Title
Multiclass Boosting for Fast Multiclass Object Detection /

Author
Saberian, Mohammad

Publication Date
2014

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Multiclass Boosting for Fast Multiclass Object Detection

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Electrical Engineering (Signal and Image Processing)

by

Mohammad Saberian

Committee in charge:

Professor Nuno Vasconcelos, Chair
Professor Serge Belongie
Professor Kenneth Kreutz-Delgado
Professor David Kriegman
Professor Bhaskar D. Rao
Professor Lawrence Saul

2014

The dissertation of Mohammad Saberian is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2014

DEDICATION

I dedicate this dissertation to my wife, Sara.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am fortunate to have benefited from the help of many people during my doctoral work, all of whom I cannot possibly acknowledge in a few paragraphs.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Nuno Vasconcelos. His vast experience, meticulous attitude toward technicalities, as well as his confidence in his students are the kinds of assets any graduate student would be very lucky to have. His broad knowledge, logic way of thinking, and deep insights to the fundamental problems of computer vision and machine learning are, without a doubt, reflected in this dissertation. I am also grateful for having an exceptional doctoral committee, and wish to thank Professor Serge Belongie, Professor David Kriegman, Professor Kenneth Kreutz-Delgado, Professor Bhaskar D. Rao and Professor Lawrence Saul for their support, valuable input, and teaching me a great deal about computer vision and machine learning.

My research was supported in part by the National Science Foundation (NSF) awards IIS-1208522, CCF-0830535, ATM-0941760 (PI: Nuno Vasconcelos) and the Korean Ministry of Trade, Industry & Energy, grant no. 10041126. I would like to thank everyone involved for providing me with the opportunity to work in this area.

I would like to thank all my colleagues at UCSD, Dr. Dashan Gao, Dr. Antoni Chan, Dr. Sunhyoung Han, Dr. Nikhil Rasiwasia, Dr. Vijay Mahadevan, Dr. Hamed Masnadi-Shirazi, Jose Maria Costa Pereira, Kritika Muralidharan, Weixin Li, Mandar Dixit, Song Lu, Can Xu, Panqu Wang, Perry Naughton, and Oscar Beijbom for their support, friendship, assistance and their collaboration.

I would like also to sincerely thank my parents, specially my parents Alireza and Mozhgan, for their endless love and support toward me. Their kindness and help will always be a source of hope and energy for me, and I am incapable to thank them for all they have done for me. And a special thanks to my mother and father in-law for their love and support. I would also like to thank my sister

Nafiseh, my brother Sadegh, my sister-in-law Sanaz and brother-in-law Hossein for their support.

Last and most importantly, I cannot thank enough my dear wife, Sara, for her continuous love and support, for giving up her career in Iran and coming to this country with me without hesitation, for standing beside and encouraging me whenever I feel frustrated. It is hard to imagine that someone could ever be luckier than I was when I chose her to be my wife.

The text of Chapter II, in part, is based on the material as it appears in: Mohammad Saberian, Hamed Masnadi-Shirazi and Nuno Vasconcelos. Taylor-Boost: First and Second-order Boosting Algorithms with Explicit Margin Control. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. The dissertation author was a primary researcher and an author of the cited material. The author would like to thank Hamed Masnadi-Shirazi for helpful discussions and comments.

The text of Chapter III, in full, is based on the material as it appears in: Mohammad Saberian and Nuno Vasconcelos. Boosting Algorithm for Learning Detector Cascade, *to appear in Journal of Machine Learning Research (JMLR)* and Mohammad Saberian and Nuno Vasconcelos. Boosting Classifer Cascades. *Proceedings of Neural Information Processing Systems (NIPS)*, 2010. The dissertation author was a primary researcher and an author of the cited material.

The text of Chapter IV, in full, is based on the material as it appears in: Mohammad Saberian and Nuno Vasconcelos. Multiclass Boosting: Theory and Algorithms. *Proceedings of Neural Information Processing Systems (NIPS)*, 2011. and Mohammad Saberian and Nuno Vasconcelos. Multiclass Boosting: Theory and Algorithms. *to be submitted to Machine Learning Research (JMLR)*. The dissertation author was a primary researcher and an author of the cited material. The author would like to thank Hamed Masnadi-Shirazi for helpful discussions and comments.

The text of Chapter V, in full, is based on the material as it appears

in:Mohammad Saberian and Nuno Vasconcelos. Learning Multi-Resolution Cascades for Fast Multiclass Object Detection. *to be submitted to Neural Information Processing Systems (NIPS)*, 2014. The dissertation author was a primary researcher and an author of the cited material.

VITA

2003-2008	Bachelor of Science, Electrical Engineering, Sharif University of Technology, Iran
2003-2008	Bachelor of Science, Computer Science, Sharif University of Technology, Iran
2008–2010	Master of Science Electrical Engineering (Signal and Image Processing), University of California at San Diego
2008–2014	Research Assistant Statistical and Visual Computing Laboratory Department of Electrical and Computer Engineering University of California at San Diego
2014	Doctor of Philosophy Electrical Engineering (Signal and Image Processing), University of California at San Diego

PUBLICATIONS

Mohammad Saberian and Nuno Vasconcelos. Boosting Algorithm for Learning Detector Cascade, *to appear in Journal of Machine Learning Research (JMLR)*

Mohammad Saberian and Nuno Vasconcelos. Learning Optimal Embedded Cascades. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34(10), pp. 2005-2018, October 2012.

Mohammad Saberian and Nuno Vasconcelos. Boosting Algorithms for Simultaneous Feature Extraction and Selection. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Mohammad Saberian and Nuno Vasconcelos. Multiclass Boosting: Theory and Algorithms. *Proceedings of Neural Information Processing Systems (NIPS)*, 2011.

Mohammad Saberian, Hamed Masnadi-Shirazi and Nuno Vasconcelos. TaylorBoost: First and Second-order Boosting Algorithms with Explicit Margin Control. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Mohammad Saberian and Nuno Vasconcelos. Boosting Classifer Cascades. *Proceedings of Neural Information Processing Systems (NIPS)*, 2010.

ABSTRACT OF THE DISSERTATION

Multiclass Boosting for Fast Multiclass Object Detection

by

Mohammad Saberian

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California, San Diego, 2014

Professor Nuno Vasconcelos, Chair

In this dissertation the problem of designing a fast multiclass object detector based on cascade architecture is considered. A classifier cascade is a sequence of simple to complex sub-classifiers where each stage either rejects the input or pass it to the next stage. Since most of the non-target inputs get rejected with the simple sub-classifiers in the early stages of the cascade, the overall classification will be fast.

Since cascade sub-classifier are usually trained with Boosting algorithms, the dissertation starts with proposing TaylorBoost which explains Boosting algorithms as iterative descent algorithms for minimizing Taylor expansion of risk of classification in function space. In the rest of this dissertation TaylorBoost is used to derive appropriate Boosting algorithms based on the requirements of the problems.

The main challenge in designing a classifier cascade is to tune speed-accuracy trade-off, e.g. more complex early stages in the cascade may increase accuracy but degrades speed of classification significantly. To address this issue, this dissertation proposes a new Boosting algorithm, FCBoost, for designing a classifier cascade by minimizing a a Lagrangian risk that jointly accounts for classification accuracy and speed of classification.

While FCBoost enables designing cascade detectors for single class of objects, designing detectors for multiple objects is still problematic. This is because each of the object detectors has to be trained independently which results in many redundant computational complexity. To address this issue, the dissertation next proposes a new multiclass Boosting framework, MCBoost. Combining FCBoost and MCBoost, makes it possible to learn detector cascade for detecting multiple objects. The remaining challenge is that in a multiclass cascade, early stages should implement binary target vs. non-target detectors of high simplicity and false-positive rate, and late stages should be multiclass classifiers of high accuracy and complexity to distinguish between target classes. The dissertation proposes a method to manipulate cost of classification in MCBoost based on cascade false-positive rate to address this issue. Experiments on the problems of multi-view car detection and simultaneous detection of multiple traffic signs show that the proposed detector is faster and more accurate than previous approaches.

Chapter I

Introduction

I.A Real-time multiclass object detector

The problem of simultaneous real-time detection of multiple classes of objects is a challenging problem and subsumes various important applications in computer vision. These range from the literal detection of many objects (e.g. an automotive vision system that must detect cars, pedestrians, traffic signs), to the detection of objects at multiple semantic resolutions (e.g. a camera that can both detect faces and recognize certain users), to the detection of different aspects of the same object (e.g. by defining classes as different poses).

