifier complexity without compromising the accuracy.

The main idea of our method is not only to avoid nonlinear SVM kernels, in such a way as to offer a faster nonlinear classifier, but also to be based on the maximal-margin principle; moreover, the proposed method is more suitable for on-the-fly applications, such as object detection [EG09], [PLN09b]. Namely, the use of SVM with nonlinear kernels may implicate in a prohibitive computational cost. Notice that the SVM decision function, $c(x)$, requires a large amount of time when the number of support vectors, $N_{sv}$, is high:

$$c(x) = \mathtt{sgn}\left(\sum_{i=1}^{N_{sv}} y_i \alpha_i K\left(x_i, x\right) + b\right) \tag{4.49}$$

where $\alpha_i$ and $b$ are SVM parameters, $(x_i, y_i)$ is the $i^{th}$ support vector data pair, $\mathrm{sgn}(\cdot)$ is 1 if the argument is greater than zero and $-1$ if it is less than zero, and $K(\cdot,\cdot)$ is a nonlinear kernel function, i.e. the algorithm fits the maximum-margin hyperplane in a transformed feature space, in order to enable a nonlinear classification.

The iterative computation (4.49) becomes more simple if a linear kernel, $(x_i^T x_j)$, is used:

$$c(x) = \texttt{sgn}\left(\sum_{i=1}^{N_{sv}} y_i \alpha_i x_i^T x + b\right) = \texttt{sgn}\left(a \cdot x + b\right) \tag{4.50}$$

where $a = \sum_{i=1}^{N_{sv}} y_i \alpha_i x_i^T$ is calculated once. The right-hand side of (4.50) does not involve an iterative computation and thus decreases the processing time by, at least, $N_{sv}$ times. However, this classifier cannot perform a nonlinear classification. Due to this fact, our research group has been working on maximal-margin training algorithms for NN, such as the MMGDX [LN10]. However, the optimization problem proposed in [LN10] is usually hard, because the objective function of MMGDX has several local minima, as well as points where the gradient vector has large magnitude. These facts motivated this new SVM-like training method for NN, named Support Vector NN (SVNN), here proposed.

In order to better understand our method, it is convenient to take into account the soft margin SVM optimization problem, as follows:

$$\min_{w,\xi_i}\left(\frac{1}{2}\left\|w\right\|^2 + C\sum_{i=1}^{N}\xi_i\right) \tag{4.51}$$

subject to

$$\forall i\,|y_i\left(w \cdot x_i - b\right) \geq 1 - \xi_i \tag{4.52}$$

$$\forall i\,|\xi_i \geq 0 \tag{4.53}$$

where $w$ and $b$ compose the separating hyperplane, $C$ is a constant, $y_i$ is the target class of the $i^{th}$ training example, and $\xi_i$ are slack variables, which measure the degree of misclassification of the vector $x_i$. The optimization is a tradeoff between a large margin $(\min \left\|w\right\|^2)$, and a small error penalty $(\min C\sum_{i=1}^{N}\xi_i)$.

We propose to train the NN by solving the following similar optimization problem:

$$\min_{W_1,\xi_i}\left(\lambda_{min} + \lambda_{max} + \frac{C}{N}\sum_{i=1}^{N}\xi_i\right) \tag{4.54}$$

subject to

$$\forall i\,|y_i\hat{y}_i \geq 1 - \xi_i \tag{4.55}$$

$$\forall i\,|\xi_i \geq 0 \tag{4.56}$$

where $\hat{y}$ is given by (4.3), $C$ is a regularization hiperparameter, $y_i$ is the target class of the $i^{th}$ training example, and $\xi_i$ are also slack variables, which measure the degree of misclassification of the vector $x_i$.

The constrained optimization problem (4.54)-(4.56) is replaced by the following equivalent unconstrained optimization problem:

$$\min_{W_1,W_2,b_1,b_2} \Phi \tag{4.57}$$

where

$$\Phi = \lambda_{min} + \lambda_{max} + \frac{C}{N}\sum_{i=1}^{N} H(y_i\hat{y}_i) \tag{4.58}$$

and $H(t) = \max(0, 1-t)$ is the Hinge loss.

Since (4.57) has the discontinuous objective function $\Phi$, which does not allow the use of gradient-based optimization methods, a real-coded GA is applied, using $\Phi$ as fitness function. Note that the last term of (4.58) penalizes models whose estimated outputs do not fit the constraint $y_i\hat{y}_i \geq 1$, in such a way as to save a *minimal margin*, while the minimization of the first two terms of (4.58) aims at the enlargement of such *minimal margin* by eigenvalue decay, acording to Theorem 4.2. Algorithm 6 details the proposed optimization process.

Fig.4.3 illustrates the two-dimensional feature space of Banana benchmark dataset and the separating surface generated by a NN trained by Algorithm 6. The examples in the yellow area are classified as positive, while the examples in the white area are classified as negative.

Training algorithms based on MSE often create unnecessary boundaries, as an effort to attribute label 1 for all the positive examples and -1 for all the negative examples, as can be seen in Fig.4.4, which illustrates the feature space of Banana benchmark dataset, as well as the separating surface generated by a NN trained by GDX.

SVNN does not apply the MSE cost function, which *penalizes* output values, $\hat{y}_i$, bigger than 1 or smaller than -1, during the training. Fig.4.5 illustrates the behaviour of a NN model trained by MMGDX [LN10] on Banana dataset. Note that MMGDX avoids unnecessary boundaries that increase the risk of miss-classification in case of large dispersion around the data clusters

In short, training algorithms based on MSE hinders the margin improvement, because this objective function bounds the value of $|\hat{y}_i|$.

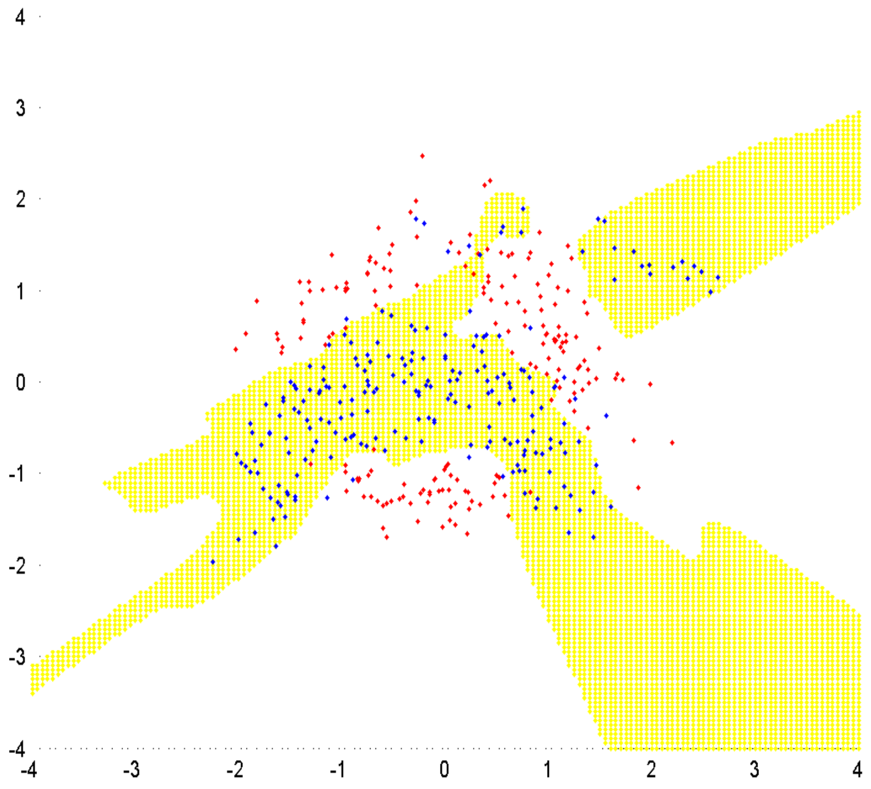


Figure 4.3: Feature space of Banana benchmark dataset, as well as the separating surface generated by a NN trained by SVNN. Blue points are positive examples, while red points are negative.

## 4.4 Improving generalization by transduction

This section deals with transduction, a concept closely related to semi-supervised learning [CSZ06]. However, differently from inductive inference, no general decision rule is inferred. In the case of transduction the inferred decision rule aims only at labelling the testing data.

The SVM-like training method, introduced in the previous section, can be exploited to address transductive learning. Therefore, we propose the transductive NN (TNN), which is similar to the transductive SVM (TSVM) [Vap98]. The proposed TNN accomplishes transduction by finding those test labels for which, after training a

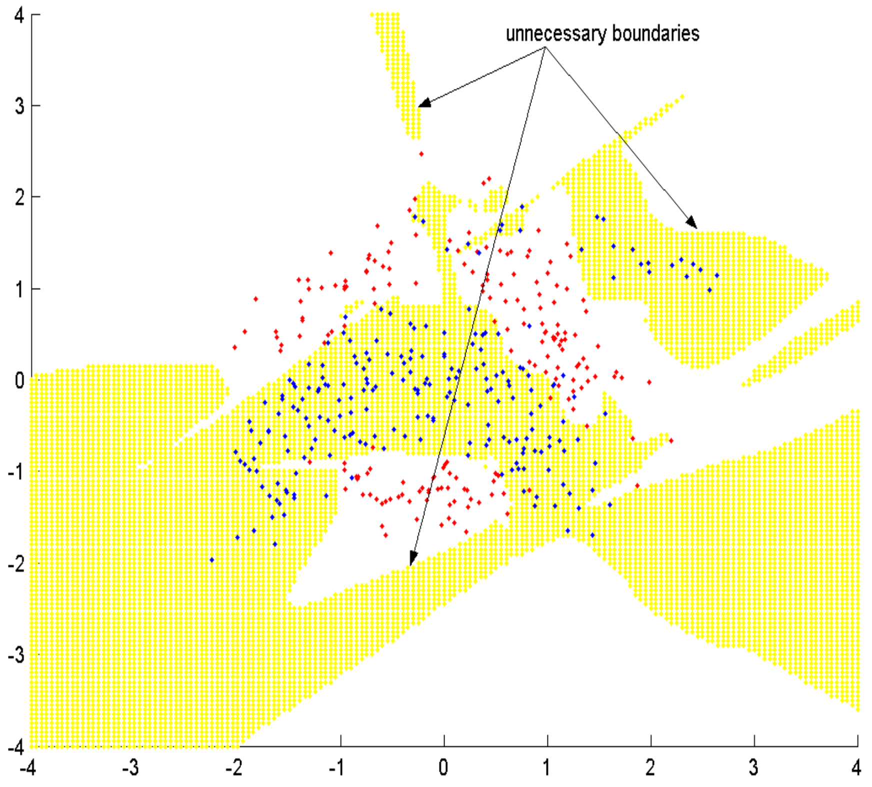


Figure 4.4: Feature space of Banana benchmark dataset, as well as the separating surface generated by a NN trained by GDX. Blue points are positive examples, while red points are negative.

NN on the combined training and test datasets, the margin on the both datasets is maximal. Therefore, similarly to TSVM, TNN exploits the geometric structure in the feature vectors of the test examples, by taking into account the principle of low density separation, i.e. the decision boundary should lie in a low-density region of the feature space, because a decision boundary in a high-density region would cut a data cluster into two different classes, which is in disagreement with the cluster assumption that is stated as follows: if points are in the same data cluster, they are likely to be of the same class.

The TNN training method can be easily implemented by including in (4.57) an additional term that penalizes all the unlabeled data which are near to the decision

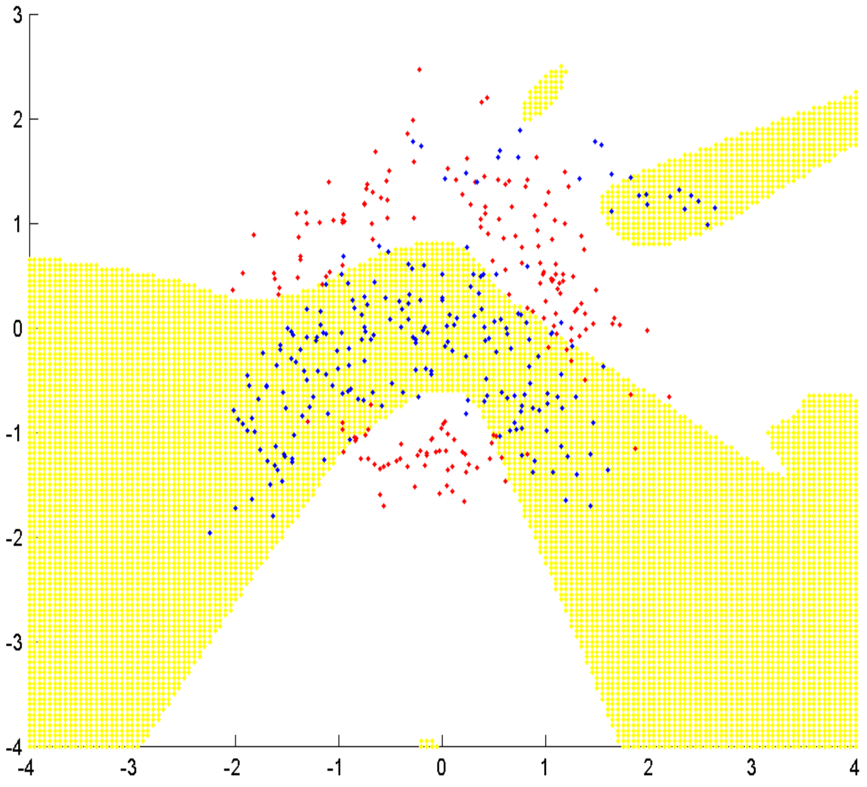


Figure 4.5: Feature space of Banana benchmark dataset, as well as the separating surface generated by a NN trained by MMGDX. Blue points are positive examples, while red points are negative.

boundary. Therefore, the new optimization problem is:

$$\min_{W_1,W_2,b_1,b_2} \Phi^* \tag{4.59}$$

where

$$\Phi^* = \lambda_{min} + \lambda_{max} + \frac{C_1}{N}\sum_{i=1}^{N} H(y_i\hat{y}_i) + \frac{C_2}{N_u}\sum_{j=1}^{N_u} H(|\hat{y}_j|), \tag{4.60}$$

$C_1$ and $C_2$ are constants, $\hat{y}_j$ is the NN output for the unlabeled data $x_j$, and $N_u$ is the cardinality of the unlabeled dataset. Notice that, the operator $|\cdot|$ in the last term of (4.60) makes this additional term independent of the class assigned by the NN for the unlabeled example, i.e. independent from the sign of $\hat{y}_j$, since we are interested only in the distance from the unlabeled data to the decision boundary. In order to illustrate the effect of the last term of (4.60), i.e. the term that penalizes the unlabeled data which are near to the decision boundary, we introduce two toy examples which enable a comparative study on the decision boundaries generated by SVNN and TNN, as can be seen in Figs. 4.6 and 4.7, where circles represent training data and points represent testing (unlabeled) data. The SVNN regularization parameter was set as $C_1 = 10^4$, while the TNN parameters were set as $C_1 = C_2 = 10^4$.

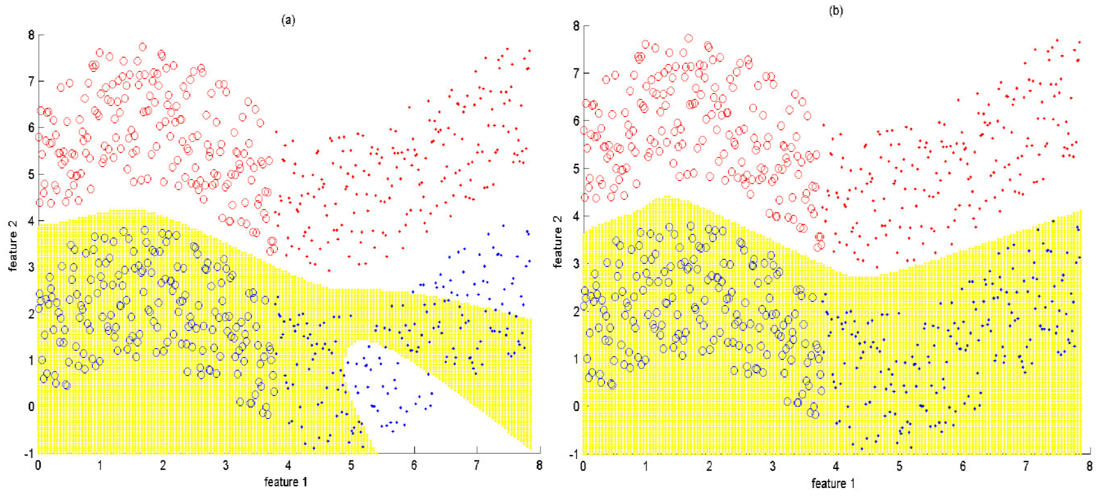


Figure 4.6: Separating surfaces generated by two NNs with 5 hidden neurons. Circles represent training data and points represent testing (unlabeled) data: (a) NN trained by SVNN, (b) NN trained by TNN.

Note that both toy examples are in accordance with the cluster assumption, i.e. there are low-density regions surrounding data clusters whose elements belong to the same class. TNN places the separating-surface along such low-density regions, in order to increase the absolute value of the margin of the unlabeled data, in such a way as to decrease the last term of (4.60). Empirically, it is sometimes observed that the solution to (4.60) is unbalanced, since it is possible to decrease the last term of (4.60) by placing the separating-surface away from all the testing instances, as can be seen in Fig. 4.8. In this case, all the testing instances are predicted in only one of the classes. Such problem can also be observed in case of TSVM, for which an heuristic solution is applied to constrain the predicted class proportion on the testing data, so that it is the same as the class proportion on the training data. This work addopts a solution similar to the heuristic usually applied to TSVM, by including in (4.60) a

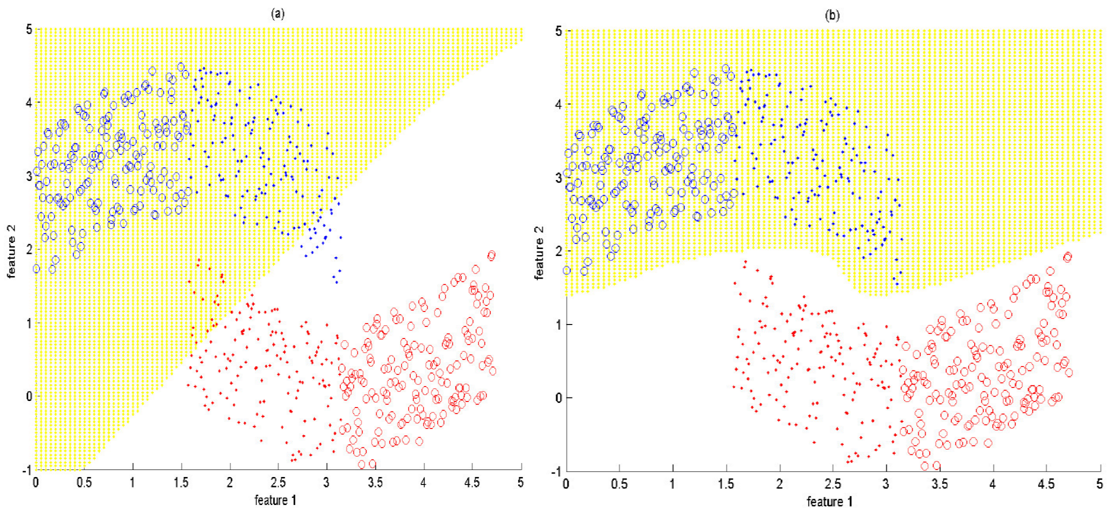


Figure 4.7: Separating surfaces generated by two NNs with 4 hidden neurons. Circles represent training data and points represent testing (unlabeled) data: (a) NN trained by SVNN, (b) NN trained by TNN.

term that penalizes models whose predicted class proportion on the testing data is different from the class proportion on the training data. Therefore, we rewrite (4.60) as:

$$\begin{aligned}\Phi^* = \lambda_{min} + \lambda_{max} + \tfrac{C_1}{N}\textstyle\sum_{i=1}^{N} H(y_i\hat{y}_i) + \tfrac{C_2}{N_u}\sum_{j=1}^{N_u} H(|\hat{y}_j|) \\ +C_3\left|\tfrac{1}{N}\textstyle\sum_{i=1}^{N} y_i - \tfrac{1}{Nu}\sum_{j=1}^{N_u}\mathtt{sgn}\,(\hat{y}_j)\right|\end{aligned} \tag{4.61}$$

where $C_3$ is a penalization coeficient. By using (4.61) with $C_1 = C_2 = 10^4$ and $C_3 = 10^3$ we obtain the separating-surface illustrated by Fig. 4.9.

## 4.5 Multi-class problems

As occurs in case of SVM, the extension of the proposed training methods from two-class to multi-class is not trivial and may be a topic for further study. However, it is possible to decompose the multi-class classification problem into multiple two-class classification problems. Some usual approaches to decompose a multi-class pattern classification problem into two-class problems are one-against-all (OAA), one-against-one (OAO), and P-against-Q (PAQ). Those approaches are popular among researchers in SVM, Adaboost, or decision trees.

The OAA modeling scheme was first introduced by Vapnik [Vap98] in the SVM

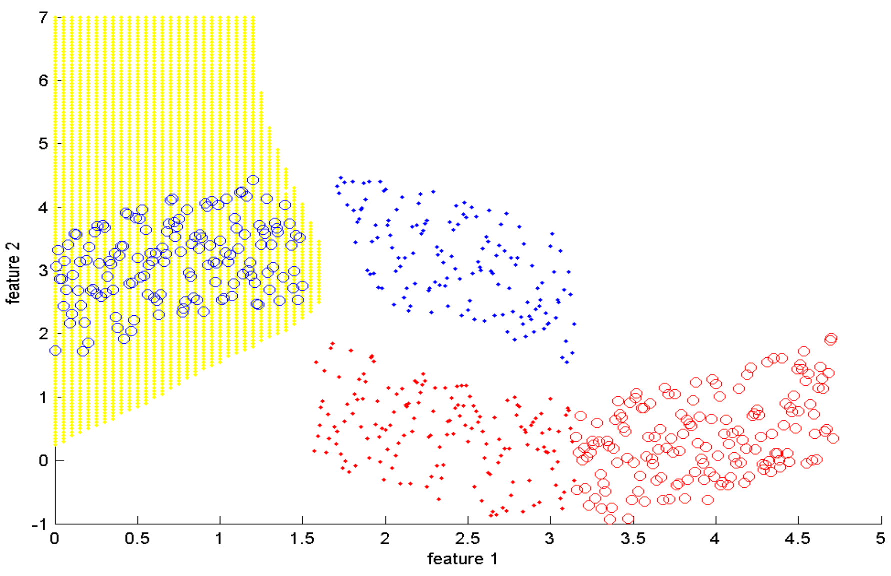


Figure 4.8: Toy experiment with TNN without the last term of (4.61). Circles represent training data and points represent testing (unlabeled) data.

context. For a $M$-class pattern classification problem the OAA scheme uses a system of $M$ binary NNs. In order to train the $m^{th}$ NN, the traning dataset $\Omega$ is decomposed in two sets, $\Omega = \Omega_m \cup \Omega_{\bar{m}}$, where $\Omega_m$ contains all the examples of class $m$, which receive the label 1, and $\Omega_{\bar{m}}$ contains all the examples belonging to all other classes, which receive the label -1. A decision function for the ensemble output can be

$$\hat{c} = \arg \max_{m=1,\ldots,M} (\hat{y}_m) \tag{4.62}$$

where $\hat{c}$ is the estimated class and $\hat{y}_m$ is the likelihood of the $m^{th}$ NN. This architecture has advantages over a single NN for multi-class problems, for instance, each NN can have its own feature space and architecture, since all of them are trained independently. However, there are some disadvantages, namely, the ensemble may not adequately cover some regions in the feature space, i.e., regions that are rejected by all NNs as other classes. Another problem is the training data, namely, when the number of classes is large, the training data for each NN is highly unbalanced, i.e., $\Omega_m << \Omega_{\bar{m}}$. This fact can lead the NN to totally ignoring the minority class $\Omega_m$.

The OAO modeling scheme, also known as pair-wise method, can avoid the OAA

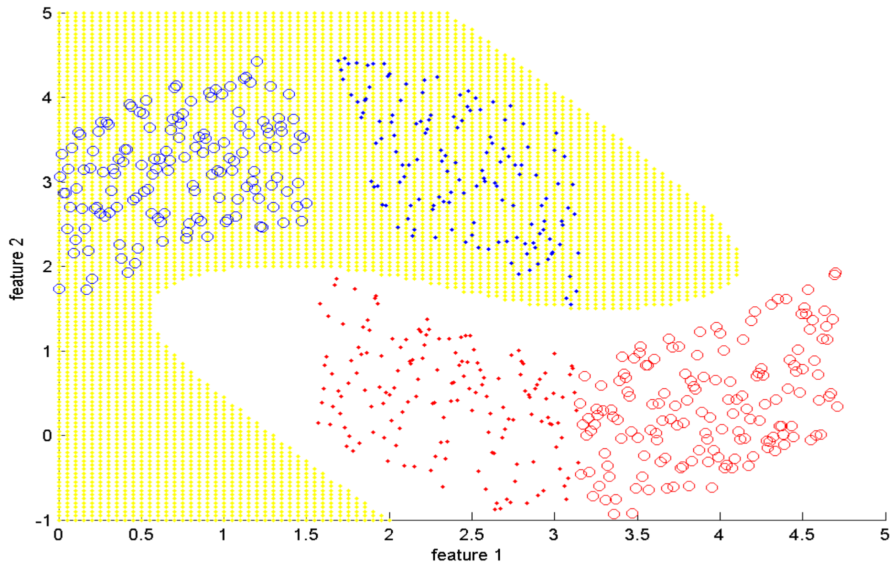


Figure 4.9: Toy experiment using TNN with (4.61). Circles represent training data and points represent testing (unlabeled) data.

problems. The pair-wise method decomposes the $M$-class pattern classification problem into $M(M-1)/2$ binary problems, i.e., each NN is trained to discriminate class $i$ from class $j$, avoiding highly unbalanced training datasets. Notice that, each class $m$ can receive up to $M-1$ votes, because there are $M-1$ NNs that are trained to discriminate class $m$ from each one of the other classes. Among many decision functions, we detach the simple majority vote, which counts the votes for each class based on the output from all NNs. The class with the largerst number of votes is the system output. In OAO modeling the feature space is less likely to have uncovered regions, due to the redundancy in the training of pattern classes. Another advantage of OAO modeling is that it has the capability of incremental class learning, i.e., a new set of NNs can be included and trained in order to represent a new class without affecting the existing NNs. However, the major disadvantage of OAO modeling is the required computational effort when the number of classes, $M$, is large. Notice that, in this method it is necessary to train $M(M-1)/2$ NNs, therefore, both time and space complexities grow in the order $O\left(M^2\right)$. Fortunately, it is possible to train all NNs simultaneously on different computers in order to speed up the training time. More details on multi-class pattern classification using NNs can be found in [OM07].

**Algorithm 6** Maximal Margin Training by GA

```
Input: $X$,$y$: matrices with $N$ training datapairs;
    $n_{neu}$: number of hidden neurons;
    $C$ : regularization hyperparameter;
    $a$: selective pressure;
    $max_{gener}$: maximum number of generations;
    $N_{pop}$: population size
Output: $W_1$, $W_2$, $b_1$, and $b_2$: NN parameters
 1: generate a set of $N_{pop}$ chromosomes, $\{Cr\}$, for the initial population, taking into account
    the number of input elements and $n_{neu}$; therefore, each chromosome is a vector $Cr =$
    $[w_1, \ldots, w_{n_w}, b_1, \ldots, b_{n_b}]$ containing all the $N_g$ synaptic weights and biases randomly generated
    according to the Nguyen-Widrow criterion [NW90];
 2: for $generation = 1 : max_{gener}$ do
 3:   for $ind = 1 : N_{pop}$ do
 4:     rearrange the genes, $Cr^{ind}$, of individual $ind$, in order to compose the NN parameters $W_1$,
        $W_2$, $b_1$, and $b_2$.
 5:     for $i = 1 : N$ do
 6:       calculate $\hat{y}_i$, according to (4.3), using the weights and biases of individual $ind$;
 7:     end for
 8:     calculate $\Phi$ for the individual $ind$, according to (4.57), using $y$ and the set of NN outputs
        $\{\hat{y}_i\}$ previously calculated;
 9:     $\Phi_{ind} \leftarrow \Phi$: storing the fitness of individual $ind$;
10:   end for
11:   rank the individuals according to their fitness $\Phi_{ind}$;
12:   store the genes of the best individual in $Cr^{best}$;
13:   $k \leftarrow 0$;
14:   for $ind = 1 : N_{pop}$ do
15:     $k \leftarrow k + 1$;
16:     $\vartheta_j \leftarrow$ random number $\in [0, 1]$ with uniform distribution, $j = (1, 2)$;
17:     $parent_j \leftarrow round\left(N_{pop}\frac{e^{a\vartheta_j}-1}{e^a-1}\right)$, $j = (1, 2)$: (randomly selecting the indexes of parents by
        using the asymmetric distribution proposed in [LNA+09]);
18:     for $n = 1 : N_g$ do
19:       $\eta \leftarrow$ random number $\in [0, 1]$ with uniform distribution;
20:       $Cr^{son}_{(k,n)} \leftarrow \eta Cr_{(parent_1,n)} + (1-\eta)\, Cr_{(parent_2,n)}$: calculating the $n^{th}$ gene, to compose the
          chromosome of the $k^{th}$ individual of the new generation, by means of weighted average;
21:     end for
22:   end for
23: end for
24: rearrange the genes of the best individual, $Cr^{best}$, in order to compose the NN parameters $W_1$,
    $W_2$, $b_1$, and $b_2$.
```

# Chapter 5

# Improving the generalization capacity of cascade classifiers by SRM

CASCADE classifier is a suitable approach in handling highly unbalanced data, since it successively rejects negative occurences in a cascade structure, keeping the processing time suitable for on-the-fly applications. Therefore, such kind of classifier ensemble is especially important for machine vision applications, such as vision based object detection, [VJ01], [MG06], for which the most usual approach is to scan the image frame by using a sliding window, which generates thousands of negative occurrences for each positive cropped image. On the other hand, similarly to other classifier ensembles, cascade classifiers are likely to have high VC dimension, which may lead to over-fitting the training data. Therefore, in this chapter the SRM principle is exploited in order to improve the generalization capacity of cascade classifiers by controlling their complexity, which depends on the model of their classifiers, the number of cascade stages, and the feature space dimension of each stage. In this context, we propose to accomplish the SRM principle by controling the number of cascade stages and the number of features in each stage.

The training method proposed in this chapter, named SRM-cascade [LPNA11], controls the number of features in each cascade stage by applying a feature selector, in such a way to find the number of features that minimizes the upper bound expected risk (4.25) of each cascade stage independently. However, the control of the number of cascade stages is not so simple, since it depends on the estimation of the strutural risk of the entire ensemble. Unfortunately, despite the methodological [GB00], [VJ01] and experimental [SK07], [MG06] contributions given in previous works, there is still a lack of theoretical analysis on the generalization capability of cascade classifiers; differently from the bagging strategy, which was theoretically analyzed by previous works such as [SFBL98] or [KWD03]. Therefore, it was required a theoretical analysis on cascade classifiers, based on statistical learning theory, in order to provide an upper-bound on the expected classification risk for the entire cascade ensemble.

Taking into account that a positive occurrence rejected by a cascade stage cannot be recovered by the following stages, the classifiers that compose the ensemble must be adjusted so that the TP is close to one. The usual approach to achieve high TP is to adjust the classifier threshold, i.e. the bias, after the training. However, this approach can be improved by adjusting all the classifier parameters (not only the bias), in such a way as to achieve the intended TP. Therefore, in this chapter it is proposed a new training method for linear classifiers that enables the control of the ratio between TP and FP during the training.

## 5.1 Brief description of cascade classifiers

This section briefly introduces the generic model of a cascade classifier ensemble, in order to contextualize our analysis. The cascade classifier can be seen as a degenerate decision tree, i.e., a positive result from the first classifier triggers the evaluation of a second classifier. A positive result from the second classifier triggers a third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the pattern, according to the scheme depicted in Fig.5.1.

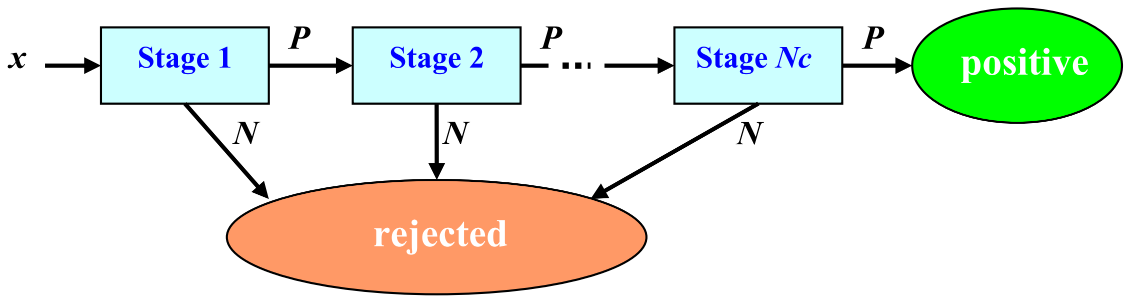


Figure 5.1: Cascade classifier scheme.

Regarding the training process, the first stage is trained by using the entire training dataset, while the other stages are trained by using a sub-set composed by the aggregation of the positive examples, which are the same for all the ensemble stages, with the false positive examples not rejected by the previous stages when running on negative training examples. All the stages are trained in such a way as to achieve high TP. This process is summarized in Algorithm 7 and illustrated in Fig.5.2.

**Algorithm 7** Generic cascade classifier training

```
Input: $TP_{des}$, $S_{neg} = \{x_{(neg,1)}, \ldots, x_{(neg,n_n)}\}$, $S_{pos} = \{x_{(pos,1)}, \ldots, x_{(pos,n_p)}\}$, $N_c$: adopted $TP$,
    sets of negative and positive $n_f$-dimensional training examples, and number of cascade stages
    respectively;
Output: $\{P_k\}$, $\{s_k\}$: sets of parameters of the cascade stages and set of vectors with the indexes
    of the selected features for each stage $k$, respectively;
 1: $k \leftarrow 1$: where $k$ is the stage index;
 2: $S_k \leftarrow S_{neg} \cup S_{pos}$;
 3: while $k \leq N_c$ do
 4:    select a set of $N$ of the $n_f$ features and compose the training dataset $S_{(k,N)}$, whose exemplars
       are obtained from the exemplars of $S_k$ by retaining only the $N$ selected features;
 5:    store the indexes of the selected features in vector $s_k$;
 6:    apply $S_{(k,N)}$ and $\mathcal{L}_{train}$ to train the $k^{th}$ cascade stage, in order to obtain its parameters $P_k$;
 7:    apply $S_{(k,N)}$, $\mathcal{L}_{train}$, and $P_k$ to compute $TP$;
 8:    while $TP < TP_{des}$ do
 9:       increase the classifier bias in order to increase the $TP$, updating $P_k$;
10:       recalculate $TP$ from $S_{(k,N)}$ and $\mathcal{L}_{train}$, using the learned cascade stage with the current
          bias;
11:    end while
12:    scan the current training dataset, $S_k$, by using the current cascade stage, i.e. using the set of
       parameters $P_k$ and the set of features $s_k$, in order to collect a set of false positive occurrences,
       $S_{FP}$;
13:    $S_{k+1} \leftarrow S_{FP} \cup S_{pos}$: composing the dataset for the next stage;
14:    $k \leftarrow k + 1$;
15: end while
```

Notice that, the training dataset used in stage $k$ is a sub-set of the set which was applied in the training of the previous stage, $k-1$. In this context it is important to change the set of features at each stage, which may enable the correct classification of the patterns that were misclassified by the previous stage. Therefore, each stage can have a feature set with its respective cardinality, which implicates in different classifier complexities from a stage to another.

Among several variations of cascade classifiers, we highlight the seminal work of Viola and Jones [VJ01], where it is proposed a boosted cascade classifier scheme, in the framework of image-based object detection, which can be viewed as an object specific focus-of-attention mechanism that discards image regions which are unlikely to contain the objects of interest. In order to decrease the processing time, the boosted

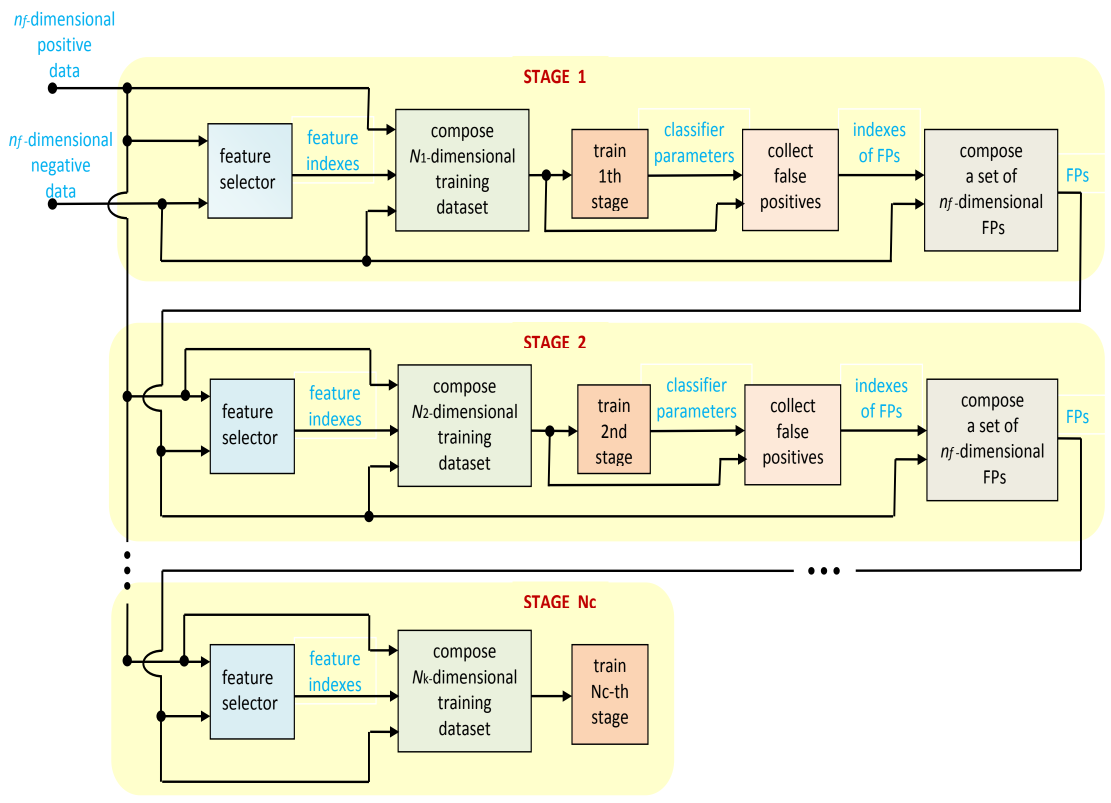


Figure 5.2: Generic cascade classifier training.

cascade applies few features in the first stage, increasing the number of features at each stage, i.e. in case of boosted cascade, the scheme of Fig.5.2 has $N_1 < N_2 \ldots < N_k$.