In most of the object detectors a sliding window is scanned throughout an image, generating hundreds of thousands of image sub-windows. A classifier must then decide if each sub-window contains certain target objects, ideally at video frame-rates, i.e. less than a micro second per window. The major challenge is the design of a very fast and accurate classifier to use in the object detector. A popular classification architecture for this task is the detector cascade of Figure I.1-a [91]. This detector is implemented as a sequence of simple to complex classification stages, each of which can either reject the example x to classify or pass it to the next stage. An example that reaches the end of the cascade is classified as a target. Since targets (or similar patterns) only occupy a very small portion of the image sub-windows, most examples are rejected in the early cascade stages, by classifiers of very limited computation. In result, the average computation per image is small, and the cascaded detector is very fast. This architecture has shown outstanding results for the tasks of face, pedestrian and car detection [92, 103, 69].

While the design of cascades for real-time detection of a single object class has been the subject of extensive research [92, 103, 60, 81, 8, 64, 69], the simultaneous detection of multiple objects has received much less attention. In fact, most existing solutions simply decompose this problem into several binary (single class) detection sub-problems. They can be grouped as follows.

Parallel cascades [96]: these methods learn a cascaded detector per

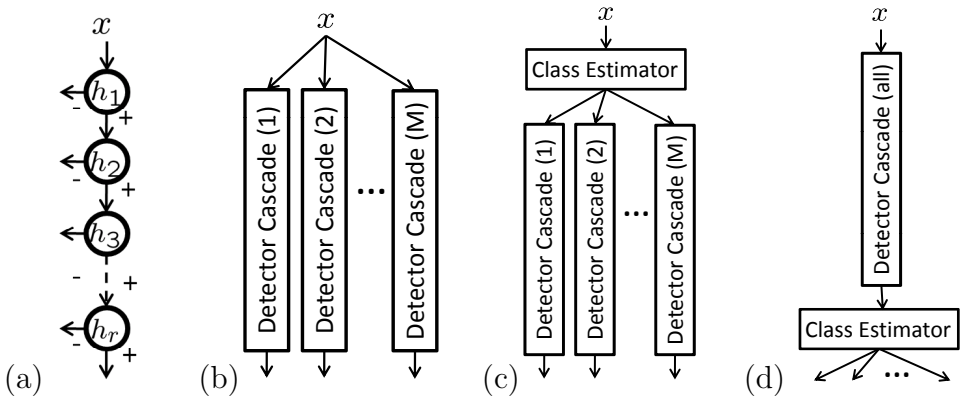


Figure I.1: a) detector cascade [91], b) parallel cascade [96], c) parallel cascade with pre-estimator [90] and d) all-class cascade with post-estimator.

object class (e.g. view), as shown in Figure I.1-b, and rely on some post-processing to combine their decisions. This has two limitations. The first is the well known sub-optimality of one-vs.-all multiclass classification, since scores of independently trained detectors are not necessarily comparable [54]. Second, because there is no sharing of features across the detectors, the overall classifier performs many redundant computations and tends to be very slow.

Parallel cascades with pre-estimator [90]: The complexity of the parallel architecture can be reduced by first making a rough guess of the target class and running only one of the binary detectors, as illustrated in Figure I.1-c. While, for some applications (e.g. where classes are object poses), it is possible to obtain a reasonable pre-estimate of the target class, pre-estimation errors are difficult to undo. Hence, this classifier must be fairly accurate. This is difficult to guarantee, in particular because it must also have reduced computation. In [36], authors proposed a variant of this method, where multiple detectors are run after the pre-estimate. This improves accuracy but increases complexity.

All-class cascade with post-estimator: In this architecture, all target classes are first grouped into an abstract class of *positive patches*. A detector cascade is then trained to distinguish these patches from everything else. A patch identified as positive is finally fed to a multiclass classifier, for assignment to one

of the target classes. This architecture is shown in Figure I.1-d. In comparison to parallel cascades, this architecture has the advantage of sharing features across all classes, eliminating redundant computation. When compared to the parallel cascade with pre-estimator, it has the advantage that the complexity of its class estimator has little weight in the overall computation, since it only processes a small percentage of the examples. This allows the use of very accurate/complex estimators. The main limitation is that the design of a cascade that detects all positive patches can be problematic. Due to the large variability of the examples in this abstract class, this can be difficult to do accurately. For example, Viola and Jones declared this strategy hopelessly inaccurate [90].

In this dissertation, we address the limitations of all previous architectures, by proposing a true multiclass method for cascade learning. The proposed detector has the structure of Figure I.1-a, but implements each cascade stage with a multiclass classifier.

I.B Contributions of the dissertation

Designing a fast and accurate multiclass object detector requires addressing several challenges such as 1) how to design a classier, 2) how to design a classifier under complexity/time constraints, or 3) how to design multiclass classifiers with minimal complexity. Each chapter of this dissertation is focused on one of these sub-problems. In the last chapter, we will put together our solutions for these sub-problems and propose a multiclass object detector. In more details, the main contributions of the dissertation are as follows.

I.B.1 TaylorBoost

Most of the classifiers used for object detection are based on Boosting [26]. In Chapter II of this dissertation, the differences between Boosting and optimization methods in function space is discussed and a new family of Boosting

algorithms, denoted *TaylorBoost,* is proposed. TaylorBoost supports any combination of loss function and first or second order optimization, and includes classical algorithms such as AdaBoost[26], AnyBoost[53], or LogitBoost [27] as special cases. TaylorBoost is then use for designing new Boosting algorithms in the rest of this dissertation.

I.B.2 A family of Boosting algorithm for designing detector cascade

Designing a detector cascade for real-time object detection [91] is very complicated, requires lots of trial-and-errors and often results in sub-optimal detectors. In Chapter III of this dissertation, the problem of learning optimal classifier cascade is considered. A new cascade Boosting algorithm, *fast cascade Boosting* (FCBoost), is proposed. FCBoost is shown to have a number of interesting properties, namely that it 1) minimizes a Lagrangian risk that jointly accounts for classification accuracy and speed, 2) generalizes AdaBoost, 3) can be made cost-sensitive to support the design of high detection rate cascades, and 4) is compatible with many predictor structures suitable for sequential decision making. It is shown that a rich family of such structures can be derived recursively from cascade predictors of two stages, denoted *cascade generators.* Generators are then proposed for two new cascade families, *last-stage* and *multiplicative* cascades, that generalize the two most popular cascade architectures in the literature. The concept of *neutral predictors* is finally introduced, enabling FCBoost to automatically determine the cascade configuration, i.e. number of stages and number of weak learners per stage, for the learned cascades. Experiments on face and pedestrian detection show that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

I.B.3 Theory and algorithm of multiclass Boosting

One of the main challenges for designing a multiclass object detector is designing an accurate and efficient multiclass object detector. Since Boosting is

the dominant learning method in object detection, in Chapter IV the problem of multiclass Boosting is considered. A new framework, based on multi-dimensional codewords and predictors is introduced. The optimal set of codewords is derived, and a margin enforcing loss proposed. The resulting risk is minimized by gradient descent on a multidimensional functional space. Two algorithms are proposed: 1) CD-MCBoost, based on coordinate descent, updates one predictor component at a time, 2) GD-MCBoost, based on gradient descent, updates all components jointly. The algorithms differ in the weak learners that they support but are both shown to be Bayes consistent and margin enforcing. They also reduce to AdaBoost when there are only two classes. Experiments show that both methods outperform previous multiclass Boosting approaches in a number of data sets.

I.B.4 Multi-Resolution detector cascades

In theory, by combining FCBoost and MCBoost algorithms it is possible to design a multiclass detector cascade. In practice, however, a multiclass detector cascade has to have a behavior denoted *multi-resolution*. In Chapter IV of this dissertation we introduce this behavior and propose a solution to guarantee it. In particular, we argue that in a multiclass cascade, 1) early stages should implement binary target vs. non-target detectors of high simplicity and false-positive rate, 2) late stages should be multiclass classifiers of high accuracy and complexity to distinguish between target classes, 3) middle stages can have intermediate numbers of classes, determined based on structure of the data. We show that learning such detector is possible using cost-sensitive Boosting. By manipulating the costs of the associated Boosting risk, it is possible to emphasize 1) discrimination between the target classes and the negative class, or 2) discrimination among the target classes. Moreover, by tying this costs to the false positive rates of the cascade stages it is possible to guarantee the multi-resolution behavior. We use this strategy to design a Boosting algorithm for learning multiclass detector cascades and show, through experiments in multi-view car detection and detection of multiple traffic signs,

that the resulting classifiers are faster and more accurate than those previously available.

I.C Organization of the dissertation

The rest of the dissertation is organized as follows. In Chapter II, we present a new framework (TaylorBoost) for understanding Boosting and deriving new Boosting algorithms. TaylorBoost is then use for designing new Boosting algorithms in the rest of this dissertation. In Chapter III, the problem of learning optimal classifier cascade for detecting single class of objects is considered. A new cascade Boosting algorithm, *fast cascade Boosting* (FCBoost), is propose. In Chapter IV, we start the transition to multiclass object detection by considering the problem of multiclass Boosting. In this chapter, we formulate multiclass Boosting as an optimization in multi-dimensional function space and propose two Boosting algorithms (CD-MCBoost & GD-MCBoost) to solve this problem. In Chapter V, we introduce the concept of multi-resolution detector and use the MCBoost and FCBoost algorithms to design a multiclass detector cascade. Finally, conclusions are provided in Chapter VI.