## 5.2 Upper bound on the expected risk of cascade classifiers

Since the application of SRM principle in cascade classifiers requires the estimation of the upper bound on the expected risk of the entire ensemble, this section presents a VC-style analysis that provides an upper bound on the false positive rate (FP) and a lower bound on the true positive rate (TP), which are composed in order to derive an upper bound on the expected classification risk of the entire ensemble, aiming at controling the number of cascade stages.

### 5.2.1 Brief review on statistical learning theory

This section briefly introduces some principles of statistical learning theory, in order to support our theorems, which require the following definition and lemmas.

**Definition 5.1.** *Let $Q(x, \alpha) = L(y, f(x, \alpha)) \in \{0, 1\}$ be a loss function, in which $y$ is the label of $x$. The growth function is defined as the quantity*

$$G(l) = \ln \sup_{x_1, \ldots, x_l} (N(x_1, \ldots, x_l)) \tag{5.1}$$

*where $N(x_1, \ldots, x_l) \leq 2^l$ is the number of different separations*[1] *that the functions of the set $Q(x, \alpha)$, $\alpha \in \Lambda$ can produce on the set of vectors $x_1, \ldots, x_l$.*

The following lemma, directly derived from the Sauer lemma [Sau72], establishes the relationship between the growth function and the VC-dimension.

**Lemma 5.1.** [Vap98] *Let $h$ be the VC-dimension of the set of functions $f(x, \alpha)$, $\alpha \in \Lambda$. Then, the growth function of a set of indicator functions $Q(x, \alpha)$, $\alpha \in \Lambda$ satisfies the relationship*

$$G(l) \begin{cases} = l \ln 2 \text{ if } l \leq h \\ \leq \ln\left(\sum_{i=0}^{h} C_l^i\right) \leq \ln\left(\frac{el}{h}\right)^h = h\left(1 + \ln \frac{l}{h}\right) \text{ if } l > h \end{cases} \tag{5.2}$$

*where $e$ is the Euler constant and $C_l^i$ is the number of $i$-combinations from a given set $S$ of $l$ elements.*

As can be inferred from Lemma 5.1, even if the set of functions $Q(x, \alpha)$, $\alpha \in \Lambda$, contains infinitely many elements, only a finite number of clusters of events is

[1] *in the context of this chapter, separation means the dichotomizing of the exemplars in some given universe of discourse into two groups.*

distinguishable on the finite set of examples $x_1, \ldots, x_l$. Therefore, in order to derive bounds on the expected risk for the infinite set of indicator functions $Q(x, \alpha)$, $\alpha \in \Lambda$, we take advantage on the relationship stated in the following lemma:

**Lemma 5.2.** [Vap98] *Let $X_1(l)$ and $X_2(l)$ denote two spaces of half-samples of length $l$,*

$$\rho\left(X^{2l}\right) = \sup_{\alpha \in \Lambda} \left| \frac{1}{l} \sum_{i=1}^{l} Q(x_i, \alpha) - \frac{1}{l} \sum_{i=l+1}^{2l} Q(x_i, \alpha) \right| \tag{5.3}$$

and

$$\pi(X_1) = \sup_{\alpha \in \Lambda} \left| \int Q(x, \alpha)\, dF(x) - \frac{1}{l} \sum_{i=1}^{l} Q(x_i, \alpha) \right| \tag{5.4}$$

*Then, the distribution of the random variable $\pi(X_1)$ is connected with the distribution of the random variable $\rho\left(X^{2l}\right)$ by the inequality*

$$P\{\pi(X_1) > \epsilon\} < 2P\left\{\rho\left(X^{2l}\right) > \epsilon - \frac{1}{l}\right\} \tag{5.5}$$

Therefore, to estimate the probability of $\pi(X_1)$, which is our interest, we can estimate the probability of $\rho\left(X^{2l}\right)$, which is based on a finite set of examples $x_1, \ldots, x_l$, that implicates in a finite number of distinguishable events.

To complete the set of foundations required for our study, it is given the following lemma, which defines the rate of convergence that connects two relative frequencies[2]:

**Lemma 5.3.** [Vap98] *For any fixed sample size $2l$, any fixed function $Q(x, \alpha^*)$, any $\epsilon > 0$, and any two randomly chosen half-samples the inequality*

$$P\left\{ \frac{1}{l} \left| \sum_{i=1}^{l} Q(x_i, \alpha^*) - \sum_{i=l+1}^{2l} Q(x_i, \alpha^*) \right| > \epsilon \right\}$$
$$\leq 2e^{-\epsilon^2 l} \tag{5.6}$$

*holds true.* (see Sections 4.5.3 and 4.5.4 of [Vap98]).

[2]In statistics the relative frequency of an event $i$ is the number of times the event occurred in the experiment, normalized by the total number of events.

### 5.2.2 VC-style analysis on cascade classifiers

This section starts by introducing the derivation of the upper bound of FP and the lower bound of TP. These bounds are applied in the derivation of bounds on the BER and classification risk of cascade classifiers. The following theorems derive bounds on FP and TP.

**Theorem 5.1**. *Let $l$ be the cardinality of the training dataset and $n_n$ be the number of negative examples; then, with probability $1-\eta$, $0<\eta<0.5$, the risk for the function $Q\left(x,\alpha_l\right)$ which minimizes the functional $FP_{emp}=\frac{1}{n_n}\sum_{i=1}^{n_n}Q\left(x,\alpha_l\right)$ satisfies the inequality*

$$\left|FP\left(\alpha_l\right)-FP_{emp}\left(\alpha_l\right)\right|\leq\frac{1}{n_n}+\sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right)-\ln\frac{\eta}{4}}{n_n}} \tag{5.7}$$

*where $FP\left(\alpha_l\right)=\int Q\left(x,\alpha_l\right)dF\left(x\right)$ and $Q\left(x,\alpha_l\right)=1$ if $x$ is a false positive, otherwise $Q\left(x,\alpha_l\right)=0$.*

***proof.*** Considering the fixed function $Q\left(x,\alpha^*\right)$ and taking into account that the functional $FP$ is based only on the negative examples, from (5.6), for any fixed sample size $2l$, with $2n_n$ negative examples, and any two randomly chosen half-samples with the same number of negative examples, the inequality

$$P\left\{\frac{1}{n_n}\left|\sum_{i=1}^{n_n}Q\left(x_i,\alpha^*\right)-\sum_{i=n_n+1}^{2n_n}Q\left(x_i,\alpha^*\right)\right|>\epsilon-\frac{1}{n_n}\right\}$$
$$\leq 2e^{-\left(\epsilon-\frac{1}{n_n}\right)^2 n_n} \tag{5.8}$$

holds true. Therefore, taking into account that the number of events depends on the sample size, $2l$, for the set of indicator functions $Q\left(x,\alpha\right)$, $\alpha\in\Lambda$ we have

$$P\left\{\frac{1}{n_n}\sup_{\alpha\in\Lambda}\left|\sum_{i=1}^{n_n}Q\left(x_i,\alpha\right)-\sum_{i=n_n+1}^{2n_n}Q\left(x_i,\alpha\right)\right|>\epsilon-\frac{1}{n_n}\right\}$$
$$\leq\sum_{\alpha^*\in\Lambda^*}P\left\{\frac{1}{n_n}\left|\sum_{i=1}^{n_n}Q\left(x_i,\alpha^*\right)-\sum_{i=n_n+1}^{2n_n}Q\left(x_i,\alpha^*\right)\right|>\epsilon-\frac{1}{n_n}\right\}$$

$$\leq 2N\left(x_1,\ldots,x_{2l}\right)e^{-\left(\epsilon-\frac{1}{n_n}\right)^2 n_n} \leq 2e^{\left(G(2l)-\left(\epsilon-\frac{1}{n_n}\right)^2 n_n\right)} \tag{5.9}$$

where $\Lambda^* = \Lambda^*(x_1\ldots,x_{2l})$ is the finite set of distinguishable functions $Q(x_i,\alpha^*)$. The last inequality of (5.9) came from (5.1).

Combining (5.9) with the statement (5.5) we obtain

$$P\left\{\sup_{\alpha\in\Lambda}\left|E\left(Q\left(x,\alpha\right)\right)-\frac{1}{n_n}\sum_{i=1}^{n_n}Q\left(x_i,\alpha\right)\right|>\epsilon\right\}$$

$$\leq 4e^{\left(G(2l)-\left(\epsilon-\frac{1}{n_n}\right)^2 n_n\right)}. \tag{5.10}$$

Doing

$$\eta := 4e^{\left(G(2l)-\left(\epsilon-\frac{1}{n_n}\right)^2 n_n\right)} \tag{5.11}$$

and solving (5.11) with respect to $\epsilon$, yields

$$\epsilon = \frac{1}{n_n}+\sqrt{\frac{G(2l)-\ln\frac{\eta}{4}}{n_n}} \tag{5.12}$$

From (5.2) and (5.12), for $h<l$, we have

$$\epsilon \leq \frac{1}{n_n}+\sqrt{\frac{h(\ln\frac{2l}{h}+1)-\ln\frac{\eta}{4}}{n_n}} \tag{5.13}$$

Rewriting (5.10) as

$$P\left\{\sup_{\alpha\in\Lambda}\left|E\left(Q\left(x,\alpha\right)\right)-\frac{1}{n_n}\sum_{i=1}^{n_n}Q\left(x_i,\alpha\right)\right|\leq\epsilon\right\}>1-\eta \tag{5.14}$$

and by substituting (5.13) into (5.14) completes the proof. ■

**Theorem 5.2**. *With probability* $1-\eta$, $0<\eta<0.5$, *the risk for the function* $Q\left(x,\alpha_l\right)$ *which maximizes the functional* $TP_{emp}=\frac{1}{n_p}\sum_{i=1}^{n_p}Q\left(x,\alpha_l\right)$ *satisfies the inequality*

$$\left|TP\left(\alpha_l\right)-TP_{emp}\left(\alpha_l\right)\right| \leq \frac{1}{n_p}+\sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right)-\ln\frac{\eta}{4}}{n_p}} \tag{5.15}$$

*where* $TP\left(\alpha_l\right)=\int Q\left(x,\alpha_l\right)dF\left(x\right)$ *and* $Q\left(x,\alpha_l\right)=1$ *if* $x$ *is a true positive, otherwise* $Q\left(x,\alpha_l\right)=0$.

***proof.*** The proof starts by determining the upper bound on the false negative rate (FN) similarly to the proof of Theorem 5.1. Therefore, the following inequality

$$\left|FN\left(\alpha_l\right) - FN_{emp}\left(\alpha_l\right)\right| \leq \frac{1}{n_p} + \sqrt{\frac{h\left(\ln\frac{2l}{h}+1\right) - \ln\frac{\eta}{4}}{n_p}} \tag{5.16}$$

holds with probability $1-\eta$, where $n_p$ is the number of positive examples. Taking into account that $TP\left(\alpha_l\right) = 1 - FN\left(\alpha_l\right)$ and $TP_{emp}\left(\alpha_l\right) = 1 - FN_{emp}\left(\alpha_l\right)$, and substituting these equations in the left-hand side of (5.16) we prove (5.15). ■

Now we have the premises to determine the upper bound on the expected risk, $R(\alpha_k)$, of a $k$-stage cascade ensemble, according to the following corollary:

**Corollary 5.1.** *Let $n_p^t$ and $n_n^t$ be the number of positive and negative occurrences on the test data, $n_p$ be the number of positive training examples, $n_{(n,k)}$ be the number of negative training examples used in the $k^{th}$ stage, $l_k = n_p + n_{(n,k)}$, $h_k$ be the VC-dimension of the $k^{th}$ cascade stage, $TP$ be the true positive rate on the training data, which is arbitrarily adopted for all the stages, and $FP_{(emp,k)}$ the false positive rate on the training data for stage $k$. Then, for a cascade classifier ensemble with $N_c$ stages, the upper bound on the expected misclassification risk is given by the inequality*

$$R(\alpha_k) \leq \frac{e_{max}}{\left(n_p^t + n_n^t\right)}, \tag{5.17}$$

*where*

$$e_{max} = n_p^t\left(1 - \prod_{k=1}^{N_c}\min\left(TP_k\right)\right) + n_n^t\prod_{k=1}^{N_c}\max\left(FP_k\right), \tag{5.18}$$

$$\max\left(FP_k\right) = FP_{(emp,k)} + \frac{1}{n_{(n,k)}} + \sqrt{\frac{h_k\left(\ln\frac{2l_k}{h}+1\right) - \ln\frac{\eta}{4}}{n_{(n,k)}}} \tag{5.19}$$

*and*

$$\min\left(TP_k\right) = TP - \frac{1}{n_p} - \sqrt{\frac{h_k\left(\ln\frac{2l_k}{h}+1\right) - \ln\frac{\eta}{4}}{n_p}} \tag{5.20}$$

*with probability $1-\eta$, $0 < \eta < 0.5$.*

***proof.*** The bounds (5.20) and (5.19) are derived directly from Theorems 5.1 and 5.2, respectively. The stages of a cascade classifier are subject to two kind of errors: false negatives and false positives. If a false negative occurs in the current stage, it cannot be reverted by the next stages; however, false positives can be detected by the other stages, excepting if it occurs in the last stage. The first parcel of (5.18) computes the upper bound on the number of false negatives which occur in all the stages and the second parcel gives the upper bound on the number of false positives that pass by all the stages. ■

### 5.2.3 Theoretical implications

The next step is to derive the number of classifier stages, $N_c^*$, that minimizes the upper bound on the expected risk. The following corollary is derived from Corollary 5.1.

**Corollary 5.2.** *Let $\bar{FP}$ and $\bar{TP}$ be the geometric means of* $\max(FP_k)$ *and* $\min(TP_k)$ *on all the stages, according to*

$$\bar{FP} = \left( \prod_{k=1}^{N_c} \max(FP_k) \right)^{1/N_c} \tag{5.21}$$

*and*

$$\bar{TP} = \left( \prod_{k=1}^{N_c} \min(TP_k) \right)^{1/N_c} \tag{5.22}$$

*Then, the number of stages that minimizes the upper bound on the expected risk is the nearest integer of*

$$N_c^* = \frac{\ln\left(\frac{n_n^t}{n_p^t}\right) + \ln\left(\frac{\ln\left(\bar{FP}\right)}{\ln\left(\bar{TP}\right)}\right)}{\ln\left(\bar{TP}\right) - \ln\left(\bar{FP}\right)} \tag{5.23}$$

*where* $\min(TP_k)$ *and* $\max(FP_k)$ *are given by* (5.20) *and* (5.19) *with probability* $1-\eta$, $0 < \eta < 0.5$.

***proof.*** Equation (5.23) can be obtained by calculating the root of the derivative of (5.18),

$$\frac{\partial e_{max}}{\partial N_c} = n_n^t \ln\left(\bar{FP}\right) \bar{FP}^{N_c} - n_p^t \ln\left(\bar{TP}\right) \bar{TP}^{N_c}, \tag{5.24}$$

with respect to $N_c$. ■

**Corollary 5.3.** *For a cascade classifier ensemble with $N_c$ stages, the upper bound on the expected BER is given by the inequality*

$$BER(\alpha_k) \leq \frac{1}{2}\left(1 - \prod_{k=1}^{N_c} \min\left(TP_k\right)\right) + \frac{1}{2}\prod_{k=1}^{N_c} \max\left(FP_k\right), \tag{5.25}$$

*where* $\min\left(TP_k\right)$ *and* $\max\left(FP_k\right)$ *are given by (5.20) and (5.19) with probability* $1-\eta$, $0 < \eta < 0.5$.

***proof.*** This corollary is proved similarly to Corollary 5.1. ■

Notice that, by substituting $n_n^t = n_p^t = 0.5$ in (5.18) we obtain (5.25). Therefore, by substituting $n_n^t = n_p^t = 0.5$ in (5.23) we obtain the number of stages that minimizes the upper bound on the expected BER, which is the nearest integer of

$$N_c^{**} = \frac{\ln\left(\frac{\ln\left(\bar{FP}\right)}{\ln\left(\bar{TP}\right)}\right)}{\ln\left(\bar{TP}\right) - \ln\left(\bar{FP}\right)} \tag{5.26}$$

As a prelude for analyzing the real experiments in pedestrian detection, which demands a large computational time, a graphical representation of (5.23) was generated, in order to support preliminary qualitative analysis. Figs. 5.3 and 5.4 are a kind of 4D-graphs where the color indicates the upper bound on the expected error, calculated according to (5.18) (the red color indicates high error, while blue indicates low error). The x-axis indicates the geometric mean of the lower bound on $TP$, while the y-axis indicates the geometric mean of the upper bound on $FP$. The z-axis indicates the value of $N_c^*$, calculated according to (5.23). Fig. 5.3 illustrates the ratio $n_n^t/n_p^t = 1$, while Fig. 5.4 illustrates the ratio $n_n^t/n_p^t = 70000$.