Chapter II

TaylorBoost

II.A Introduction

Modern solutions to many machine learning and computer vision problems involve designing a classifier. Boosting is a reliable tool for this task and can produce a very accurate classifier by combining several weak classifiers. The first successful Boosting algorithm, AdaBoost, was introduced in [26] and since then a number explanations have been proposed to justify the effectiveness of Boosting including [27, 74, 53]. In particular, [27] showed that from *statistical* point of view AdaBoost is equivalent to gradient descent in function space to minimizing risk of exponential loss. In addition, [53] proposed more a general algorithm that can implement first order descent methods such as gradient descent, for any arbitrary loss function.

In this chapter, we 1) overview the statistical view of Boosting and show the characteristics of its optimization problem, 2) show that the current statistical view of Boosting is only valid when using gradient descent and will fail if using second-order-descent methods, 3) propose a new Boosting framework *TaylorBoost*. TaylorBoost is based on the Taylor series expansion of the risk, that can be applied to any loss function. It leads to a family of Boosting algorithms of either first or second order (depending on the approximation), denoted GradBoost and QuadBoost respectively. It is shown that first order TaylorBoost is equivalent to AnyBoost [53, 28] and when applied to logistic and exponential losses QuadBoost is identical to LogitBoost and GentleBoost [27]. Moreover although QuadBoost is a second order method it dose not require the computational intensive task of matrix inversion and its complexity is the same as GradBoost. Finally TaylorBoost frameworks shows that there are two different weighting mechanisms in Boosting. The first one originates form the first order derivative and concentrates on harder examples. The second one originates form the second order derivative and concentrates on points that are close to the boundary of classification.

II.B Boosting and optimization in function space

A binary classifier is a mapping $F : \mathcal{X} \rightarrow \mathcal{Y}$ which maps example $x \in \mathcal{X}$ to its label $y \in \mathcal{Y} = \{+1, -1\}$ where x, y are random variables. $F(x)$ is usually implemented as

$$F(x) = sign[f(x)], \tag{II.1}$$

where $f(x)$ is a real valued function called *predictor*. If $\mathcal{H}$ is the set of all possible real-valued functions, i.e. $\mathcal{H} = \{f | f : \mathcal{X} \rightarrow \mathbb{R}\}$, then the optimal predictor is the minimizer of

$$\begin{cases} \min_{f(x)} & R_c(f) \\ s.t & f(x) \in \mathcal{H} \end{cases} \tag{II.2}$$

where

$$R_c(f) = E_{X,Y}\{L[yf(x)]\}, \tag{II.3}$$

is called *risk of classification* and $L(.)$ is a *loss function* that penalizes the errors. The commonly used losses are functions of margin, $yf(x)$.

The optimization problem of (II.2) does not have a closed form solution and is usually solved by iterative descent methods in function space called Boosting. At k^{th} iteration of any Boosting method, an update $h^k(x) \in \mathcal{H}$ is added to the current estimate, i.e. $f^{k+1}(x) = f^k(x) + h^k(x)$, to reduce the objective function. Most Boosting methods rely on Gradient descent or Newton methods which require computation of the first and second order derivatives of the objective function, $R_c(f)$, in the function space of $\mathcal{H}$ which is an infinite dimensional space. However, the concept of derivative in this space is different than the traditional Gradient and Hessian in finite dimensional vector spaces. In this case there are several definition: *Gautex* derivative [80], *Frechet* derivative [29] and derivative with conditioning [27]. In this work we mainly consider Gautex derivative because of its convenient definition and properties.

II.B.1 Gautex derivative

Gautex derivative is the analogous of the directional gradient in finite dimensional case. The Gautex derivative of $R_c : \mathcal{H} \to \mathbb{R}$ at point $f \in \mathcal{H}$ along a direction $g \in \mathcal{H}$ is [80]

$$\delta R_c(f;g) = \left.\frac{\partial R_c(f+\xi g)}{\partial \xi}\right|_{\xi=0}, \tag{II.4}$$

where we omitted x for notational simplicity. Therefore $\delta R_c(f;g)$ is a real number which indicates the changes in the surface of R_c at point f if a particle moves along the direction g.

Using (II.3) and linear property of expected value

$$\begin{aligned}
\delta R_c(f;g) &= \left.\frac{\partial}{\partial \xi} E_{X,Y}\{L[y(f(x)+\xi g(x))]\}\right|_{\xi=0} && \text{(II.5)}\\
&= E_{X,Y}\left\{\left.\frac{\partial L[yf(x)+\xi y g(x)]}{\partial \xi}\right|_{\xi=0}\right\} && \text{(II.6)}\\
&= E_{X,Y}\left\{y g(x) L'[yf(x)]\right\}, && \text{(II.7)}
\end{aligned}$$

where $L'(v) = \frac{dL(v)}{dv}$. Similarly the second order variation of R_c at point f along a direction $g \in \mathcal{H}$ is [29]

$$\begin{aligned}
\delta^2 R_c(f;g) &= \left.\frac{\partial^2 R_c(f+\xi g)}{\partial \xi^2}\right|_{\xi=0} && \text{(II.8)}\\
&= \left.\frac{\partial^2}{\partial \xi^2} E_{X,Y}\{L[yf(x)+\xi y g(x)]\}\right|_{\xi=0} && \text{(II.9)}\\
&= E_{X,Y}\left\{\left.\frac{\partial^2 L[yf(x)+\xi y g(x)]}{\partial \xi^2}\right|_{\xi=0}\right\} && \text{(II.10)}\\
&= E_{X,Y}\left\{y^2 g^2(x) L''[yf(x)]\right\} && \text{(II.11)}\\
&= E_{X,Y}\left\{g^2(x) L''[yf(x)]\right\}, && \text{(II.12)}
\end{aligned}$$

where $L''(v) = \frac{d^2L(v)}{dv^2}$. $\delta^2 R_c(f;g)$ is also a real number measuring the curvature of the path of a particle which moves on the surface of R_c at point f along the direction g.

Using (II.4), (II.8) the first order Taylor approximation of R_c around $f(x)$ in

function space is

$$R_c(f + \epsilon g) = R_c(f) + \epsilon \delta R_c(f; g) + o(\epsilon^2), \tag{II.13}$$

and the second order approximation would be

$$R_c(f + \epsilon g) = R_c(f) + \epsilon \delta R_c(f; g) + \frac{\epsilon^2}{2} \delta^2 R_c(f; g) + o(\epsilon^3). \tag{II.14}$$

II.B.2 Practical limitations

In theory, the definitions presented in the previous section can be used for solving optimization problem of (II.2) by any first or second-order methods. In practice, however, there are two main limitations: 1) only a relatively small set of examples from $\mathcal{X}$ and $\mathcal{Y}$, named training samples, are available and 2) only a small portion of functions, named weak learners, are available to use. We call this problems *sampling* and *function* limitations respectively.

Sampling effect

The training set in most of the learning problems are only a small set of examples along with their labels, $S_t = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y} = \{+1, -1\}\}$. Given the limited training data, computing the expected values over the whole example space of $\mathcal{X}$ in (II.17), (II.7) and (II.12) is impossible and they should be approximated by the empirical averages

$$\delta R_c(f; g) \approx \frac{1}{|S_t|} \sum_{(x_i, y_i) \in S_t} y_i g(x_i) L'[y_i f(x_i)] \tag{II.15}$$

$$\delta^2 R_c(f; g) \approx \frac{1}{|S_t|} \sum_{(x_i, y_i) \in S_t} g^2(x_i) L''[y_i f(x_i)] \tag{II.16}$$

$$R_c(f) \approx \frac{1}{|S_t|} \sum_{(x_i, y_i) \in S_t} L[y_i f(x_i)]. \tag{II.17}$$

Function limitations

In practice only a limited part of $\mathcal{H}$ is available for use and most of the elements (function) of $\mathcal{H}$ are either unknown or very expensive to find and

compute. The available function, $g : \mathcal{X} \to \mathbb{R}$ are called weak learner and let $\mathcal{G}$ be the set of all weak learners, i.e. $\mathcal{G} = \{g_1(x), .., g_m(x)\}$. This limitation make it impossible to find the best predictor, $f^* \in \mathcal{H}$ and f^* should be approximated as a combination of weak learners. For this purpose, Boosting methods usually consider f^* as a linear combination of weak learners i.e. $f^*(x) = \sum_k \alpha_k g_k(x)$. In this case Boosting algorithms solve

$$\begin{cases} \min_{f(x)} & R_c(f) = \frac{1}{|S_t|} \sum_{(x_i,y_i)\in S_t} L[y_i f(x_i)] \\ s.t & f(x) \in \Omega_{\mathcal{G}}, \end{cases} \tag{II.18}$$

where $\Omega_{\mathcal{G}}$ is the set of all linear combinations of weak learners in $\mathcal{G}$. Note that there is no limitation on the magnitude of weak learners, and they only represents directions in the function space, therefore in the rest of this dissertation we assume $g_i \in \mathcal{G}$ are normalized i.e. $< g, g >= 1$. We next propose a family of optimization algorithms for solving (II.18).

II.C TaylorBoost

In this section, we derive a new family of Boosting algorithms, TaylorBoost, for solving (II.18). The distinguishing features of TaylorBoost are 1) it relies on Taylor series expansion of the risk R_c in the function space, 2) encompasses first and second order descent methods, and 3) generalizes previous Boosting methods.

At k^{th} iteration of TaylorBoost an update, $g^*(x) \in \mathcal{G}$, is selected and added to the current predictor as

$$f^{k+1}(x) = f^k(x) + \alpha^* c_{g^*} g * (x), \tag{II.19}$$

where c_g in (II.19) is the best step size along a weak learner g to minimize the approximation $\overline{R_c}(f^k + cg)$

$$c_g \quad = \quad \arg\min_{c\in\mathbb{R}} \overline{R_c}(f^k + cg), \tag{II.20}$$

$\overline{R_c}(f^k + cg)$ is the Taylor series expansion of the risk R_c around the current predictor, f^k

$$\overline{R_c}(f^k + cg) = R_c(f^k) + c\delta R_c(f^k; g) + \frac{c^2}{2}\delta^2 R_c(f^k; g) + .. \tag{II.21}$$

and g^* is the best weak learner

$$g^* \quad = \quad \arg\min_{g \in \mathcal{G}} \overline{R_c}(f^k + c_g g). \tag{II.22}$$

Finally, α^* is another step size to guarantee minimization of the true objective function $R_c(f)$

$$\alpha^* \quad = \quad \arg\min_{\alpha \geq 0} \; R_c(f_k + \alpha h^k). \tag{II.23}$$

Note that α^* may have a closed form as in AdaBoost [26] or should be found by a line search.

By controlling the order of the approximation in (II.19), it is possible to obtain different Boosting algorithms. For an approximation of the second order

$$c_g \quad = \quad \arg\min_{c \in \mathbb{R}} R_c(f^k) + c\delta R_c(f^k; g) + \frac{c^2}{2}\delta^2 R_c(f^k; g) \tag{II.24}$$

$$= \quad -\frac{\delta R_c(f^k; g)}{\delta^2 R_c(f^k; g)}. \tag{II.25}$$

and

$$g^* \quad = \quad \arg\min_{g \in \mathcal{G}} R_c(f^k) + c_g \delta R_c(f^k; g) + \frac{c_g^2}{2}\delta^2 R_c(f^k; g) \tag{II.26}$$

$$= \quad \arg\min_{g \in \mathcal{G}} R_c(f^k) - \frac{\delta R_c(f^k; g)}{\delta^2 R_c(f^k; g)}\delta R_c(f^k; g) \tag{II.27}$$

$$+ \quad \frac{1}{2}\frac{[\delta R_c(f^k; g)]^2}{[\delta^2 R_c(f^k; g)]^2}\delta^2 R_c(f^k; g) \tag{II.28}$$

$$= \quad \arg\min_{g \in \mathcal{G}} R_c(f^k) - \frac{1}{2}\frac{[\delta R_c(f^k; g)]^2}{\delta^2 R_c(f^k; g)} \tag{II.29}$$

$$\equiv \quad \arg\max_{g \in \mathcal{G}} \frac{[\delta R_c(f^k; g)]^2}{\delta^2 R_c(f^k; g)}. \tag{II.30}$$

Using (II.25) , (II.30) and (II.23) the predictor is updated according to (II.19). We denote this method *Quadratic Boosting*, or QuadBoost.

For an approximation of the first order

$$c_g = \arg\min_{c\in\mathbb{R}} R_c(f^k) + c\delta R_c(f^k; g) \tag{II.31}$$

$$g^* = \arg\min_{g\in\mathcal{G}} R_c(f^k) + c_g\delta R_c(f^k; g), \tag{II.32}$$

but in this case c_g is not well defined and the optimization problem of (II.31) is unbounded. In addition, the optimization of (II.32) is sensitive to the magnitude of weak learners. Therefore we assume weak learners are normalized i.e. $< g, g >= 1\ \forall g \in \mathcal{G}$, and set $c_g = 1\ \forall g \in \mathcal{G}$. It follows that

$$g^* = \arg\min_{g\in\mathcal{G}} R_c(f^k) + \delta R_c(f^k; g) \tag{II.33}$$

$$\equiv \arg\max_{g\in\mathcal{G}} -\delta R_c(f^k; g). \tag{II.34}$$

Using (II.34), $c_g = 1$ and (II.23) the predictor is updated according to (II.19). This method is identical to GradientBoost or AnyBoost [28, 53] and is called GradBoost in the rest of this dissertation.

In summary, the weak learner selection rule for first order method is

$$g^* = \arg\max_{g\in\mathcal{G}} -\delta R_c(f^k; g), \quad c_{g^*} = 1, \tag{II.35}$$

and the second order method is

$$g^* = \arg\max_{g\in\mathcal{G}} \frac{[\delta R_c(f^k; g)]^2}{\delta^2 R_c(f^k; g)}, \quad c_{g^*} = -\frac{\delta R_c(f^k; g^*)}{\delta^2 R_c(f^k; g^*)}. \tag{II.36}$$

The first and second order TaylorBoost algorithms are presented in Algorithm 1. These algorithms are simple and compatible with any definition of risk of classification, R_c. However, the classical definition of risk, (II.3), has a special form which makes it possible to further simplify these algorithms.

II.C.1 Simplified TaylorBoost

The classical definition of risk of classification is the *summation* of the loss function over all training examples (II.17). This summation form induces several simplification on TaylorBoost. First, using (II.15), (II.16) the expressions

Algorithm 1 TaylorBoost

Input: Training set $S_t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y \in \{1, -1\}$ is the class label of example $\mathbf{x}$. A normalized set of weak learners $\mathcal{G} = \{g_1, ...g_m\}$. Number of weak learners in the final classifier N. A definition of risk of classification R_c.

Initialization: Set $k = 0$ and $f^k(x) = 0$.

while $k < N$ **do**

For first order TaylorBoost (GradBoost)

$$g^* = \arg\max_{g \in \mathcal{G}} -\delta R_c(f^k; g), \quad c_{g^*} = 1.$$

For second order TaylorBoost (QuadBoost)

$$g^* = \arg\max_{g \in \mathcal{G}} \frac{[\delta R_c(f^k; g)]^2}{\delta^2 R_c(f^k; g)}, \quad c_{g^*} = -\frac{\delta R_c(f^k; g^*)}{\delta^2 R_c(f^k; g^*)}.$$

Set $h^k(x) = c_{g^*} g^*(x)$ and find the optimal step size

$$\alpha^* = \arg\min_{\alpha \geq 0} R_c(f^k + \alpha h^k).$$

Update $f^{k+1}(x) = f^k(x) + \alpha^* h^k(x)$.

Update $k = k + 1$.

end while

Output: decision rule: $sign[f^N(x)]$

for $\delta R_c(f^k; g)$ and $\delta^2 R_c(f^k; g)$ are simple summations of derivatives of the loss function. Second, as shown in appendix II.G.1, solving the weak learner selection rules in TaylorBoost (II.35) and (II.36) can be summarized as

$$g^* = \arg\max_{g \in \mathcal{G}} \frac{[\sum_i y_i w_i^f g(x_i)]^2}{\sum_i w_i^s g(x_i)^2}, \quad c_{g^*} = \frac{\sum_i y_i w_i^f g(x_i)}{\sum_i w_i^s g(x_i)^2}, \tag{II.37}$$

where

$$w_i^f = w^f(x_i) = -L'(y_i f(x_i)) \tag{II.38}$$

$$w_i^s = w^s(x_i) = \begin{cases} 1 & \text{in GradBoost} \\ L''(y_i f(x_i)) & \text{in QuadBoost.} \end{cases} \tag{II.39}$$

Algorithm 2 Simplified TaylorBoost

Input: Training set $S_t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y \in \{1, -1\}$ is the class label of example $\mathbf{x}$. A normalized set of weak learners $\mathcal{G} = \{g_1, -g_1, ..., g_m, -g_m\}$. Number of weak learners in the final classifier N. A convex loss function, $L[yf(x)]$.