These figures help us to highlight some facts regarding the theoretical analysis:

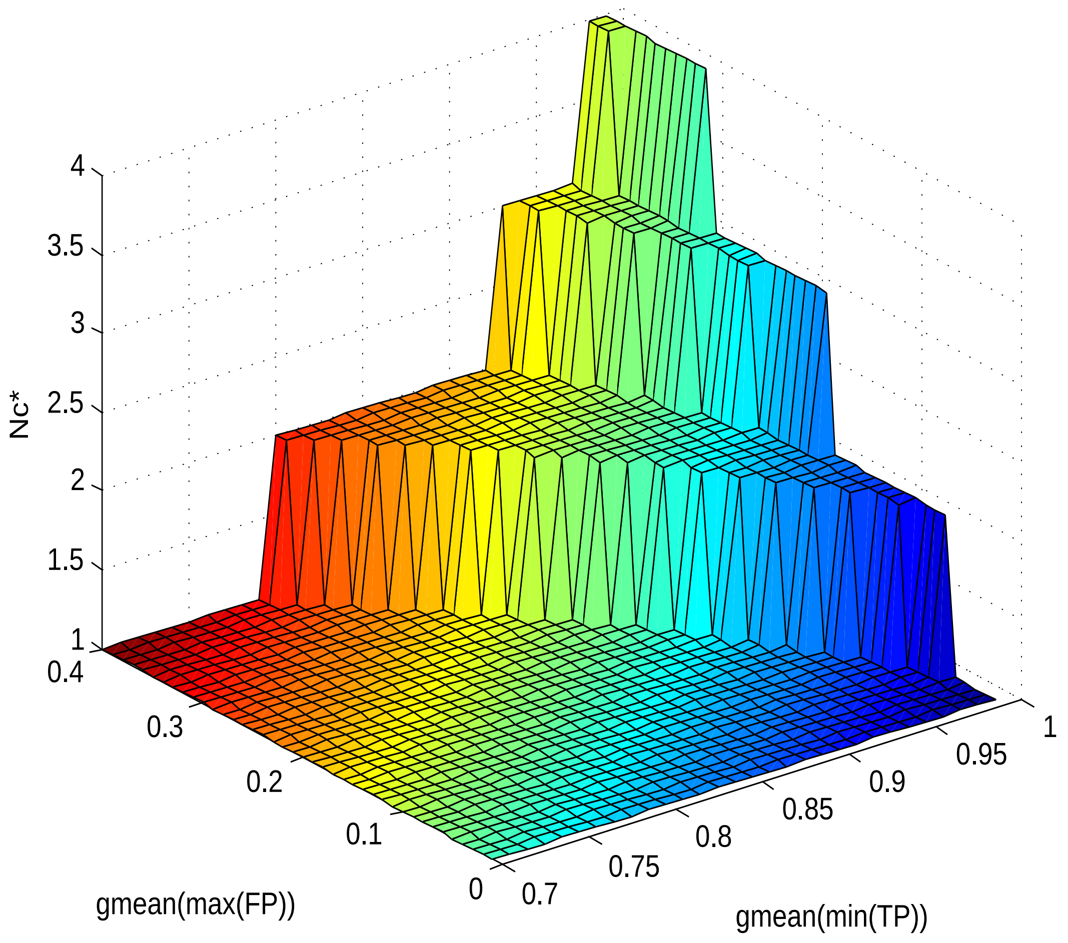


Figure 5.3: Optimal number of stages with $n_n^t/n_p^t = 1$ (*gmean* means geometric mean).

1. By analysing the colors of Figs. 5.3 and 5.4 it is possible to conclude that, if the TP is mantained high, high values of FP can be partialy decreased by increasing the number of stages, i.e., this kind of classifier ensemble is more sensitive to TP than FP, as already expected. Therefore, as usual in previous works, the classifier threshold has to be adjusted in such a way as to enable high TP on the training data set. However, this approach does not assure high TP on the testing dataset, which depends on the number of positive training data, as stated in (5.20);

2. If the number of stages is smaler than $N_c^*$, some false positives are not detected, which increases the total error. However, if the number of stages is bigger than $N_c^*$, the error also increases due to the false negatives (i.e. missing) accumulated along the stages;

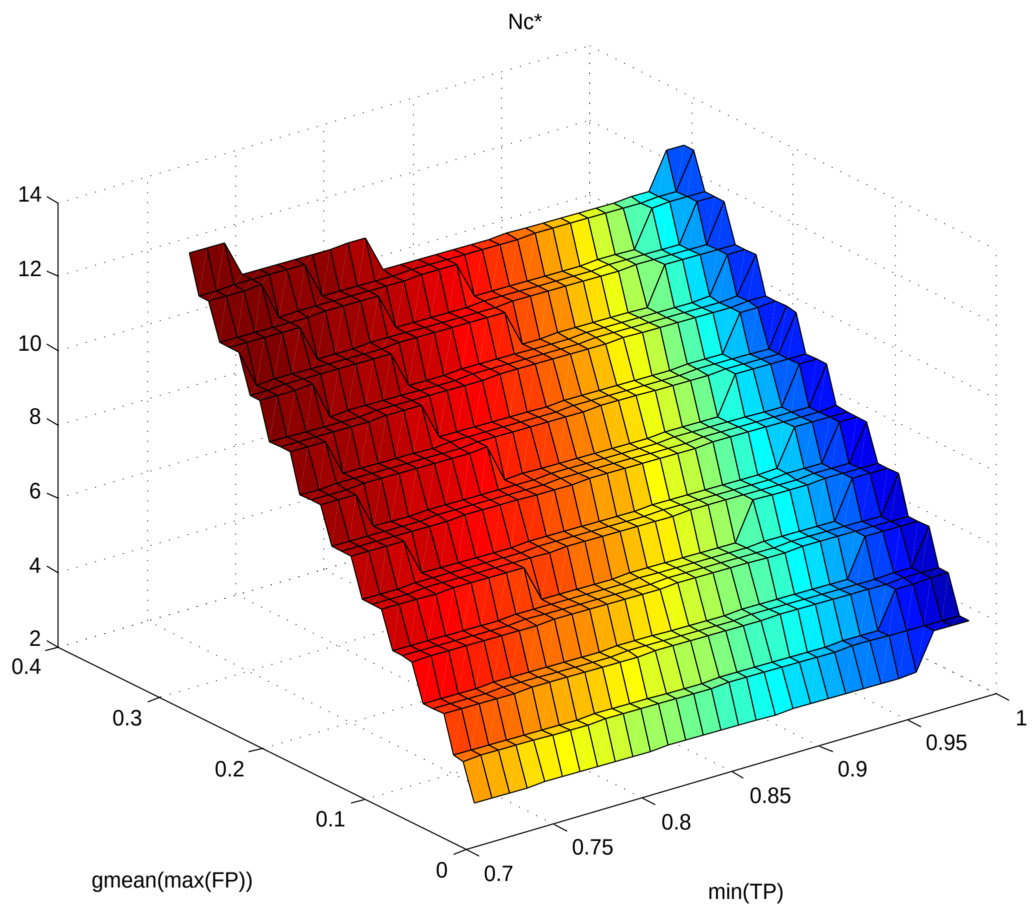


Figure 5.4: Optimal number of stages with $n_n^t/n_p^t = 70000$ (*gmean* means geometric mean).

3. By observing the first term in the numerator of (5.23) it is possible to reason that the larger the ratio $n_n^t/n_p^t$, the larger $N_c^*$, what enables a good performance even under a high FP. Taking into account that pedestrian detection applications have high ratio $n_n^t/n_p^t$, it is possible to understand why cascade ensembles are a favorable choice in such kind of applications;

4. When $n_n^t/n_p^t$ is small, a single classifier may be a better option than a cascade ensemble. Note that Fig. 5.3 has a triangular plateau where the optimal number of classifiers is one, i.e. in this region a single classifier is a better option than a cascade classifier.

## 5.3 SRM-cascade

This section exploits the previous theoretical analysis aiming at the application of the SRM principle in the training of cascade classifiers. This work deals with cascade ensembles composed by linear classifiers, since such model offer a good opportunity to apply SRM schemes, because their VC dimension can be precisely determined, enabling the perfect control on the classifier complexity and, consequently, a better control on the upper bound on the expected risk. Namely, in case of usual linear classifiers, *VC dimension*$= N+1$, where $N$ is the number of features. However, because this work also applies linear SVMs, this value will be used as an upper bound, since the VC dimension of SVM is limited by the margin constraints, *i.e.* in case of linear SVM we have *VC dimension* $\leq N+1$.

SRM is an inductive principle for model selection introduced by Vapnik [Vap98]. It describes a general model of capacity control that optimizes a tradeoff between the quality of the approximation and the hypothesis space complexity, e.g. the VC dimension. The SRM principle can be justified by considering the inequality (4.25). Namely, as the VC dimension, $h$, increases, the minima of empirical risk are decreased; however, the term responsible for the confidence interval, i.e. the second parcel of (4.25), is increased. Fig. 5.5 illustrates a real world example of the tradeoff between the quality of the approximation and the hypothesis space complexity. Namely, this figure illustrates the empirical risk, the confidence interval given by the second parcel of (4.25), and the sum of both terms, i.e. the upper bound on the expected risk, of a linear SVM applied to the Daimler Pedestrian Classification benchmark dataset [MG06]. This example is based on the worst case, i.e. it considers the VC dimension as the feature space dimension plus one.

The SRM procedure divides the class of functions into a hierarchy of nested subsets in order of increasing complexity, which in our case is related to linear models working in feature spaces of increasing dimensionalities. Then it performs empirical risk minimization on each subset and selects the model whose sum of empirical risk and the confidence interval, given by the second parcel of (4.25), is minimal; i.e. it selects the model whose upper bound (4.25) is minimal. In case of SRM-cascade the scheme of Fig.5.2 has the number of features, $N_k$, in each stage $k$ choosen through the application of the SRM procedure at each stage independently; however, the the choice of the number of stages, $N_c$, is based on Corollary 5.1, as summarized in Algorithm 8.

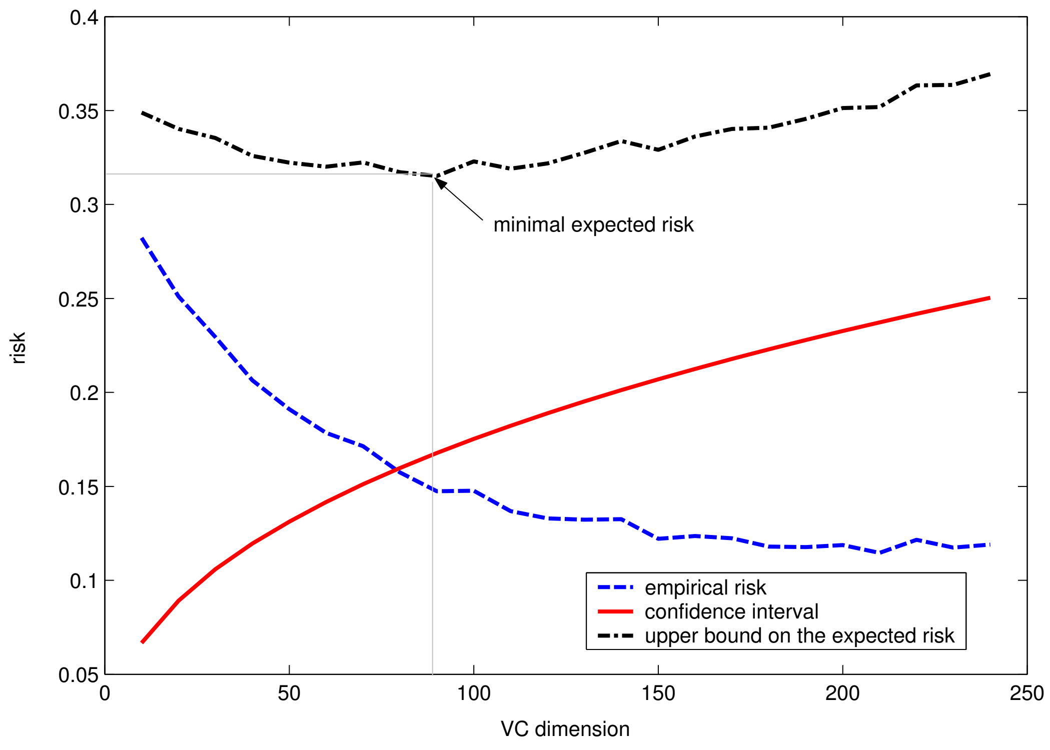


Figure 5.5: Example of empirical risk, confidence interval, and upper bound on the expected risk of a linear SVM applied to the Daimler Pedestrian Classification dataset.

For each stage $k$ an iterative process is applied to determine the optimal number of features (see steps 8 to 25 of Algorithm 8), which in our context is the number of features that results on the minimal upper bound on the expected risk, defined by Theorem 4.1. Therefore, a feature selection algorithm, such as Algorithm 1, is applied recursively, in order to select different numbers of features (step 9), composing different training sets, $S_{(N,n)}$. The bigger the number of features, $N$, the bigger the linear classifier complexity. The ensemble stages are recursively trained in such a way to obtain the desired TP (steps 13 to 16). The trained stage is applied to the training data , in order to obtain the empirical risk, $R_{emp}(\alpha)$ (step 15). Then, $R(\alpha)$ is computed by replacing the number of training data, $l$, and the upper bound on the VC-dimension of the SVM, $h = N + 1$, in (4.25) (step 18). The composition *selected features+classifier* with the smallest $R(\alpha)$ is chosen to compose the current ensemble stage (steps 19 to 24). Then, the upper bound on the expected risk of the entire ensemble, $R(\alpha_k)$, is evaluated by applying (5.17). If $R(\alpha_k)$ increases, the training stops and the current stage is discarded (steps 26 to 35).

As can be seen in Fig. 5.2, excepting the first stage, each stage is trained by using a dataset which was generated by the previous stages (steps 32 and 33). Namely, the training set which is used in the current stage $k$ is composed by all the positives training examples and the false positives not rejected by the previous stages $(1,\ldots,k-1)$.

## 5.4 Weighted Error Rate Minimization

In this section it is proposed a new training method for linear classifiers, named weighted error rate minimization (WERM), which enables the control of the ratio between TP and FP during the training.

As can be inferred from (5.18), if the TP is maintained high, high values of FP can be partially decreased by increasing the number of stages. Therefore, as usual in previous works, the classifier must be adjusted in such a way as to enable high TP on the training dataset. However, this approach does not assure high TP on the test dataset, which depends on the number of positive training data as well as the dataset cardinality. Taking into account that $l = n_n + n_p$, it is possible to state that the larger the number of negative examples, the smaller the lower bound on the expected TP, since $l$ is in the numerator of the right-hand side of (5.15). Therefore, the more unbalanced the dataset, i.e. the larger is the ratio $n_n/n_p$, the more important is to sustain high TP on the training dataset.

The usual approach to achieve high TP is to adjust the classifier threshold, i.e. the bias, after the training. However, this approach can be improved by adjusting all the classifier parameters (not only the bias), in such a way as to achieve the intended TP, without affecting the overall classifier performance.

The idea is to solve the unconstrained optimization problem:

$$\min_{w,b} J = \frac{k}{n_p}\sum_{i=1}^{n_p}\left(1-\hat{y}_{(i,p)}\right)^2 + \frac{1}{n_n}\sum_{j=1}^{n_n}\left(-1-\hat{y}_{(j,n)}\right)^2, \tag{5.27}$$

where

$$\hat{y}_{(i,p)} = w^T x_{(i,p)} + b, \tag{5.28}$$

$$\hat{y}_{(j,n)} = w^T x_{(j,n)} + b, \tag{5.29}$$

$x_{(i,p)}$ is the $i^{th}$ positive training example, $x_{(j,n)}$ is the $j^{th}$ negative training example, and $k > 1$ is a hyperparameter that weights the first parcel. Notice that, minimizing the first parcel is equivalent to minimize the FN, while minimizing the second parcel is equivalent to minimize the FP; therefore, the bigger $k$, the smaller will be the FN after the minimization of $J$.

The classifier is trained iteratively; the algorithm starts with $k = 1$, at each iteration the value of the hiperparameter $k$ is increased by a constant amount, then the optimization problem (5.27) is solved and the obtained parameters $w$ and $b$ are applied to evaluate the TP. This process continues until the desired TP is reached.

The optimization problem (5.27) requires to solve the system of equations $\frac{\partial J}{\partial w} = 0$ and $\frac{\partial J}{\partial b} = 0$ with respect to $w$ and $b$, which yields:

$$w^T = \left(c_1 - c_2 + \frac{c_6 c_3}{c_5}\right) c_4^{-1} \left(I_{(n\times n)} - \frac{c_3^T c_3}{c_5} c_4^{-1}\right)^{-1} \tag{5.30}$$

and

$$b = \frac{-c_6 - w^T c_3^T}{c_5}, \tag{5.31}$$

where

$$c_1 = \frac{2k}{n_p} \sum_{i=1}^{n_p} x_{(i,p)}^T, \tag{5.32}$$

$$c_2 = \frac{2}{n_n} \sum_{j=1}^{n_n} x_{(j,n)}^T, \tag{5.33}$$

$$c_3 = c_1 + c_2, \tag{5.34}$$

$$c_4 = \frac{2k}{n_p} \sum_{i=1}^{n_p} x_{(i,p)} x_{(i,p)}^T + \frac{2}{n_n} \sum_{j=1}^{n_n} x_{(j,n)} x_{(j,n)}^T, \tag{5.35}$$

$$c_5 = 2\left(k+1\right), \tag{5.36}$$

$$c_6 = 2\left(1-k\right) \tag{5.37}$$

**Algorithm 8** SRM-cascade training

```
Input: $TP_{des}$, $S_{neg} = \left\{x_{(neg,1)}, \ldots, x_{(neg,n_n)}\right\}$, $S_{pos} = \left\{x_{(pos,1)}, \ldots, x_{(pos,n_p)}\right\}$, $N_c$: adopted $TP$,
    sets of negative and positive $n_f$-dimensional training examples, and number of cascade stages
    respectively;
Output: $\{w_k^*\}$, $\{b_k^*\}$, and $\{s_k^*\}$: sets of classifier parameters and set of vectors with the indexes of
    the selected features for each stage $k$, respectively;
 1: $k \leftarrow 1$: where $k$ is the stage index;
 2: $S_k \leftarrow S_{neg} \cup S_{pos}$;
 3: $step \leftarrow$ increment in the number of features;
 4: $R_{all}^* \leftarrow 1$: upper bound on expected risk of the entire ensemble;
 5: $flag \leftarrow 0$;
 6: while $flag = 0$: loop over the stages; do
 7:   $R^* \leftarrow 1$: upper bound on expected risk of the current stage;
 8:   for $N = 1 : step : n_f$ do
 9:     select a set of $N$ of the $n_f$ features by applying a feature selector method [LN10] to the $S_k$
        dataset, and compose the training dataset $S_{(k,N)}$, whose exemplars are obtained from the
        exemplars of $S_k$ by retaining only the $N$ selected features;
10:     store the indexes of the selected features in vector $s$;
11:     apply $S_{(k,N)}$ and $\mathcal{L}_{train}$ to train a linear SVM or WERM, in order to obtain the classifier
        parameters $w_{(k,N)}$ and $b_{(k,N)}$;
12:     apply $S_{(k,N)}$, $\mathcal{L}_{train}$, $w_{(k,N)}$, and $b_{(k,N)}$ to compute $R_{emp}(\alpha)$ and $TP$;
13:     while $TP < TP_{des}$ do
14:       $b_{(k,N)} \leftarrow b_{(k,N)} + 0.05$: increasing the bias (or $k$ in the case of WERM) in order to increase
          the $TP$;
15:       recalculate $TP$ and the empirical risk, $R_{emp}$, from $S_{(k,N)}$ and $\mathcal{L}_{train}$, using the learned
          classifier model with the current bias $b_{(k,N)}$;
16:     end while
17:     $h \leftarrow N + 1$: VC-dimension of the current classifier;
18:     replace $l = |S_{(k,N)}|$, $R_{emp}$, and $h$ in (4.25), in order to obtain the expected risk $R(\alpha)$;
19:     if $R(\alpha) < R^*$ then
20:       $R^* \leftarrow R(\alpha)$;
21:       $w_k^* \leftarrow w_{(N,k)}$;
22:       $b_k^* \leftarrow b_{(N,k)}$;
23:       $s_k^* \leftarrow s$; // feature indexes
24:     end if
25:   end for
26:   calculate the upper bound on expected risk of the entire ensemble, $R(\alpha_k)$, by applying (5.17);
27:   if $R(\alpha_k) \geq R_{all}^*$ then
28:     $flag \leftarrow 1$: stopping training;
29:     $k \leftarrow k - 1$: discarding the current stage (making the previous stage as the last cascade
        stage);
30:   else
31:     $R_{all}^* \leftarrow R(\alpha_k)$;
32:     scan the current training dataset, $S_k$, by using the current cascade stage, i.e. using the
        classifier with parameters $w_k^*$, $b_k^*$, and the set of features $s_k^*$, in order to collect a set of false
        positive occurrences, $S_{FP}$;
33:     $S_{k+1} \leftarrow S_{FP} \cup S_{pos}$: composing the dataset for the next stage;
34:     $k \leftarrow k + 1$: increasing the stage index;
35:   end if
36: end while
```

# Chapter 6

# Experiments on Pedestrian Detection and Classification

FOR evaluating the performance of the proposed methods, two kind of problems are considered: pedestrian classification and pedestrian detection. The first problem is approached by applying a MLP on a balanced dataset: the Daimler Pedestrian Classification benchmark [MG06]. The second problem, pedestrian detection, is approached by applying a cascade classifier on the Laser and Image Pedestrian Detection (LIPD) dataset [LPNR11], a highly unbalanced dataset.

## 6.1 Image descriptors

The experiments performed in this work apply two image descriptors: histogram of oriented gradients (HOG) [DT05] and covariance features (COV) [TPM06], [TPM07].

The COV descriptor applied in this work computes four sub-regions within a region $R$, which represents the area of a cropped image. Each sub-region overlaps half of its area. Let $I$ be the input image matrix, and $z_p$ the corresponding $d$-dimensional feature vector calculated for each pixel $p$:

$$z_p = \left[x, y, |I_x|, |I_y|, \sqrt{I_x^2 + I_y^2}, |I_{xx}|, |I_{yy}|, arctan\frac{|I_y|}{|I_x|}\right] \tag{6.1}$$

where $x$ and $y$ are the pixel $p$ coordinates, $I_x$ and $I_y$ are the first order intensity derivatives regarding to $x$ and $y$ respectively, $I_{xx}$ and $I_{yy}$ are the second order derivatives, and the last term in (6.1) is the edge orientation. For the $i^{th}$ rectangular sub-region $R_i$, the covariance matrix $C_{R_i}$ is expressed by

$$C_{R_i} = \frac{1}{N_i - 1} \sum_{p=1}^{N_i} (z_p - \mu_i)(z_p - \mu_i)^T \tag{6.2}$$

where $\mu_i$ is the statistical mean of $z_p$ over the $i^{th}$ sub-region $R_i$ and $N_i$ is the number of pixels in $R_i$. Notice that, due to the symmetry of $C_{R_i}$, only the upper triangle part need to be stored, hence the covariance descriptor of a sub-region is an $8 \times 8$ matrix. The features of the whole region $R$ are also calculated, therefore a feature vector with 180 features is generated, i.e. 4 sub-regions $R_i$, totalizing 144 features, plus 36 features of the whole region $R$.

Regarding the HOG descriptor, the histogram channels were calculated over rectangular cells (*i.e.* R-HOG) by the computation of unsigned gradient. The cells overlap half of their area, meaning that each cell contributes more than once to the final feature vector. In order to account for changes in illumination and contrast, the gradient strengths were locally normalized, *i.e.* normalized over each cell. The HOG parameters were adopted after a set of experiments performed over the training dataset. The better accuracy was achieved by means of 9 rectangular cells and 9 bin histogram per cell. The nine histograms with nine bins were then concatenated to make a 81-dimensional feature vector. The Matlab source code of the HOG descriptor applied in this experiment was made available for download at Matlabcentral[1].

## 6.2 Experiments on pedestrian classification

In this section the contributions introduced in Chapter 4 and part of the contributions of Chapters 3 and 5 are evaluated. Namely, the Daimler Pedestrian Classification dataset [MG06] is used to evaluate the training method for linear classifiers, WERM, and all the MLP training methods proposed in Chapter 4.

From the Daimler Pedestrian Classification dataset were extracted HOG and COV features. The experiments with MLPs made use of HOG features, while the experiments with WERM made use of both HOG and COV features, since such experiment apply a boosted cascade classifier, which demands a large amount of features.

[1]http://www.mathworks.com/matlabcentral/fileexchange/28689-hog-descriptor-for-matlab

### 6.2.1 Evaluating the proposed training methods for MLP

In this sub-section the proposed MLP training methods are evaluated. The MLP architecture was chosen by means of 5-fold cross validation on the training dataset. After such procedure, it was addopted an MLP architecture with 20 hidden neurons for all the training methods, excepting for the RNN. The RNN was composed by four MLPs, which were previously trained by GDX, MMGDX, SVNN, and TNN, in order to improve the diversity of behaviours among the neural subsystems, as suggested by Corollary 4.1, which is based on the assumption that the distribution of the error among the neural subsystems is independent. All the subsystems have 20 hidden neurons, therefore, the RNN is homogeneous, according to the assumption of Corollary 4.1. Since the Daimler dataset has large cardinality, the computational effort was high for all the proposed training methods. The following paragraphs report the specificities of each training method.

The MMGDX had to be applied iteratively by changing the Lp-norm, from smaller to larger $p$, in order to avoid optimization problems that usually occur when using high Lp-norms on large datasets. The training started with the L4-norm, followed by other 4 training stages with the norms L8, L10, L12, and L14. The training was performed with 4000 epochs per stage, using $\alpha = 10^{-3}$ and $\beta = 10^{-4}$. The total CPU time was around 28 hours. The amount of memory required by MMGDX was small when compared with SVM-RBF, which was time- and space-consuming when running on the Daimler dataset.

In the case of SVNN the training was split into two stages, in order to deal with memory constraints. In the first stage the GA parameters were set as $a = 6$, $max_{gener} = 40$, $N_{pop} = 10000$. In order to refine the search, the second stage started by seeding an individual, composed by the MLP parameters obtained in the first stage, in the initial GA population. In the second stage the GA parameters were set as $a = 4$, $max_{gener} = 20$, $N_{pop} = 5000$. The idea is to avoid a large number of individuals, i.e. to save memory. The parameter of SVNN objective function was set as $C = 10^4$ and the total CPU time was around 64 hours.

The TNN parameters were set as $C_1 = 7 \times 10^3$, $C_2 = 3 \times 10^3$, $C_3 = 0$. The MLP parameters obtained by SVNN were used to compose a chromossome that was seeded in the initial GA population, in order to speed up the TNN algorithm convergence.

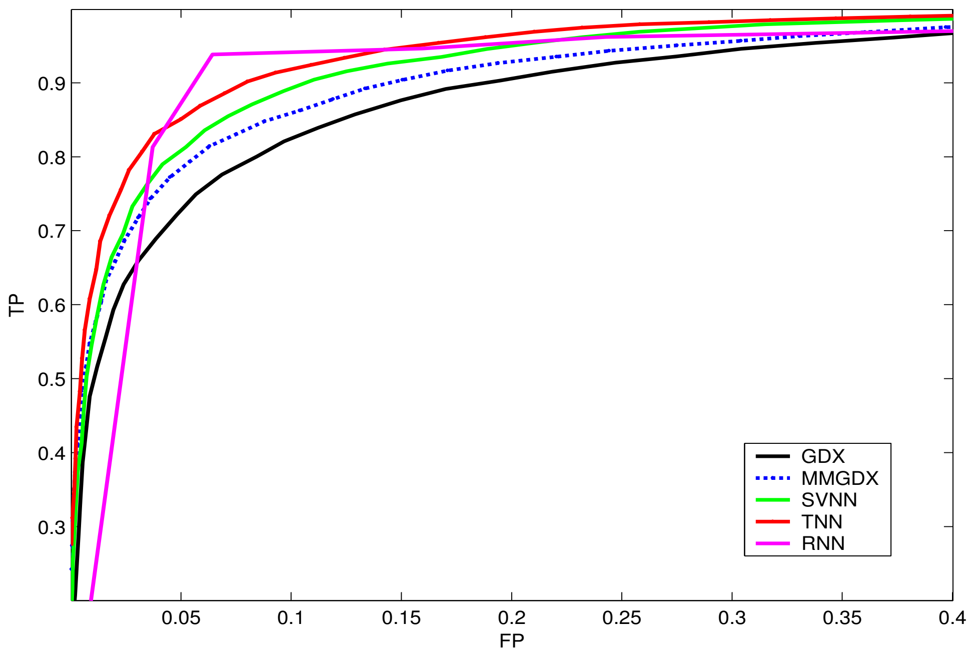


Figure 6.1: ROC curves relative to the proposed MLP training methods applied to the Daimler Pedestrian Classification dataset.

The GA parameters were set as $a = 6$, $max_{gener} = 20$, $N_{pop} = 7000$. The total CPU time was around 33 hours.

Figure 6.1 illustrates the ROC curves of all the proposed methods plus the ROC curve of GDX, while Table 6.1 summarizes the performance indexes of all the training methods plus the performance indexes of SVM with RBF kernel [LDGN09]. The RNN had the best performance indexes, excepting for the AUC. Notice that, due to the step-like activation function, the RNN output is quantized in five discrete levels, since such model is composed by four neural subsystems. Therefore, as can be seen in Figure 6.1, the ROC curve can only have six points, which limits the AUC.

Another interesting fact is the relationship between the accuracies on the training and testing datasets in the case of SVNN and TNN. Note that due to the regularization and the margin constraints the NN cannot fit well the training dataset; however, the large margin leads to a better performance on the testing dataset; specially in the case of TNN, since this alghorithm also takes into account margin constraints on the

Table 6.1: Performance indexes on Daimler dataset

| **method** | $acc_{train}$ | $acc_{test}$ | **TP** | **FP** | **BER** | **AUC** |
|---|---|---|---|---|---|---|
| GDX | 0.9070 | 0.8628 | 0.8792 | 0.1530 | 0.1369 | 0.9373 |
| MMGDX | 0.9077 | 0.8788 | 0.8902 | 0.1322 | 0.1210 | 0.9515 |
| SVNN | 0.8971 | 0.8965 | 0.9065 | 0.1130 | 0.1033 | 0.9613 |
| TNN | 0.8818 | 0.9089 | 0.9171 | 0.0990 | 0.0910 | **0.9704** |
| RNN | **0.9557** | **0.9369** | **0.9382** | **0.0643** | **0.0631** | 0.9506 |
| SVM-RBF | - | 0.8828 | 0.9019 | 0.1356 | 0.1168 | 0.9543 |

Table 6.2: Number of calculations per input data

| **classifier model** | **# sum** | **# product** | **# nonlinear funct.** | **Equation** |
|---|---|---|---|---|
| MMGDX/SVNN/TNN | 1661 | 1640 | 20 | (4.3) |
| RNN | 6648 | 6564 | 84 | (4.22) |
| SVM-RBF | 344646 | 348849 | 4203 | (4.49) |

testing dataset.

Regarding the computational effort in classifying images, the neural models were quite less expensive than the SVM-RBF, since the trained SVM-RBF has 4203 support vectors. Table 6.2 presents the number of calculations required to classify a 81-dimensional input data (features HOG) for each model, with the last column indicating the classifier model.

Note that GDX, MMGDX, SVNN, and TNN have 20 hidden neurons, while RNN has 80 neurons in the first hidden layer and 4 neurons in the second hidden layer. As can be seen in Table 6.2, the computational cost of the SVM-RBF hinders on-the-fly applications.

### 6.2.2 Evaluating WERM

The experiment reported in this sub-section uses the Daimler Pedestrian Classification dataset to evaluate WERM and SVM in the training of linear classifiers. The results are illustrated in Fig.6.2 in the form of two ROC curves. The blue curve was obtained by repeating the WERM training with different values of the hyperparameter $k$, see

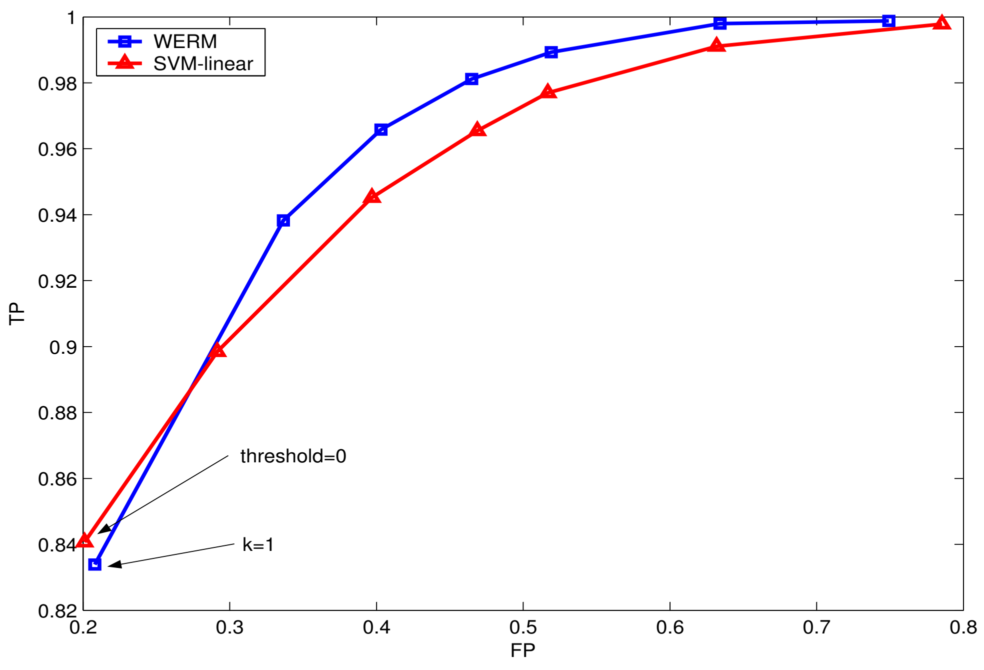


Figure 6.2: ROC curves resulting from the application of a linear model on the Daimler dataset. The blue curve was obtained by repeating the WERM training with different values of the hyperparameter $k$, while the red curve was obtained by varying the threshold after a single training with SVM.

(5.27), while the red curve was obtained by varying the threshold, i.e. the parameter $b$ of (4.49), after a single training with SVM. The idea is to allow a comparative analysis of their behaviors for high values of TP, as required in the training of cascade classifiers. In order to reproduce the usual conditions in the training of cascade classifiers, both methods make use of only 20 features, selected from the set of 261 features generated from HOG and COV descriptors by using the feature selector introduced in Chapter 3.

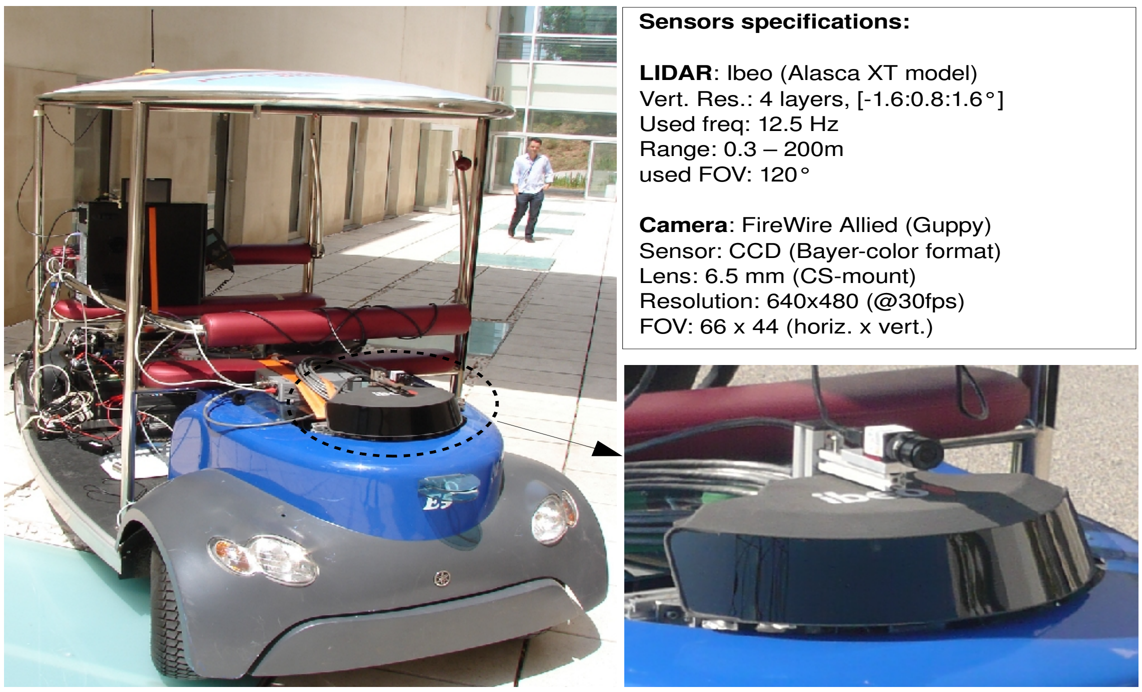
Notice that, in spite of the slightly better performance of SVM for the threshold $= 0$, i.e. the performance for the value of $b$ that fits the objective function and constraints of the SVM training, WERM presents better performance when both methods are forced to improve the TP. For instance, for TP $= 0.98$, SVM has FP $= 0.54$, while WERM has FP $= 0.46$, which seems to be a suitable value for only 20 features. This low FP makes possible to decrease the number of cascade stages, which decreases the computational cost.