Initialization: Set $k = 0$ and $f^k(x) = 0$.

while $k < N$ **do**

Compute $w_i^f = -L'[y_i f^k(x_i)]$.

For first order TaylorBoost (GradBoost) set $w_i^s = 1$.

For second order TaylorBoost (QuadBoost) set $w_i^s = L''[y_i f^k(x_i)]$.

Find the best update $h^k(x) = c_{g^*} g^*(x)$ using

$$g^* = \arg\max_{g \in \mathcal{G}} \frac{[\sum_i y_i w_i^f g(x_i)]^2}{\sum_i w_i^s g(x_i)^2}, \quad c_{g^*} = \frac{\sum_i y_i w_i^f g(x_i)}{\sum_i w_i^s g(x_i)^2},$$

or

$$(g^*, c_{g^*}) \quad = \quad \arg\max_{g \in \mathcal{G}, c_g \in \mathbb{R}} \sum_i w_i^s \left[y_i \frac{w_i^f}{w_i^s} - c_g g(x_i) \right]^2.$$

Find the optimal step size $\alpha^* = \arg\min_{\alpha \geq 0} R_c(f^k + \alpha h^k)$.

Update $f^{k+1}(x) = f^k(x) + \alpha^* h^k(x)$.

Update $k = k + 1$.

end while

Output: decision rule: $sign[f^N(x)]$

Third, Appendix II.G.2 shows that solving (II.37) is equivalent to solving the least square problem of

$$(g^*, c_{g^*}) \quad = \quad \arg\max_{g \in \mathcal{G}, c_g \in \mathbb{R}} \sum_i w_i^s \left[y_i \frac{w_i^f}{w_i^s} - c_g g(x_i) \right]^2. \tag{II.40}$$

Using these results, the simplified TaylorBoost is presented in Algorithm 2. In addition for more convenient use of Algorithm 2, Table II.1 shows w_i^f and w_i^s for four loss functions of Exponential loss, Logistic loss, Canonical Boosting loss

(CBL) [51] and laplace loss [70]. The definition and shape of these loss functions are presented in Figure II.1. Note that when using exponential and Logistic losses QuadBoost is equivalent to GentleBoost and LogitBoost respectively [27].

II.D Properties

The TaylorBoost family has a number of interesting properties. The first property of TaylorBoost is its insight to Boosting. This makes it possible to modify current Boosting methods or create new Boosting algorithms to address requirements of specific problems where different definitions for loss functions, risk of classification or weak learners are required. In the rest of this dissertation, TaylorBoost will be used for this purpose. Second, TaylorBoost generalizes the classical Boosting methods and unify them on a simple framework of algorithm 2. In particular GradientBoost or AnyBoost [28, 53] are equivalent to TaylorBoost of first order (GradBoost) and LogitBoost and GentleBoost are application of second order TaylorBoost (QuadBoost) to logistic and exponential loss respectively. Third, due to more accurate approximation of second order methods, QuadBoost selects better weak learners and has faster rate of convergence than GradBoost. Finally, although second order methods are sometimes expensive (need to invert the Hessian), in QuadBoost there is no need for inverting the hessian either in the Simplified TaylorBoost or the regular TaylorBoost. Therefore TaylorBoost framework makes it possible to build faster Boosting method with no additional computational cost.

II.D.1 Weights in TaylorBoost

From the algorithmic point of view, Boosting is about reweighing training examples and focusing on more difficult ones in each iteration [26]. This section clarifies the origin and functionality of those weighting mechanisms.

Using the notations of algorithm 2, in AdaBoost and first order methods,

Loss	$L[y, f(x)]$
Exponential	$e^{-yf(x)}$
Logistic	$\log(1 + e^{-2yf(x)})$
Laplace	$e^{-\lvert yf(x)\rvert} + \lvert yf(x)\rvert - yf(x)$
CBL	$[\sqrt{4 + (yf(x))^2} - yf(x)]$

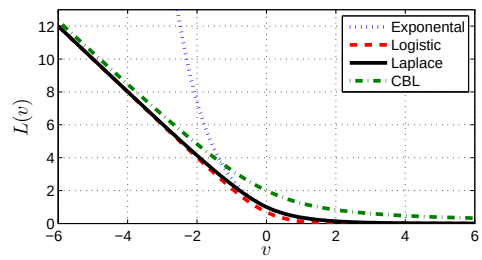
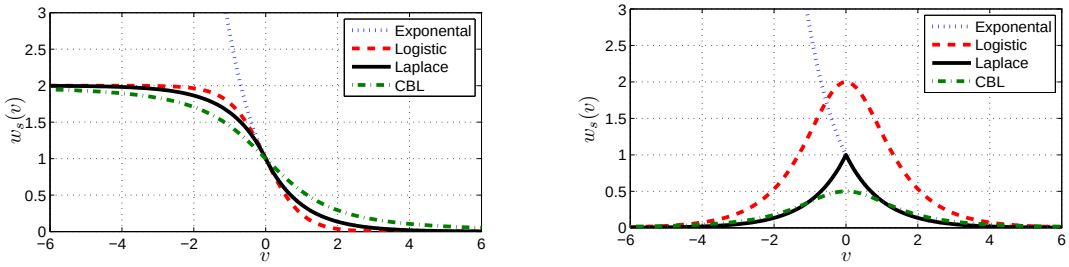


Figure II.1: Left: Definition of different loss function. Right: Plots of different losses as a function of $v = yf(x)$.

Table II.1: Formula for computing first order weights, $w^f(x)$, and second order weights, $w^s(x)$, for different loss function.

Loss	$w^f(x)$	$w^s(x)$
Exponential	$e^{-yf^k(x)}$	$e^{-yf(x)}$
Logistic	$\frac{2e^{-2yf(x)}}{1+e^{-2yf(x_i)}}$	$\left[\frac{2}{e^{yf(x)}+e^{-yf(x)}}\right]^2$
Laplace	$sign[yf(x)][1 - e^{-\lvert yf(x)\rvert}] - 1$	$e^{-\lvert yf(x)\rvert}$
CBL	$1 - \frac{f(x)}{\sqrt{4+f(x)^2}}$	$\frac{4}{\sqrt{(4+f(x)^2)^3}}$

w^f is known as weight [27]. Similarly for LogitBoost and GentleBoost, which are second order methods, the coefficients w^s are called the weights [27]. Therefore, algorithm 2 clarifies that 1) there are two type of weighting mechanisms in Boosting, 2) the first kind originates form the first order derivative of the loss function while the other kind originates from the second order derivative and, 3) first order methods use only w^f while second order methods use both w^f and w^s. Figure II.2-a,b show $w^f(.)$ and $w^s(.)$ as functions of margin $v = yf(x)$ for loss functions shown in Figure II.1. As shown in the Figure II.2-a, w^f is always larger for points with negative margin. But negative margin indicates misclassification, therefore w^f helps Boosting to concentrate on misclassified (harder) examples. Moreover, Figure II.2-b shows that, except for exponential loss, w^s is larger if v is close to zero. But $v = 0$ represent the classification boundary, therefore w^s helps Boosting to concentrate on the examples that are closer to the boundary. Note that for exponential loss the effect of w^s is the same as w^f.



(a) (b)

Figure II.2: a) first, w^f, and b) second, w^s, type of weights in Boosting as functions of margin $v = yf(x)$ for different loss functions.

The effect of w^s can be explained by statistical view of Boosting. In [27] it is shown that Boosting learns the log-likelihood ratio (LLR) surface. However this only holds *asymptotically* (infinite training sets) and [54, 49] showed that, for *finite* samples, Boosting only learns the LLR surface in a *neighborhood* $\mathcal{N}_B$ of the classification boundary. Outside this neighborhood the approximation can be very weak. Therefore using w^s Boosting can trades-off the quality of the approximation within $\mathcal{N}_B$, by that of outside $\mathcal{N}_B$.