## 6.3 Experiments on pedestrian detection

Pedestrian protection systems can be divided, in general, in two fields of research: passive and active safety systems [GT07]. Active safety systems, which is of interest here, are based on pedestrian detection using sensors on-board the vehicle, and/or on the infrastructure, with the role of predicting and anticipating possible risks of collision. In particular, active pedestrian detection systems using on-board laserscanner and monocular camera will be emphasized in these experiments. More specifically, this experimental evaluation is focused on cascade ensembles of SVMs designed to detect pedestrian evidences inside ROIs generated by a laserscanner-based processing module. The proposed cascades, involving a series of SVMs, perform direct negative rejection in each stage, with the purpose of reducing the number of negatives and the computational time in the subsequent stages, which is of particular importance in pedestrian detection, since a key problem in monocular image-based pedestrian detection, namely in the field of Advanced Driver Assistance Systems (ADAS) applications, is the huge amount of negatives (potential false alarms) in contrast with the number of positives, which demands a vast processing time consumption and a high confidence detector.

### 6.3.1 The LIPD dataset

The LIPD dataset [PLN09a], which was collected, labeled, and arranged by our colleague Cristiano Premebida, contains, besides images from a monocular camera and scans from a 4-layers laserscanner, data from two proprioceptive sensors, an IMU and an incremental encoder, in conjunction with DGPS and battery-bank state data (terminal voltage, current and temperature). The dataset was recorded using the sensor system mounted on the ISRobotCar (autonomous electric vehicle with a chassis from Yamaha Europe and control systems developed in the Institute for Systems and Robotics of Coimbra University), see Fig. 6.3.



**Sensors specifications:**

**LIDAR**: Ibeo (Alasca XT model)
Vert. Res.: 4 layers, [-1.6:0.8:1.6°]
Used freq: 12.5 Hz
Range: 0.3 – 200m
used FOV: 120°

**Camera**: FireWire Allied (Guppy)
Sensor: CCD (Bayer-color format)
Lens: 6.5 mm (CS-mount)
Resolution: 640x480 (@30fps)
FOV: 66 x 44 (horiz. x vert.)

Figure 6.3: ISRobotCar: a electric vehicle and its sensors setup, enlarged at the bottom-right part, used in the dataset collection. A short specification of the sensors are presented at the top-right side of the figure.

The ISRobotCar was driven through areas of the engineering *campus* of the University of Coimbra and neighboring areas[2]. Table 6.3 outlines the sensors and their manufacturers, the data communication interface protocols, and the frequency of data acquisition used to record the dataset in a host PC.

Table 6.3: LIPD dataset: sensors, interfaces and used acquisition frequency

| Sensor | Manufacturer | Interface | Acquisition rate |
|---|---|---|---|
| LIDAR (Alasca-XT) | Ibeo | Ethernet | 12.5Hz |
| Camera (Guppy) | Allied | FireWire | 30fps |
| IMU | XSens | USB | 120Hz |
| DGPS | TopCon | USB | 5Hz |
| Encoder and batteries | - | USB | 10Hz |

The manual labeling process, inherent to any supervised dataset, was carried out using the image frames as primary reference for pedestrian and non-pedestrians annotation. The labeled segments, extracted from raw data laser-scans, were validated using the corresponding image frame (for *ground truth* confirmation). All the samples of interest were labeled under user supervision.

[2]http://www.isr.uc.pt/cpremebida/PoloII-Google-map.pdf

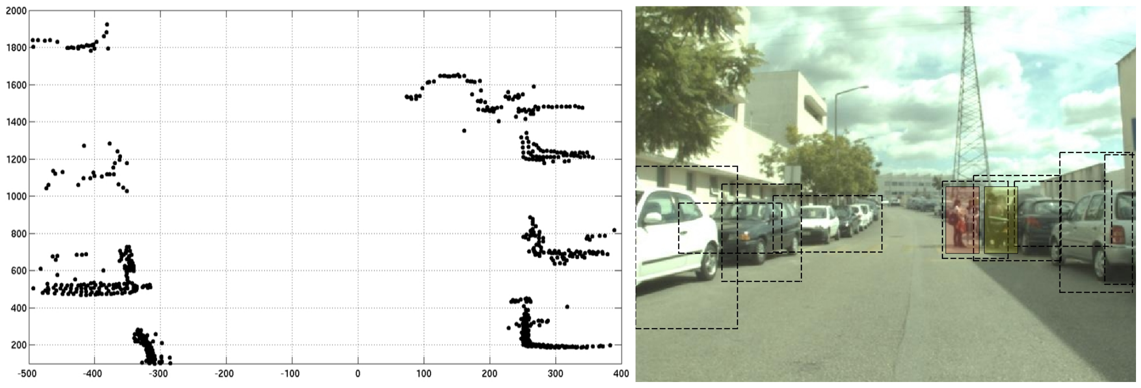


Figure 6.4: Laser-based ROI projections in an image frame, with clusters of range points in the left part and their projections in the image frame, represented by dashed regions.

In summary, the LIPD dataset comprises a training dataset, ($\mathcal{D}_{train}$) and a testing dataset, $\mathcal{D}_{test}$. $\mathcal{D}_{train}$ is used to train the classifier parameters and also to perform cross-validation and feature selection, while $\mathcal{D}_{test}$ is used to evaluate the performance of the proposed techniques. The training dataset contains 5237 positive occurrences composed by laser-segments and ROIs (see Fig. 6.4), i.e. cropped images of pedestrians, and 6328 full-frames of 640x480 resolution images without any pedestrian evidence, i.e. a free-number of negative ROIs can be extracted from such negative frames by using a sliding window. The testing dataset contains 4823 full-frame of 640x480 resolution images with detailed annotations regarding the pedestrian appearance, in terms of occlusion [EG09], more specifically, occluded/partial pedestrians are labeled as class type 0, while entire body pedestrians are labeled as class type 1.

**Definition 6.1.** *a positive sample is defined by an entire body pedestrian (PED) present in both the camera and laser field of view (FOV). A negative sample is defined by any other object (nPED) present in the FOV of both sensors, while an occluded pedestrian denotes a partial occluded PED.*

**Definition 6.2.** *ROIs are projections of laser-segment returns onto the image, defined considering the extremes of a segment rather than individual laser-points. The ROIs*

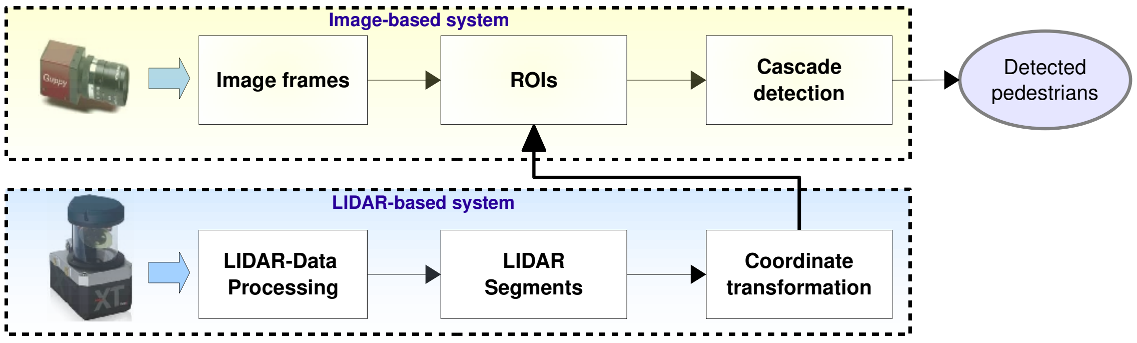


Figure 6.5: Pedestrian detection architecture illustrating the main processing modules in the laserscanner and Image-based systems [PLN09a].

*coordinates are calculated under the assumption of flat surface, knowing the distance of the laser setup from the ground and considering the maximum height of the objects as 2.5m.*

### 6.3.2 Method to define ROIs in the image

A key problem in monocular image-based pedestrian detection is the huge amount of negatives in contrast with the number of positives, what demands a vast processing time consumption and a high confidence detector. An usual solution adopted to avoid using brute-force multiscale sliding windows approach is to combine, in a sensor fusion architecture, the image-based sensors with active sensors like laserscanners. Therefore, in this work the laserscanner acting as a primary object detection subsystem, where each detected object, represented by its laser-segments, constitutes a hypothesis of being a positive or a negative. The laser-segments are projected in the image plane, generating regions of interest, ROIs, which are scanned by a sliding window, as illustrated in Fig. 6.5.

The principal processing modules of our pedestrian detection system are described below[3]:

1. Laserscanner-data processing: a module comprising a set of data processing tasks that decrease complexity and processing time of subsequent stages, such as: filtering-out isolated/spurious range-points, discarding measurements that

[3]Details about the processing modules used in this work are given in [PLN09b].

occur out a predefined FOV, and data alignment;

2. Laserscanner segmentation: this module outputs a set of segments obtained by a range-data segmentation method applied per laserscanner layer, which are then combined;

3. Coordinate transformation: a module that performs a set of rigid coordinate transformations, obtained by system calibration [ZP04], to project laser-segments into the image plane. This module outputs a set of ROIs;

4. Cascade detection: this module combines HOG and COV descriptors with a cascade of SVMs, aiming at detecting potential pedestrians inside the ROIs, by using a sliding window.

The number of window detectors, used to scan the ROIs in searching for pedestrian evidence, is limited and it is defined by the size of the ROI. These window detectors are shifted by horizontal and vertical step factors, and the window scale is estimated using the depth information provided by the ROIs, i.e. laserscanner measurements are also used for scale estimation. This approach decreases the computational processing time, restricting the areas of interest in the image, at most, a dozen ROIs, keeping the false positives at low values. For instance, the number of window detectors generated by this laserscanner-based approach is, in average, thousands times lower than the usual full-scanning image approach.

### 6.3.3 Training data selection

Image-based pedestrian detection using multi-scale sliding window approach demands a large computational effort, and faces a very unbalanced number of negatives $n_n$ against the positives $n_p$, i.e. $n_p \ll n_n$. To avoid bias problems and unfeasible computational requirements in such large unbalanced datasets, a under-sampling algorithm is desirable [KC06]. In order to preserve the information which is relevant to compute the classifier separating hypersurface and, at the same time, to reduce the training dataset cardinality, it is applied a SVM-based data selection (under-sampling) algorithm, developed by our collegue Cristiano Premebida [LPNA11], which is inspired in the parallel SVM architecture introduced in [GCB$^+$04]. In short terms, the data re-sampling algorithm applied in this work selects, from the negative training set ($S_{neg}$), a set of instances which correspond to support vectors ($S_{\mathcal{SV}}$).

Given the initial training set $S_{neg}$, with $n_n$ negative training examples, our undersampling algorithm selects $n_s$ instances which correspond to the support vectors of $S_{neg}$. The first step of the resampling algorithm is to split $S_{neg}$ in $n_S$ subsets $S_i \subset S_{neg}, i = 1, \cdots, n_S$; further, for each subset $S_i$, a SVM classifier is used to extract the support vectors which will be used to compose a set of negative support vectors, $S_{\mathcal{SV}}$. Thus, each $i^{th}$-SVM is trained with a subset comprising $n_p$ positives and $\frac{n_n}{n_S}$ negatives. The final step is to aggregate all the negative support vectors obtained from the $n_S$ SVMs. Lastly, the final training set, $S$ is composed by aggregating the selected negative support vectors, $S_{\mathcal{SV}}$, and the $n_p$ positive examples, i.e. $S = S_{\mathcal{SV}} \cup S_{pos}$.

### 6.3.4 SRM-cascade training

This sub-section describes the application of the proposed SRM-cascade in pedestrian detection using the LIPD dataset, which requires the pre-processing of image frames before applying Algorithm 8. The training process starts by collecting approximately $1.8 \times 10^7$ non-pedestrians samples by using a multiscale sliding window approach inside regions of interest (ROIs), which are defined by a laserscanner-based detection system on the 6328 full-frames without pedestrians of the training dataset, see step 1 of Algorithm 9. From each cropped image a set of features are extracted by using two image descriptors: HOG and COV (step 2 and 3). Then, the data selector described in Section 6.3.3 is employed to reduce the number of negative examples from approximately $1.8 \times 10^7$ to 151528 (step 4). Finally, the cascade training is carried out by using 151528 non-pedestrian examples and 5237 pedestrian examples (step 5). Fig. 6.6 details the training dataset composition.

As detailed in Algorithm 8 in Section 5, for each stage $k$ an iterative process is applied, in order to determine the optimal number of features that in our context is the number of features for which results the minimal upper bound on the expected risk. Therefore, the feature selector is applied iteratively, in order to select different numbers of features, $N$, from both HOG and COV descriptors. These data compose different training sets, $S_{(N,n)}$. The ensemble stages are trained in order to obtain the empirical risk, $R_{emp}(\alpha)$. Then, the expected risk, $R(\alpha)$, is computed by (4.25), setting the number of training data, $l$, and the upper bound on the VC-dimension

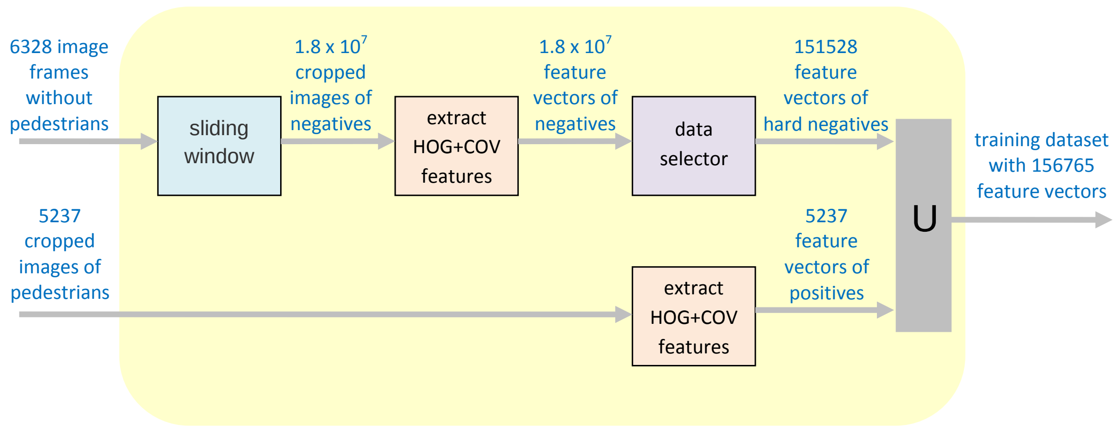

6328 image frames without pedestrians
sliding window
1.8 x 10^7 cropped images of negatives
extract HOG+COV features
1.8 x 10^7 feature vectors of negatives
data selector
151528 feature vectors of hard negatives
U
training dataset with 156765 feature vectors
5237 cropped images of pedestrians
extract HOG+COV features
5237 feature vectors of positives

Figure 6.6: Training dataset composition.

**Algorithm 9** Training the SRM cascade by using the LIPD dataset

```
Input: $\mathcal{D}_{train} = \mathcal{D}_{pos} \cup \mathcal{D}_{neg}$, $N_c$, and $TP_{des}$: training dataset with cropped images of pedestrians ($\mathcal{D}_{pos}$) and frames without pedestrians ($\mathcal{D}_{neg}$), number of cascade stages, and desired TP respectively;
Output: $\{w_n^*\}$, $\{b_n^*\}$, and $\{s_n^*\}$: sets of SVM parameters and set of vectors with the indexes of the selected features for each stage $n$, respectively;
1: collect non-pedestrians samples from $\mathcal{D}_{neg}$, by using sliding window approach, in order to compose a training dataset with cropped images of non-pedestrians $\mathcal{D}_{neg}^*$;
2: extract $n_f$ HOG and COV features from $\mathcal{D}_{pos}$, in order to generate the training dataset $S_{pos}$ of $n_f$-dimensional exemplars;
3: extract $n_f$ HOG and COV features from $\mathcal{D}_{neg}^*$, in order to generate the training dataset $S_{neg}^{all}$ of $n_f$-dimensional exemplars;
4: apply the data selector described in Section 6.3.3 on $S_{neg}^{all}$ and $S_{pos}$, in order to generate the smaller dataset $S_{neg}$;
5: apply $S_{neg}$ and $S_{pos}$ to Algorithm 8;
```

of the SVM space, $h = N + 1$. The composition *selected features+SVM* with the smallest $R(\alpha)$ is chosen to compose the current ensemble stage. The training stops when the upper bound on the expected risk of the entire ensemble, given by (5.17), does not decrease.

### 6.3.5 Scanning settings

Since we have the ground truth of our testing dataset, it is possible to calculate the performance indexes of the proposed cascade ensemble, and consequently to empirically evaluate our theoretical analysis.

The sliding window parameters used in the testing followed the setting $S_4$, from [EG09]: spatial stride ($\Delta_x = 0.1, \Delta_y = 0.025$) and scale step $\Delta_s = 1.25$. The samples were extracted from the ROIs projections instead of the whole image.

In our experiments a miss is computed when a pedestrian (class 1) is not detected, and a false positive occurs when an area with no label is detected. The methodology used to assess the detection performance of the cascade is similar to the one described in [EG09]: the matching criterion is based on the intersection area, in pixels, between a window detector and the ground-truth bounding-box; if the area of a window detector covers more than 25% of the area of a ground-truth event, then a TP is considered. Moreover, a nonmaximum suppression method [Dal06] is used to discard overlapping detections, i.e. detections at close locations. Thus, from the set of windows containing detected objects with a ratio of intersection area above 0.6 it is retained the detection with the greatest confidence and the remaining are discarded.

### 6.3.6 Evaluating the SRM-cascade

For the purpose of comparison, we made experiments with SRM-cascade and boosted cascade [VJ01] using the LIPD dataset. The boosted cascade applied in our experiments follows the definitions of our previous work [LPNA11]. The results were ploted as a function of the number of stages, $Nc$, in order to verify the efficience of Corollary 5.1 in forecasting the optimal number of cascade stages, in SRM sense, i.e. the idea is to verify the contribution of (5.17) in the application of SRM principle to cascade classifiers.

The experimental results with SRM-cascade are summarized in four graphics, shown in Figures 6.7, 6.8, 6.9, and 6.10, with the following metrics: risk on testing dataset (R), BER, FP, and TP. These metrics were evaluated for an arbitrary TP of 0.98. Concerning the experiments on the training dataset, the aforementioned metrics are denoted in the graphics as: $R_{emp}$, $BER_{train}$, $TP_{emp}$, and $FP_{emp}$, while the results on the testing dataset are denoted as: $R_{test}$, $BER_{test}$, $TP_{test}$, and $FP_{test}$.

Figures 6.11 and 6.12 show the results of SRM-cascade and boosted cascade [VJ01] applied to LIPD dataset, respectively.

Regarding the training, the SRM-cascade presented its lowest value of empirical risk, $R_{emp} = 0.2592$, at $Nc = 17$ and its lowest value of BER, $BER_{train} = 0.2569$, at $Nc = 14$, while the boosted cascade presented its lowest value of empirical risk ,$R_{emp} = 0.3061$, at $Nc$=12 and the its lowest value of BER, $BER_{train} = 0.2569$ at $Nc$=12. Regarding the testing, the SRM-cascade presented its lowest value of risk, $R_{test} = 0.0327$, at $Nc = 17$, more specifically, for $FP_{test}$=0.0326 and $TP_{test}$=0.699. The upper bound on the expected risk of the entire ensemble, given by (5.17), did not decrease for $Nc > 12$, as can be seen in the red curve of Fig. 6.7, which was ploted by applying (5.17). Therefore, the SRM-based training adopts only 12 stages, despite the decreasing of the empirical risk along the following stages, as can be seen in the blue curve of Fig. 6.7. The SRM-cascade with 12 stages showed $R_{test} = 0.0331$, which is only 0.0004 larger than the risk of the cascade with 17 stages. Therefore, the SRM-based training avoided the, almost unusefull, computational effort related to the last 5 stages. Moreover, the SRM-cascade presented risk on the testing data slightly smaller than the risk of the boosted cascade, which was $R_{test} = 0.0337$.

The SRM-cascade also was the best approach in terms of BER, as can be seen by comparing Figs 6.8 and 6.12; namely, the SRM-cascade presented $BER_{test} = 0.0316$ at $Nc = 1$, while the boosted cascade presented $BER_{test} = 0.0360$, also at $Nc = 1$. Notice that, BER is not an usual metric for object detection aplications, since FP is the major concern in such kind of application. Nonetheless, if BER was assumed to be the stop criterion of the cascade training, the upper bound curve in Fig. 6.8, i.e. the red curve that was generated through the application of Corollary 5.3, correctly indicates one single stage ($Nc = 1$), while the training curve (blue curve in Fig. 6.8) is still decreasing, which indicates that the empirical risk is not a viable reference for the optimum number of stages.

The performances of both methods on the testing dataset were better than their performances on the training dataset, which were even close to the upper bound on the expected risk (the red curve). This apparently unexpected fact is due to the used data selection algorithm, which composes a training dataset with hard-negatives, i.e. non-pedestrian examples quite similar to pedestrian examples, since each one is a support vector (see Section 6.3.3).

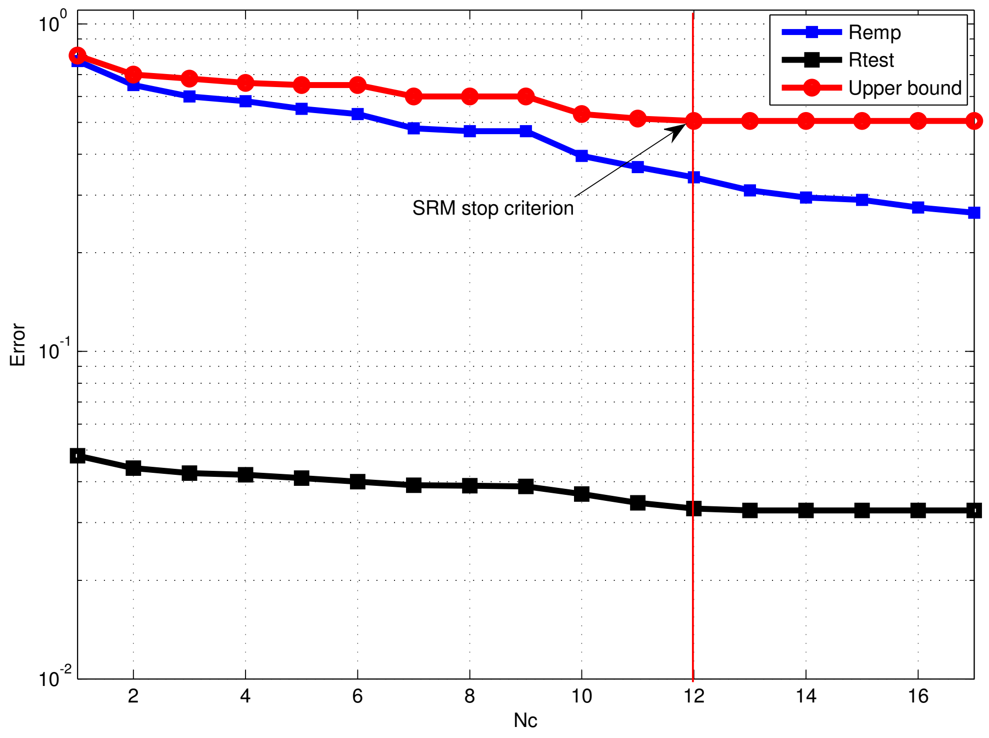


Figure 6.7: SRM-cascade applied to LIPD dataset: empirical risk (Remp), risk on the testing dataset (Rtest), and upper bound on the expected risk, as function of $Nc$.

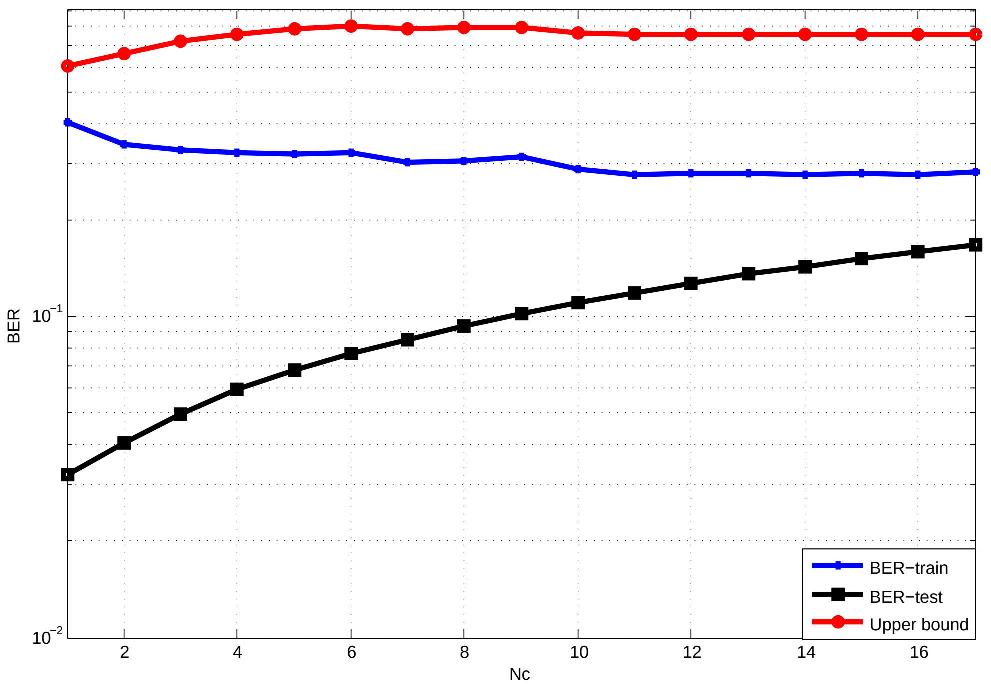


Figure 6.8: SRM-cascade applied to LIPD dataset: empirical BER, BER on the testing dataset, and upper bound on the expected BER, as function of $Nc$.

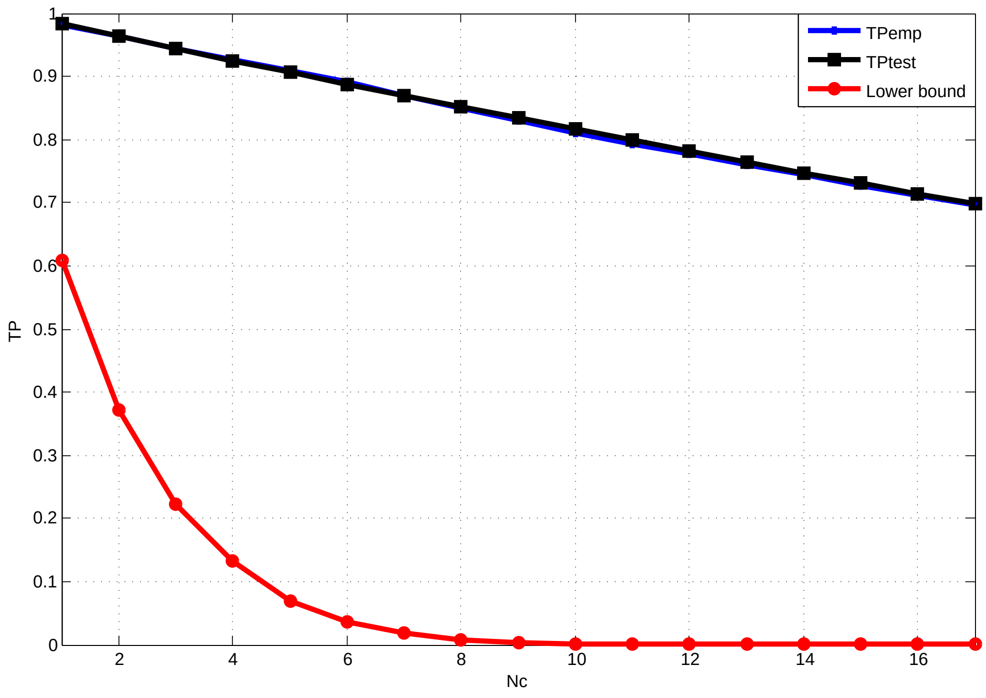


Figure 6.9: SRM-cascade applied to LIPD dataset: $TP_{emp}$, $TP_{test}$, and lower bound on the expected $TP$, as function of $Nc$.

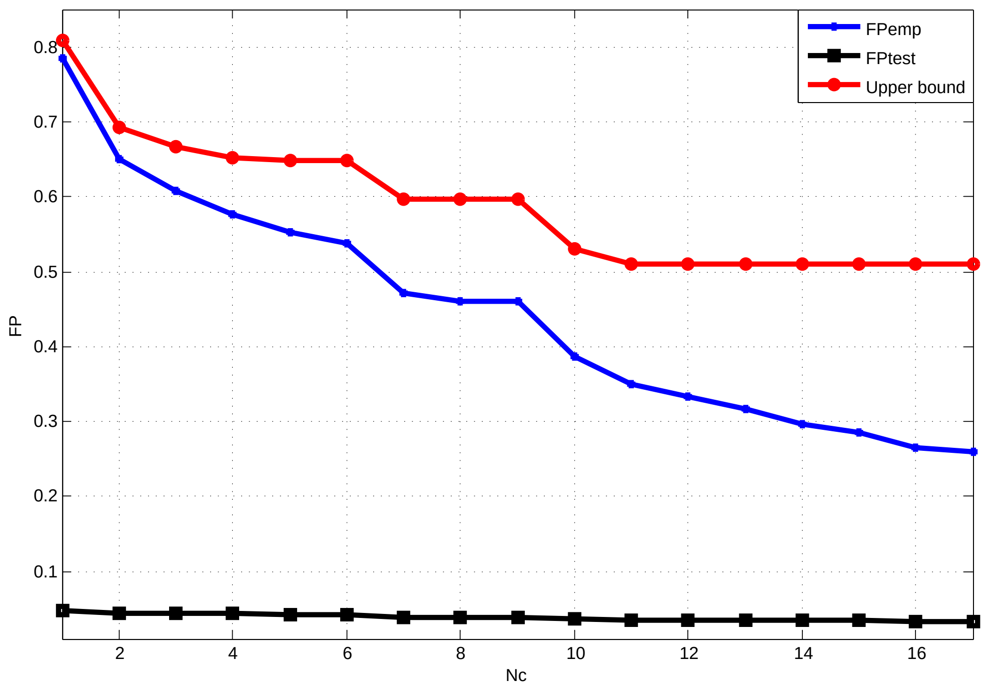


Figure 6.10: SRM-cascade applied to LIPD dataset: $FP_{emp}$, $FP_{test}$, and upper bound on the expected $FP$, as function of $Nc$.

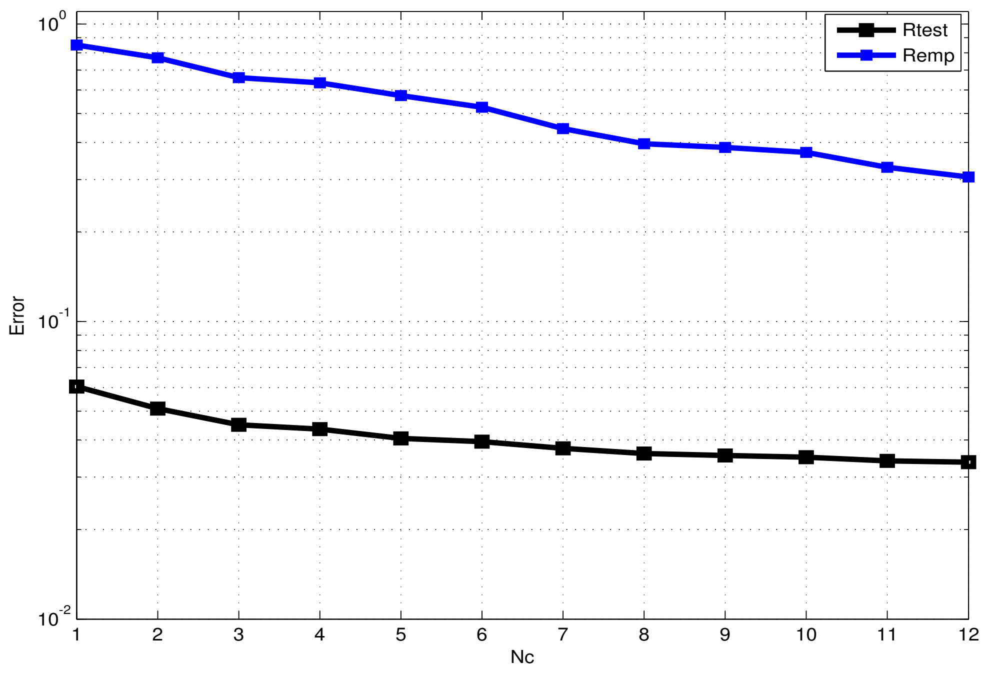


Figure 6.11: Boosted cascade applied to LIPD dataset: empirical risk (Remp), risk on the testing dataset (Rtest), and upper bound on the expected risk, as function of $Nc$.

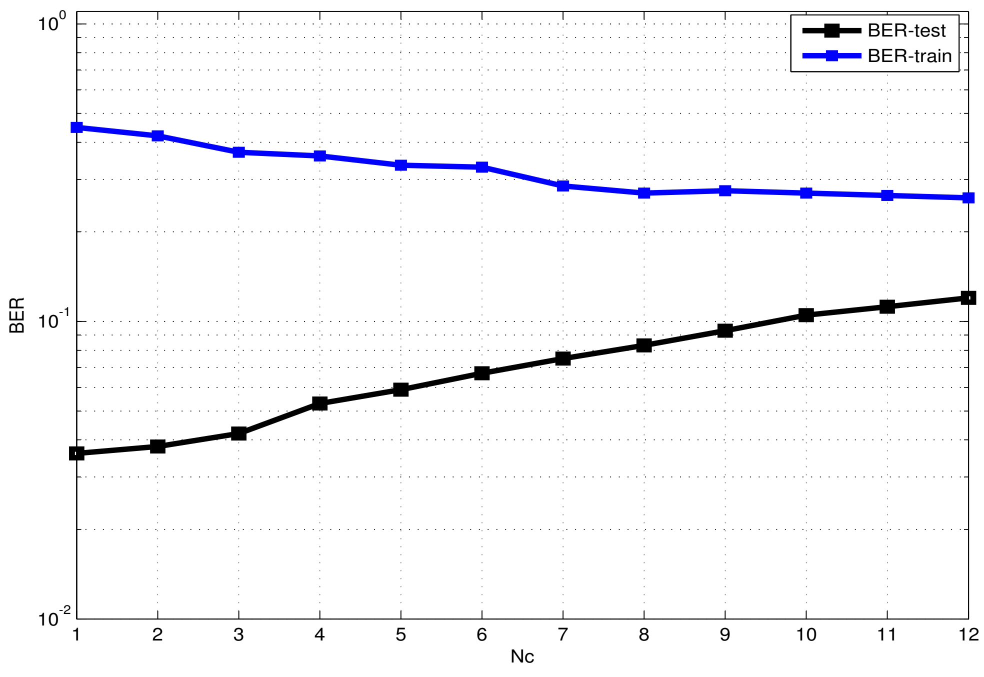


Figure 6.12: Boosted cascade applied to LIPD dataset: empirical BER, BER on the testing dataset, and upper bound on the expected BER, as function of $Nc$.

# Chapter 7

# Conclusions and discussion

THIS thesis proposes and analyzes different techniques to deal with two types of datasets: balanced and unbalanced. It was proposed the use of MLP in approaching balanced datasets and cascade classifier for unbalanced datasets.

Experiments on a real world benchmark dataset provide evidence of the effectiveness of the proposed MLP training methods regarding accuracy, AUC, and BER; especially when compared to the usual GDX. An MLP trained by the proposed methods is a suitable option for a faster non-linear classification, by avoiding the time-consuming calculation of the decision-function of nonlinear SVMs, which may hinder on-the-fly applications, as can be seen in Table 6.2 or in Section 4.2 of [EG09]. Moreover, the experimental results on the Daimler dataset indicate that an MLP trained by SVNN, TNN, and RNN can outperform the SVM-RBF.

The proposed MLP training methods have time and space complexities $O(N)$, while usual SVM training algorithms have $O(N^3)$ time and $O(N^2)$ space complexities, where $N$ is the training-dataset cardinality [TKC05]. However, despite the favorable time complexity, in practice all the proposed MLP training methods were time-consuming when applied to the Daimler dataset. However, specially in the case of MMGDX, which has low memory requirements, a second order gradient-based training method might be a suitable approach in optimizing the objective function (4.17), because such method may avoid local minima and decrease the CPU time.

The theoretical analysis developed in Section 4.3 led us to propose a new regularization method which aims at increasing the classification margin. This fact allowed the synthesis of two novel SVM-like training methods for NN, including a transductive training algorithm. The experiments indicated that, regarding the classification accuracy, both SVNN and TNN are quite similar; however, algorithms based on transductive learning, such as TNN, can better take advantage of the unlabeled data when few labeled data are available.

The theoretical analysis on the proposed redundant training method, RNN, was validated by the experiments on pedestrian classification, which showed the clear advantage of this method, which reached an impressive accuracy, specially taking into account that the MLP applied only HOG features. Moreover, such experimental result indicates that the variety of training methods for MLP proposed in this thesis is especially useful in combining MLPs into redundant architectures, since the diversity of behavior among the neural subsystems leads to larger independence in the distribution of the error among the neural subsystems, which improved the generalization capability of the RNN.

Regarding object detection, given that such type of application usually generates highly unbalanced datasets, it was adopted cascade classifiers, because such type of classifier ensemble can quickly process unbalanced data by sequential negative rejection. However, similarly to other classifier ensembles, cascade classifiers are likely to have high VC dimension, which may lead to over-fitting the training data. Therefore, this work applies the SRM principle aiming at improving the generalization capacity of cascade classifiers by defining the number of cascade stages and the number of features in each stage. To control the number of cascade stages it was required the derivation of a upper bound on the expected risk of the entire ensemble, which also yielded a new bound on BER[1] that can also be applied in the analysis of single classifiers, being specially useful in analyzing their performance under high unbalanced data.