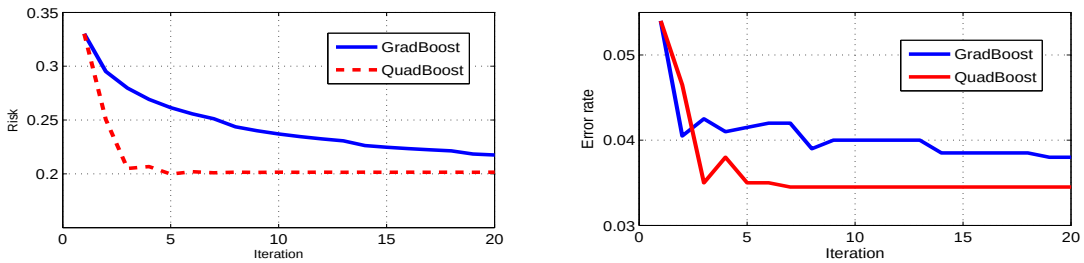
II.D.2 Convexity

If $R_c(f)$ of (II.3) is a (strictly) convex function of f in $\mathcal{H}$, then (II.2) would be a (strictly) convex optimization problem and will have a (unique) solution. On the other hand $R_c(f)$ is convex if and only if $\delta^2 R_c(f;g) \geq 0 \quad \forall f, g \in \mathcal{H}$ which, using (II.12), is equivalent to

$$\forall f \in \mathcal{H}, \qquad \forall x, y \in \mathcal{X}, \mathcal{Y} \qquad L''(yf(x)) \geq 0. \tag{II.41}$$

II.E Experiments

In this section several experiments were performed to evaluate our theory and algorithms. The first experiment compares first and second order methods (GradBoost vs QuadBoost) both on synthetic and real data sets. The second experiment illustrates application of QuadBoost in discriminant tracking, and compares



(a) (b)

Figure II.3: a) Risk of classification, $R_c(f)$ and b) error rates as function of iterations for classifiers trained with GradBoost and QuadBoost.

its performance with previous Boosting methods. Similar to [27], In all experiments the weak learners are regression on the example coordinates, (features).

II.E.1 GradBoost vs. QuadBoost

We start with comparing the first and second order Boosting methods on synthetic and real data sets.

Synthetic data

The data set in this section is based on random sampling from two Gaussian distribution in $\mathbb{R}^2$ with means $[2, 2]$, $[2, -2]$ and covariance matrices $[1, 0.5; 0.5, 2]$, $[0.4, 0.1; 0.1, 0.8]$ respectively. Both training and test set have $1,000$ example form each distribution. We trained first and second order Boosted classifiers using logistic loss. Figure II.3 shows the evolution of risk of classification (left) and error rate (right) on the test set. As this figure shows when using the second order method, Boosting converges to the minimum faster and results in better performance.

Real Data

For comparing GradBoost and QuadBoost on real data we have selected 6 UCI data set and three large object detection data sets for face, car and pedestrian detection. The UCI data set are 'Image', 'Ringnorm', 'Splice', 'Thyroid', 'Titanic', 'Twonorm', 'Waveform'. This data comes with 100 predefined splits for training

and test sets [1] [1] and we report the average results. The face data set consists of $9,000$ face and $9,000$ non-face images, of size 24×24. Car detection is based on the UIUC data set [3] of $1,100$ positives and $10,000$ negatives, of size 20×50. Pedestrian detection is based on the MIT Pedestrian data set [59] of $1,000$ positives and $10,000$ negatives, of size 40×20. In all cases, the data was split into five folds, four of which were used for training and one for testing. All experiments were repeated with each fold taking the role of test set, and the results averaged.

Table II.2 presents the error rates of classifiers learned with 20 iterations of all combinations of GradBoost, QuadBoost and the losses of Figure II.1. A number of observation can be made. First, in 28/36 cases QuadBoost has better performance in 5/36 GradBoost was better and in 3/36 the performance of Grad and QuadBoost are the same. Second, the improvements of QuadBoost vs GradBoost is up to 51% in car data set when using CBL loss function and in the worst case using QuadBoost degrades the results by 2% (Thyroid and Exp loss). Third comparing the best result on each data set (bold entries on Table II.2) in 6/9 QuadBoost had the best performance, in 2/9 GradBoost was better and in 1/9 Quad and GradBoost have the same accuracy. Fourth, comparing the loss functions, CBL loss achieved the best detectors in 5 data sets, Laplace in 4 data sets, logistic loss in 2 data sets and exponential loss in 1 data sets. Fifth, note that the improvements in QuadBoost are gained without any substantial increase in learning complexity because, according to algorithm 2, the only difference between QuadBoost and GradBoost is in the definition of coefficient $w^s(x)$.

In summary, comparing to GradBoost, QuadBoost has better performance without any increase in complexity of training. This is a consequence of its tighter local approximation of R_c (second order vs first order) which results in selection of more efficient weak learners and faster convergence.

[1] For 'Ringnorm', 'Splice' the data has 20 splits

Table II.2 Classication error of various combinations of Boosting and loss functions.

	Exp		*Log*		*CBL*		*Lap*	
	Grad	Quad	Grad	Quad	Grad	Quad	Grad	Quad
Waveform	$14.3 \pm .1$	$13.7 \pm .1$	$14.1 \pm .1$	$13.4 \pm .1$	$13.4 \pm .1$	$\mathbf{13.3 \pm .1}$	$14.2 \pm .1$	$\mathbf{13.3 \pm .1}$
Twonorm	$4.5 \pm .0$	$\mathbf{3.3 \pm .0}$	$5.1 \pm .0$	$3.3 \pm .0$	$4.3 \pm .0$	$3.4 \pm .0$	$5.1 \pm .0$	$3.3 \pm .1$
Thyroid	$11.1 \pm .3$	$11.3 \pm .3$	$11.2 \pm .3$	$10.6 \pm .2$	$12.7 \pm .3$	$\mathbf{10.5 \pm .2}$	$11.7 \pm .3$	$\mathbf{10.5 \pm .2}$
Splice	$16.9 \pm .2$	$16.8 \pm .2$	$16.5 \pm .2$	$16.4 \pm .2$	$16.6 \pm .2$	$16.7 \pm .2$	$\mathbf{16.4 \pm .1}$	$16.6 \pm .2$
Ringnorm	$26.3 \pm .1$	$26.4 \pm .1$	$\mathbf{25.3 \pm .1}$	$\mathbf{25.3 \pm .1}$	$25.9 \pm .1$	$25.4 \pm .1$	$25.4 \pm .1$	$25.4 \pm .1$
Image	$20.4 \pm .2$	$20.3 \pm .2$	$20.0 \pm .2$	$19.6 \pm .2$	$22.2 \pm .3$	$\mathbf{19.2 \pm .2}$	$20.2 \pm .2$	$\mathbf{19.2 \pm .2}$
Face	$16.6 \pm .3$	$16.6 \pm .3$	$\mathbf{16.4 \pm .3}$	$16.5 \pm .4$	$17.4 \pm .3$	$16.5 \pm .3$	$16.8 \pm .3$	$16.5 \pm .3$
Car	$4.6 \pm .1$	$3.6 \pm .1$	$4.3 \pm .1$	$3.2 \pm .1$	$6.2 \pm .1$	$\mathbf{3.0 \pm .1}$	$5.1 \pm .1$	$3.1 \pm .1$
Pedestrian	$8.6 \pm .2$	$8.4 \pm .2$	$8.0 \pm .1$	$7.2 \pm .1$	$8.8 \pm .1$	$\mathbf{6.9 \pm .1}$	$8.3 \pm .2$	$7.0 \pm .0$

II.E.2 Discriminant Tracking

Discriminant tracking is a state-of-the-art object tracking approach. In this method, a classifier is trained to distinguish the target object from the surrounding background in each video frame. This classifier is then applied to the next frame in order to detect the object. This procedure is repeated and results in tracking the object of interest [5]. Various methods have been proposed for learning the classifiers, including AdaBoost [5], discriminant saliency [48], and a combination of discriminant saliency and TangentBoost [52]. The latter has been reported to achieve the best results in the literature.

One of the main challenges in design of the classifiers for this approach is the rate of convergence of the learning method. This is in particular important because of the time constraints, such as real-time tracking, requires the learning algorithm to run only for a small number of iterations, i.e. 10 to 20. Therefore if a learning method has faster rate of convergence then it will produce more accurate classifier with small number of iterations and result in more accurate tracker. Using this fact and the results of previous experiment, in this section we combine discriminant tracking with QuadBoost.

For QuadBoost-discriminant tracking we used regression of discriminant saliency features provides by [48] as weak learners and the classifier for each frame was learned with up to 20 iterations. The experiment is repeated with four loss functions of exponential, logistic, CBL and laplace loss. In addition we have also implemented the TangentBoost [52] for comparison. Note that TangentBoost is a first order method and its loss function, Tangent loss, is not convex. Therefore it is not possible to create a proper second order Boosting, QuadBoost, using the tangent loss. Table II.3 presents the error rates (as defined in [48]) for each method on five video clips. The combination of Laplace, CBL and QuadBoost each achieves the best result on two clips, while TangentBoost has the lowest error rate on the last one.

Table II.3 Tracking accuracy for five video clips.

Method	Tangent	Log+Quad	Exp+Quad	Lap+Quad	CBL+Quad
Gravel	18.6 %	18.8 %	19.2%	18.8 %	**17.8** %
Athlete	36.6 %	37.2 %	36.4%	35.8 %	**35.4** %
Karls	79.9 %	31.7 %	33.7%	**31.4** %	35.3 %
Montins	86.9 %	92.2 %	92.5%	**83.4** %	87.7 %
Plush	**8.9**%	9.4 %	9.5%	10.2 %	10.9 %

II.F Acknowledgments

The text of Chapter II, in part, is based on the material as it is appeared in: Mohammad Saberian, Hamed Masnadi-Shirazi and Nuno Vasconcelos. TaylorBoost: First and Second-order Boosting Algorithms with Explicit Margin Control. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. The dissertation author was a primary researcher and an author of the cited material.