The preliminary results on pedestrian detection, presented in the Chapter 6, indicate that the theoretical framework developed in Section 5.2.2 is useful in applying the SRM principle on cascade classifiers, i.e. in our experiments Corollary 5.1 gave the correct estimate of the optimal[2] number of stages, which enables the use of the entire training dataset, $S$, in the cascade training, since, differently from the boosted

[1]there are some works in digital signal processing which establish bounds on the bit error rate (BER), which is different from balanced error rate

[2]In SRM sense

cascade, the SRM-cascade does not require a validation dataset to evaluate its generalization capacity (aiming at avoiding overtraining). However, to enable a fair comparison of those methods, the SRM-cascade was trained with the same subset of the training dataset that was used in training the boosted cascade, i.e. without aggregate the validation data reserved for training the boosted cascade. In this sense, our theoretical framework also decreases the computational effort, which is usually high in training methods whose stop criterion is based on cross-validation.

The theoretical analysis developed in Section 5.2.3 highlights the trade-off between true positive rate and computational effort, indicating that the larger the adopted true positive rate for the cascade stages, the larger is the number of stages and, consequently, the computational effort. Moreover, Corollary 5.2 indicates that, in case of balanced datasets, a single classifier may be a better option than a cascade ensemble (see Fig. 5.3).

As future work we intend to apply other theoretical frameworks, such as Radamacher complexity [BM03], to derive a tighter upper bound on expected risk of the cascade ensemble. Moreover, we intend to repeat the experiments aggregating the validation data in the SRM-cascade training, aiming at a better detection rate.

Despite the limited scope of the case study adopted by this thesis, the empirical analysis reported here is in accordance with the experimental studies performed on several benchmark datasets and reported in our works [LN10], [LN11], and [LPNR11].

# Bibliography

[Abe05] S. Abe, *Support vector machines for pattern classification (advances in pattern recognition)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[AGHR05] S. Agarwal, T. Graepel, R. Herbrich, and D. Roth, *A large deviation bound for the area under the roc curve*, In Advances in Neural Information Processing Systems, vol. 17, MIT Press, 2005, pp. 9–16.

[AI01] S. Abe and T. Inoue, *Fast training of support vector machines by extracting boundary data*, Artificial Neural Networks, ICANN 2001 (2001), 308–313.

[AMM$^+$97] S. Amari, N. Murata, K.R. Muller, M. Finke, and H.H. Yang, *Asymptotic statistical theory of overtraining and cross-validation*, Neural Networks, IEEE Transactions on **8** (1997), no. 5, 985–996.

[Bar98] P.L. Bartlett, *The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network*, Information Theory, IEEE Transactions on **44** (1998), no. 2, 525–536.

[BdPB00] M. Barros, A. de Padua, and J.P. Braga, *SVM-KM: speeding SVMs learning with a priori cluster selection and k-means*, Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on, IEEE, 2000, pp. 162–167.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth, *Learnability and the vapnik-chervonenkis dimension*, Journal of the Association for Computing Machinery **36** (1989), no. 4, 929–965.

[BGV92] B.E. Boser, I.M. Guyon, and V.N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.

[BH89] E.B. Baum and D. Haussler, *What size net gives valid generalization?*, Neural computation **1** (1989), no. 1, 151–160.

[Bis96] C.M. Bishop, *Neural networks for pattern recognition*, 1 ed., Oxford University Press, USA, 1996.

[BM03] P.L. Bartlett and S. Mendelson, *Rademacher and gaussian complexities: Risk bounds and structural results*, The Journal of Machine Learning Research **3** (2003), 463–482.

[BP02] L. Bruzzone and D.F. Prieto, *A partially unsupervised cascade classifier for the analysis of multitemporal remote-sensing images*, Pattern Recognition Letters **23** (2002), no. 9, 1063 – 1071.

[Bur98] C.J.C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery **2** (1998), 121–167.

[CAL94] D. Cohn, L. Atlas, and R. Ladner, *Improving generalization with active learning*, Machine Learning **15** (1994), no. 2, 201–221.

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*, vol. 2, MIT press Cambridge, MA, 2006.

[CV95] C. Cortes and V.N. Vapnik, *Support-vector networks*, Machine learning **20** (1995), no. 3, 273–297.

[Dal06] N. Dalal, *Finding people in images and videos*, PhD thesis of Institut National Polytechnique de Grenoble (2006).

[DAR03] DARPA, *The urban and grand challenges. [online]*, http://www.darpa.mil/grandchallenge/, (accessed: 2010), 2003.

[DB98] H. Demuth and M. Beale, *Neural network toolbox user's guide*, The MathWorks Inc (1998).

[DFR07] B. Douillard, D. Fox, and F. Ramos, *A spatio-temporal probabilistic model for multi-sensor object recognition*, Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference on, 29 2007-Nov. 2 2007, pp. 2402–2408.

[DGC07] A.K. Deb, M. Gopal, and S. Chandra, *SVM-based tree-type neural networks as a critic in adaptive critic designs for control*, Neural Networks, IEEE Transactions on **18** (2007), no. 4, 1016 –1030.

[DT05] N. Dalal and B. Triggs, *Histograms of oriented gradients for human detection*, Computer Vision and Pattern Recognition, CVPR. IEEE Conference on (Washington, DC, USA), IEEE Computer Society, 2005, pp. 886–893.

[DWSP09] P. Dollar, C. Wojek, B. Schiele, and P. Perona, *Pedestrian detection: A benchmark*, 2009, pp. 304–311.

[EG09] M. Enzweiler and D.M. Gavrila, *Monocular pedestrian detection: Survey and experiments*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009), no. 12, 2179 –2195.

[ELR06] ELROB, *The european robot trial. [online]*, http://www.elrob.org/, (accessed: 2011), 2006.

[FC07] F. Fayad and V. Cherfaoui, *Tracking objects using a laser scanner in driving situation based on modeling target shape*, Intelligent Vehicles Symposium, IVS. IEEE, 0-0 2007, pp. 44–49.

[FS95] Y. Freund and R. Schapire, *A desicion-theoretic generalization of on-line learning and an application to boosting*, Computational Learning Theory (Paul Vitanyi, ed.), Lecture Notes in Computer Science, vol. 904, Springer Berlin Heidelberg, 1995, pp. 23–37.

[GB00] J. Gama and P. Brazdil, *Cascade generalization*, Machine Learning **41** (2000), 315–343.

[GBD92] S. Geman, E. Bienenstock, and R. Doursat, *Neural networks and the bias/variance dilemma*, Neural computation **4** (1992), no. 1, 1–58.

[GCB$^+$04] H.P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V.N. Vapnik, *Parallel support vector machines: The cascade SVM*, NIPS, 2004.

[Gla87] R.B. Glassman, *An hypothesis about redundancy and reliability in the brains of higher species: Analogies with genes, internal organs, and engineering systems*, Neuroscience and Biobehavioral Reviews **11** (1987), no. 3, 275 – 285.

[GLSG10] D. Gerónimo, A.M. López, A.D. Sappa, and T. Graf, *Survey of pedestrian detection for advanced driver assistance systems*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **32** (2010), no. 7, 1239–1258.

[GT07] T. Gandhi and M.M. Trivedi, *Pedestrian protection systems: issues, survey, and challenges*, Intelligent Transportation Systems, IEEE Transactions on **8** (2007), no. 3, 413–430.

[HCH$^+$04] Z. Huang, H. Chen, C.J. Hsu, W.H. Chen, and S. Wu, *Credit rating analysis with support vector machines and neural networks: a market comparative study*, Decision support systems **37** (2004), no. 4, 543–558.

[HCR$^+$07] Jae Pil Hwang, Seung Eun Cho, Kyung Jin Ryu, Seungkeun Park, and Euntai Kim, *Multi-classifier based lidar and camera fusion*, Intelligent Transportation Systems Conference, ITSC. IEEE International Conference on, 30 2007-Oct. 3 2007, pp. 467–472.

[Hoe63] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), no. 301, 13–30.

[HSW90] K. Hornik, M. Stinchcombe, and H. White, *Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks*, Neural networks **3** (1990), no. 5, 551–560.

[IKW08] R. Ilin, R. Kozma, and P.J. Werbos, *Beyond feedforward models trained by backpropagation: A practical training tool for a more efficient universal approximator*, Neural Networks, IEEE Transactions on **19** (2008), no. 6, 929–937.

[IP90] Y. Izui and A. Pentland, *Analysis of neural networks with redundancy*, Neural Computation **2** (1990), no. 2, 226–238.

[IYM03] M.M. Islam, X. Yao, and K. Murase, *A constructive algorithm for training cooperative neural network ensembles*, Neural Networks, IEEE Transactions on **14** (2003), no. 4, 820 – 834.

[Jin04] Y. Jin, *Neural network regularization and ensembling using multi-objective evolutionary algorithms*, In: Congress on Evolutionary Computation (CEC'04), IEEE, IEEE Press, 2004, pp. 1–8.

[KC06] P. Kang and S. Cho, *EUS SVMs: Ensemble of under-sampled SVMs for data imbalance problems*, Neural Information Processing, Lecture Notes in Computer Science, vol. 4232, Springer Berlin / Heidelberg, 2006, pp. 837–846.

[KD95] E.B. Kong and T.G. Dietterich, *Error-correcting output coding corrects bias and variance*, In Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 313 – 321.

[KDMC83] M.P. Kovac, W.J. Davis, E.M. Matera, and R.P. Croll, *Organization of synaptic inputs to paracerebral feeding command interneurons of pleurobranchaea californica. I. excitatory inputs*, Journal of Neurophysiology **49** (1983), no. 6, 1517–1538.

[KL08] C.T. Kim and J.J. Lee, *Training two-layered feedforward networks with variable projection method*, IEEE Transactions on Neural Networks **19** (2008), no. 2, 371–375 (English).

[KM95] M. Karpinski and A. Macintyre, *Polynomial bounds for VC dimension of sigmoidal neural networks*, in proc. 27th ACM Symposium on Theory of Computing, 1995.

[KS97] P. Koiran and E.D. Sontag, *Neural networks with quadratic VC dimension*, Journal of Computer and System Sciences **54** (1997), no. 1, 190–198.

[Kun02] L.I. Kuncheva, *Switching between selection and fusion in combining classifiers: an experiment*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **32** (2002), no. 2, 146 –156.

[KW96] R. Kohavi and D.H. Wolpert, *Bias plus variance decomposition for zero-one loss functions*, Machine learning: proceedings of the Thirteenth International Conference (ICML'96), Morgan Kaufmann Pub, 1996, p. 275.

[KWD03] L. I. Kuncheva, C. J. Whitaker, and R. P. W. Duin, *Limits on the majority vote accuracy in classifier fusion*, Pattern Analysis and Applications **6** (2003), 22–31.

[LDGN09] O. Ludwig, D. Delgado, V. Goncalves, and U. Nunes, *Trainable classifier-fusion schemes: An application to pedestrian detection*, oct. 2009, pp. 1 –6.

[LGL06] O. Ludwig, P.C. Gonzalez, and A.C. Lima, *Optimization of ANN applied to non-linear system identification*, Proceedings of the 25th IASTED international conference on Modeling, indentification, and control (Anaheim, CA, USA), ACTA Press, 2006, pp. 402–407.

[Liu04] C. Liu, *Gabor-based kernel PCA with fractional power polynomial models for face recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), 572–581.

[LM01] Y.J. Lee and O.L. Mangasarian, *RSVM: Reduced support vector machines*, Proceedings of the First SIAM International Conference on Data Mining, 2001, pp. 00–07.

[LN10] O. Ludwig and U. Nunes, *Novel maximum-margin training algorithms for supervised neural networks*, Neural Networks, IEEE Transactions on **21** (2010), no. 6, 972 –984.

[LN11] ______, *Transductive neural networks*, Submitted to IEEE Transactions on Neural Networks (2011).

[LNA$^+$09] O. Ludwig, U. Nunes, R. Araujo, L. Schnitman, and H.A. Lepikson, *Applications of information theory, genetic algorithms, and neural models to predict oil flow*, Communications in Nonlinear Science and Numerical Simulation **14** (2009), no. 7, 2870 – 2885.

[LPNA11] O. Ludwig, C. Premebida, U. Nunes, and R. Araújo, *Evaluation of boosting-SVM and SRM-SVM cascade classifiers in laser and vision-based pedestrian detection*, IEEE International Conference on Intelligent Transportation Systems, ITSC'11, Oct. 2011.

[LPNR11] O. Ludwig, C. Premebida, U. Nunes, and B. Ribeiro, *Theoretical analysis on cascade classifiers with application in pedestrian detection*, Submitted to Pattern Recognition (2011).

[LW94] N. Littlestone and M.K. Warmuth, *The weighted majority algorithm*, Information and Computation **108** (1994), 212–261.

[Mat92] K. Matsuoka, *Noise injection into inputs in back-propagation learning*, Systems, Man and Cybernetics, IEEE Transactions on **22** (1992), no. 3, 436–440.

[MG06] S. Munder and D.M. Gavrila, *An experimental study on pedestrian classification*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **28** (2006), no. 11, 1863 –1868.

[Mor78] J. More, *The levenberg-marquardt algorithm: implementation and theory*, Numerical analysis (1978), 105–116.

[MRW$^+$99] S. Mika, G. Rotsch, J. Weston, B. Scholkopf, and K.R. Muller, *Fisher discriminant analysis with kernels*, vol. IX, pp. 41–48, IEEE, 1999.

[MSRD06] M. Mahlisch, R. Schweiger, W. Ritter, and K. Dietmayer, *Sensorfusion using spatio-temporal aligned video and lidar for improved vehicle detection*, Intelligent Vehicles Symposium, IVS. IEEE, 0-0 2006, pp. 424–429.

[NW90] D. Nguyen and B. Widrow, *Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*, International Joint Conference on Neural Networks, IJCNN'90, IEEE, 1990, pp. 21–26.

[NZ06] R. Neal and J. Zhang, *High dimensional classification with bayesian neural networks and dirichlet diffusion trees*, Feature Extraction, Studies in Fuzziness and Soft Computing, vol. 207, Springer Berlin / Heidelberg, 2006, pp. 265–296.

[OM07] G. Ou and Y. Murphey, *Multi-class pattern classification using neural networks*, Pattern Recognition **40** (2007), no. 1, 4–18.

[PLD05] H. Peng, F. Long, and C. Ding, *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **27** (2005), no. 8, 1226–1238.

[PLN09a] C. Premebida, O. Ludwig, and U. Nunes, *Exploiting lidar-based features on pedestrian detection in urban scenarios*, Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on, IEEE, 2009, pp. 1–6.

[PLN09b] ______, *Lidar and vision-based pedestrian detection system*, Journal of Field Robotics **26** (2009), no. 9, 696–711.

[Pow77] MJD Powell, *Restart procedures for the conjugate gradient method*, Mathematical programming **12** (1977), no. 1, 241–254.

[Sam04] B. Samanta, *Gear fault detection using artificial neural networks and support vector machines with genetic algorithms*, Mechanical Systems and Signal Processing **18** (2004), no. 3, 625–644.

[Sau72] N. Sauer, *On the density of families of sets*, Journal of Combinatorial Theory, Series A **13** (1972), no. 1, 145 – 147.

[SC03] H. Shin and S. Cho, *Fast pattern selection for support vector classifiers*, Advances in Knowledge Discovery and Data Mining (2003), 567–567.

[SDJ99] R.S. Sexton, R.E. Dorsey, and J.D. Johnson, *Optimization of neural networks: a comparative analysis of the genetic algorithm and simulated annealing*, European Journal of Operational Research **114** (1999), no. 3, 589–601.

[SFBL98] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, *Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods*, The Annals of Statistics **26** (1998), no. 5, 1651–1686.

[SK07] A. Schmidt and A. Kasinski, *The performance of the haar cascade classifiers applied to the face and eyes detection*, Computer Recognition Systems 2, Advances in Intelligent and Soft Computing, vol. 45, Springer Berlin / Heidelberg, 2007, pp. 816–823.

[SL86] B.L. Strehler and R. Lestienne, *Evidence on precise time-coded symbols and memory of patterns in monkey cortical neuronal spike trains*, Proceedings of the National Academy of Sciences **83** (1986), no. 24, 9812–9816.

[SS08] L. Spinello and R. Siegwart, *Human detection using multimodal and multidimensional features*, Robotics and Automation, ICRA. IEEE International Conference on, May 2008, pp. 3264–3269.

[SSO06] M. Szarvas, U. Sakai, and J. Ogata, *Real-time pedestrian detection using lidar and convolutional neural networks*, Intelligent Vehicles Symposium, IVS. IEEE, 0-0 2006, pp. 213–218.

[TG99] N.K. Treadgold and T.D. Gedeon, *Exploring constructive cascade networks*, Neural Networks, IEEE Transactions on **10** (1999), no. 6, 1335 –1350.

[TKC05] I.W. Tsang, J.T. Kwok, and P.M. Cheung, *Core vector machines : Fast SVM training on very large data sets*, Journal of Machine Learning Research **6** (2005), no. 1, 363–392.

[TPM06] O. Tuzel, F. Porikli, and P. Meer, *Region covariance: A fast descriptor for detection and classification*, In Proc. 9th European Conf. on Computer Vision, 2006, pp. 589–600.

[TPM07] O. Tuzel, F. Porikli, and P. Meer, *Human detection via classification on Riemannian manifolds*, Computer Vision and Pattern Recognition, CVPR. IEEE Conference on, June 2007, pp. 1–8.

[Val84] L.G. Valiant, *A theory of the learnable*, Communications of the ACM **27** (1984), no. 11, 1134–1142.

[Vap82] V.N. Vapnik, *Estimation of dependences based on empirical data*, Springer Verlag, 1982.

[Vap98] ______, *Statistical learning theory*, John Wiley, 1998.

[Vap99] ______, *An overview of statistical learning theory*, Neural Networks, IEEE Transactions on **10** (1999), no. 5, 988–999.

[VC71] V.N. Vapnik and A.Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability and its Applications **16** (1971), 264.

[VJ01] P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, vol. 1, 2001, pp. 511–518.

[Wil95] P.M. Williams, *Bayesian regularization and pruning using a laplace prior*, Neural Computation **7** (1995), no. 1, 117–143.

[YD98] S. Young and T. Downs, *Carve-a constructive algorithm for real-valued examples*, IEEE Transactions on Neural Networks **9** (1998), 1180–1190.

[YNW$^+$07] D.S. Yeung, W.W.Y. Ng, D. Wang, E.C.C. Tsang, and X.Z. Wang, *Localized generalization error model and its application to architecture selection for radial basis function neural network*, Neural Networks, IEEE Transactions on **18** (2007), no. 5, 1294–1305.

[ZBS07] P. Zhang, T.D. Bui, and C.Y. Suen, *A novel cascade ensemble classifier system with a high recognition performance on handwritten digits*, Pattern Recognition **40** (2007), no. 12, 3415 – 3429.

[ZP04] Q. Zhang and R. Pless, *Extrinsic calibration of a camera and laser range finder (improves camera calibration)*, Intelligent Robots and Systems, IROS. IEEE/RSJ International Conference on, vol. 3, Sept.-2 Oct. 2004, pp. 2301–2306 vol.3.