II.G Appendix

II.G.1 Weak learner selection rule for simplified TaylorBoost

We start with the assumption that if $\forall g \in \mathcal{G} \Rightarrow -g \in \mathcal{G}$. This is a valid assumption in most of the cases because we can build $-g(x)$ by simply changing the sign of $g(x)$. Using this assumption,

$$g^* = \arg\max_{g\in\mathcal{G}} -\delta R_c(f^k; g) = \overline{g} \times sign\left[-\delta R_c(f^k; \overline{g})\right], \tag{II.42}$$

where

$$\overline{g} = \arg\max_{g\in\mathcal{G}} \left[\delta R_c(f^k; \overline{g})\right]^2. \tag{II.43}$$

In this case $\overline{g} = g^*$ or $\overline{g} = -g^*$ and $sign\left[-\delta R_c(f^k; \overline{g})\right]$ corrects the sign such that (II.42) holds. Combining (II.30), (II.42) and using the normalization assumption about weak learners, results in

$$g^* = \arg\max_{g\in\mathcal{G}} \frac{[\sum_i y_i w_i^f g(x_i)]^2}{\sum_i w_i^s g(x_i)^2}, \quad c_{g^*} = \frac{\sum_i y_i w_i^f g(x_i)}{\sum_i w_i^s g(x_i)^2} \tag{II.44}$$

where

$$w_i^f = w^f(x_i) = -L'(y_i f(x_i)) \tag{II.45}$$

$$w_i^s = w^s(x_i) = \begin{cases} 1 & \text{in GradBoost} \\ L''(y_i f(x_i)) & \text{in QuadBoost.} \end{cases} \tag{II.46}$$

II.G.2 Least square interpretation

Solving

$$(g^*, c_{g^*}) = \arg\min_{g\in\mathcal{G}, c_g\in\mathbb{R}} \sum_i w_i^s \left[y_i \frac{w_i^f}{w_i^s} - c_g g(x_i)\right]^2, \tag{II.47}$$

we first compute the derivative with respect to c_g and set it to zero

$$\sum_i w_i^s g(x_i)[y_i \frac{w_i^f}{w_i^s} - c_g g(x_i)] = \sum_i g(x_i) y_i w_i^f - c_g \sum_i g^2(x_i) w_i^s = 0. \tag{II.48}$$

Therefore

$$c_g = \frac{\sum_i g(x_i) y_i w_i^f}{\sum_i g^2(x_i) w_i^s}, \tag{II.49}$$

and using this in (II.47)

$$
\begin{aligned}
g^*(x) &= \arg\min_{g\in\mathcal{G}} \sum_i w_i^s \left[y_i \frac{w_i^f}{w_i^s} - c_g g(x_i) \right]^2 \\
&= \arg\min_{g\in\mathcal{G}} \sum_i y_i^2 \frac{[w_i^f]^2}{w_i^s} + c_g^2 \sum_i w_i^s g(x_i)^2 - 2c_g \sum_i y_i w_i^f g(x_i) \\
&= \arg\min_{g\in\mathcal{G}} \left[\frac{\sum_i g(x_i) y_i w_i^f}{\sum_i g^2(x_i) w_i^s} \right]^2 \sum_i w_i^s g(x_i)^2 - 2 \frac{\sum_i g(x_i) y_i w_i^f}{\sum_i g^2(x_i) w_i^s} \sum_i y_i w_i^f g(x_i) \\
&= \arg\max_{g\in\mathcal{G}} \frac{[\sum_i g(x_i) y_i w_i^f]^2}{\sum_i g^2(x_i) w_i^s}. \qquad \text{(II.50)}
\end{aligned}
$$

Chapter III

Boosting Detector Cascade

III.A Introduction

Designing an object detector requires designing a classifier to distinguish between target and non-target input images. This classifier is then applied to all candidate regions in an image to find all instance of a target. The main challenge in designing a real-time detector is that in each image there are millions of candidate region and the classifiers has to process all these candidates in a fraction of a second. One possibility to deal with this problem is to adopt sophisticated search strategies, such as branch-and-bound or divide-and-conquer, to reduce the number of sub-windows to classify [40, 89, 39]. While these methods are compatible with popular classification architectures, e.g. the combination of a support vector machine (SVM) and the bag-of-words image representation, they do not speed up the classifier itself. An alternative solution is to examine all sub-windows but adapt the complexity of the classifier to the difficulty of their classification. This strategy has been the focus of substantial attention since the introduction of the detector cascade architecture in [91]. As illustrated in Figure III.1 a) this architecture is implemented as a sequence of binary classifiers $h_1(x), \ldots h_m(x)$, known as the *cascade stages*. These stages have increasing complexity, ranging from a few machine operations for $h_1(x)$ to extensive computation for $h_m(x)$. An example x is declared a target by the cascade if and only if it is declared a target by all its stages. Since the overwhelming majority of sub-windows in an image do not contain the target object, a very large portion of the image is usually rejected by the early cascade stages. This makes the average detection complexity quite low. However, because the later stages can be arbitrarily complex, the cascade can have very good classification accuracy. This was convincingly demonstrated by [91], who used the cascade architecture to design the first real-time face detector with state-of-the-art classification accuracy. This detector has since found remarkable practical success, and is today popular in applications of face detection involving low-complexity processors, such as digital cameras or cell phones.

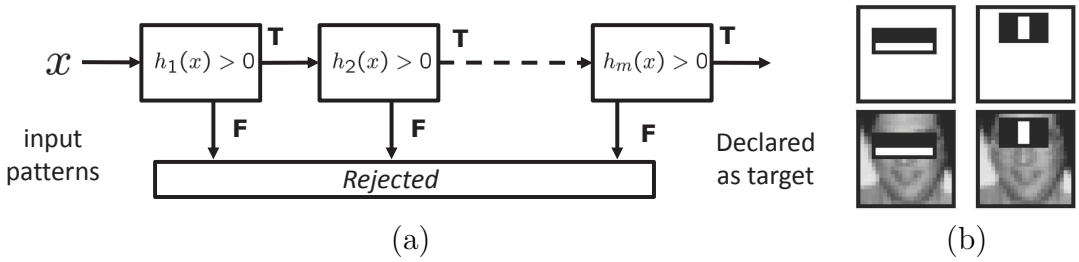


Figure III.1: (a) detector cascade and (b) examples of weak learners used for face detection [91].

In the method of [91], cascade stages are designed sequentially, by simply training each detector on the examples rejected by its predecessors. Each stage is designed by Boosting decision stumps that operate on a space of Haar wavelet features, such as those shown in Figure III.1-b). Hence, each stage is a linear combination of weak learners, each consisting of a Haar wavelet and a threshold. This has two appealing properties. First, because it is possible to evaluate each Haar wavelet with a few machine operations, cascade stages can be very efficient. Second, it is possible to control the complexity of each stage by controlling its number of weak learners. However, while fast and accurate, the detector of [91] is not optimal under any sensible definition of cascade optimality. For example, it does not address the problems of 1) how to automatically determine the optimal cascade configuration, e.g. the numbers of cascade stages and weak learners per stage, 2) how to design individual stages so as to guarantee optimality of the cascade as a whole, or 3) how to factor detection speed as an explicit variable of the optimization process. These limitations have motivated many enhancements to the various components of cascade design, including 1) new features [45, 15, 63, 60], 2) faster feature selection procedures [97, 62], 3) post-processing procedures to optimize cascade performance [45, 47, 82], 4) extensions of AdaBoost for improved design of the cascade stages [92, 50, 81, 76, 44, 87], 5) alternative cascade structures [99, 8, 98, 81], and 6) joint, rather than sequential stage design [22, 42, 81, 8]. While these advances improved on [91], the optimal design of a whole cascade is still an open problem. Most existing solutions rely on assumptions, such as the

independence of cascade stages, that do not hold in practice.

In this section, we address the problem of automatically learning both the configuration and the stages of a high detection rate detector cascade, under a definition of optimality that accounts for both classification accuracy and speed. This is accomplished with the *fast cascade Boosting* (FCBoost) algorithm, an extension of AdaBoost derived from a Lagrangian risk that trades-off detection performance and speed. FCBoost optimizes this risk with respect to a predictor that complies with the sequential decision making structure of the cascade architecture. These predictors are called *cascade predictors*, and it is shown that a rich family of such predictors can be derived recursively from a set of *cascade generator* functions, which are cascade predictors of two stages. Boosting algorithms are derived for two elements of this family, *last-stage* and *multiplicative* cascades. These are shown to generalize the cascades of embedded [99, 8, 98, 81, 50, 64] or independent [91, 76, 9, 97, 78, 79] stages commonly used in the literature. The search for the cascade configuration is naturally integrated in FCBoost by the introduction of *neutral predictors*. This allows FCBoost to automatically determine 1) number of cascade stages and 2) number of weak learners per stage, by simple minimization of the Lagrangian risk. The procedure is compatible with existing cost-sensitive extensions of Boosting [92, 50, 64, 49] that guarantee cascades of high detection rate, and generalizes AdaBoost in a number of interesting ways. A detailed experimental evaluation on face and pedestrian detection shows that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

The rest of this chapter is organized as follows. Section III.B reviews the challenges of cascade learning and previously proposed solutions. Section III.C briefly reviews AdaBoost, the most popular stage learning algorithm, and proposes its generalization for the learning of detector cascades. Section III.D studies the structure of cascade predictors, introducing the concept of cascade generators. Two generators are then proposed, from which two cascade families (last-stage

and multiplicative) are derived. The search for the cascade configuration is then studied in Section III.E. In this section the Lagrangian extension of the cascade Boosting algorithm is introduced, so as to account for detector complexity in the cascade optimization, and a procedure for the automatic addition of cascade stages during Boosting is developed, using neutral predictors. All these contributions are consolidated into the FCBoost algorithm in Section III.F, whose specialization to last-stage and multiplicative cascades is shown to generalize the two main previous approaches to cascade design. A number of interesting properties of the algorithm are also discussed, and a cost-sensitive extension is derived. Finally, an experimental evaluation is presented in Section III.G and some conclusions drawn in Section III.H. An early version of this work was presented in [71].

III.B Prior work

A large literature on detector cascade learning has emerged over the past decade. In this section, we briefly review the main problems in this area and their current solutions.

III.B.1 The problems of cascade learning

As illustrated in Figure III.1, a cascaded detector is a sequence of detector *stages*. The aim is to detect instances from a *target* class. Examples from this class are denoted *positives* while all others are denoted *negatives*. An example rejected, i.e. declared a negative, by any stage is rejected by the cascade. Examples classified as positives are propagated to subsequent stages. To be computationally efficient, the cascade must use simple classifiers in the early stages and complex ones later on. Under the procedure proposed by [91] the cascade designer must first select a number of stages and the target detection/false-positive rate for each stage. A high detection rate is critical, since improperly rejected positives cannot be recovered. The false-positive rate is less critical, since the cascade false-positive

rate can be decreased by addition of stages, although at the price of extra computation. The stages are designed with AdaBoost. The target detection rate is met by manipulating the stage threshold, and the target false-positive rate by increasing the number of weak learners. This frequently leads to an exceedingly complex learning procedure. One difficulty is that the optimal cascade configuration (number of stages and stage target rates) is unknown. We refer to this as the *cascade configuration problem*. While some configurations have evolved by default, e.g. 20 stages, with a detection rate of 99.5% and a false-positive rate of 50%, there is nothing special about these values. This problem is compounded by the fact that, for late stages where negative examples are close to the classification boundary, it may be impossible to meet the target rates. In this case, the designer must backtrack (redesign some of the previous stages). Frequently, various iterations of parameter tuning are needed to reach a satisfactory cascade. Since each iteration requires Boosting over a large set of examples and features, the process can be tedious and time consuming. We refer to this as the *design complexity* problem.

Even when a cascade is successfully designed, the process has no guarantees of optimal classification performance. One problem is that, while computationally efficient, the feature set of [91] lacks discriminant power for many applications. This is the *feature design* problem. This problem is frequently compounded by lack of convergence of AdaBoost. Note that while AdaBoost is consistent [6], there are no guarantees that a classifier with small number of Boosting iterations, e.g. early stages of a cascade, will produce classifiers that generalize well.

We refer to this as the *convergence problem*. This problem is magnified by the mismatch between the AdaBoost risk, which penalizes misses/false-positives equally, and the asymmetry of the target detection and false-positive rates used in practice. Although a stage can always meet the target detection rate by threshold manipulation, the resulting false-positive rate can be strongly sub-optimal [49]. In general, better performance is obtained with *asymmetric* learning algorithms, that optimize the detector explicitly for the target detection rate. This is the *cost-*

sensitive learning problem. Besides classification optimality, the learned cascade is rarely the fastest possible. This is not surprising, since speed is not an explicit variable of the cascade optimization process. While the specification of stage false-positive rates can be used to shuffle computation between stages, there is no way to predict the amount of computation corresponding to a particular rate. This is the *complexity optimization* problem.

III.B.2 Previous solutions

Over the last ten years, significant research has been devoted to all of the above problems.

Feature design: [91] introduced a very efficient set of Haar wavelets. They showed that these features could be extracted, with a few operations, from an integral image (cumulative image sum). While all features in the original Haar set were axis-aligned, [45] developed an extension for 45° rectangles and several authors pursued extensions to other orientations [10, 19, 55]. More recently, [63] have shown how to compute integral images over arbitrary polygonal regions. Beyond these features, integral images can also be used to efficiently compute histograms [65]. This reduces to quantizing the image into a set of *channels* (associated with the histogram bins) and computing an integral image per channel. A computationally efficient version of the HOG descriptor [15] was then developed by [103], and used to design a real-time pedestrian detector cascade. More recently, [60] extended the idea to multiple other channels. Finally, extensions have been developed for more general statistical descriptors, e.g. the covariance features of [87]. While the algorithms proposed in this work support any of these features, we adopt the Haar set of [91]. This is mostly for consistency with the cascade learning literature, where Haar wavelets are predominant.

Design of stage classifiers: A number of enhancements to the stage learning method of [91] have been proposed specifically to address the problems of convergence rate, cost-sensitive learning, and training complexity. One poten-

tial solution to the convergence problem is to adopt recent extensions of AdaBoost, which converge with smaller numbers of weak learners. Since AdaBoost is a greedy feature selection algorithm, the effective number of weak learners can be reduced by using forward-backward feature selection procedures [101] or reweighing weak learners by introduction of sparsity constraints in the optimization [11, 20]. This results in more accurate classification with less weak learners, i.e. a faster classifier. While these algorithms have not been used in the cascade learning literature, several authors have used similar ideas to improve stage classifiers. For example, [44] augmented AdaBoost with a floating search that eliminates weak learners of small contribution to classifier performance. [78, 79] proposed a similar idea, based on linear discriminant analysis (LDA). Moreover, by interpreting the Boosted classifier as a hyperplane in the space of weak learner outputs, several authors have shown how to refine the hyperplane normal so as to maximize class discrimination. Procedures that recompute the weight of each weak learner have been implemented with SVMs [99], variants of LDA [97, 78, 79], and non-linear feature transformations [76]. The hyperplane refinement usually optimizes classification error directly, rather than the exponential loss of AdaBoost, further improving the match between learning objective and classification performance.

Finally, faster convergence is usually possible with different weak learners, e.g. linear SVMs [103] or decision trees of depth two [60], and Boosting algorithms such as realBoost or logitBoost [81, 76, 44, 87].

Beyond classification performance, some attention has been devoted to design complexity. Since the bulk of the learning time is spent on weak learner selection, low-complexity methods have been proposed for this. For example, [97] proposed a forward feature selection method that trades off memory for computational efficiency. An alternative, proposed by [62], is to model Haar wavelet responses as Gaussian variables, whose statistics can be computed efficiently. While speeding up the design of each stage, these methods do not eliminate all aspects of threshold tunning, stage backtracking, etc. It could be argued that this is the worst

component of design complexity, since these operations require manual supervision. A number of enhancements have been proposed in this area. While [91] proposed stage-specific threshold adjustments, [47, 82] formulated threshold adjustments as an a-posteriori optimization of the whole cascade. These methods are hampered by the limited effectiveness of threshold adjustments when stage detectors have poor ROC performance, [49]. Better performance is usually achieved with cost-sensitive extensions of Boosting, which optimize a cost-sensitive risk directly. An early algorithm was proposed by [92] and later extended by [50, 64]. More recently, [49] proposed Bayes consistent cost-sensitive extensions of AdaBoost, logitBoost, and realBoost. These algorithms were shown to substantially improve the false-positive performance of cascades of high detection rate [50]. These could be combined with the method of [9], which devised a predictor of the optimal false positive and detection rate for each stage, from statistics of the previous stages, so as to design a cascade of cost-sensitive stages automatically. An alternative procedure for the joint optimization of all stages of a cascade of known configuration was proposed in [22].

Cascade configuration: Most of the above enhancements are within the framework of [91], i.e. assume a known cascade configuration and sequential stage learning. This is a suboptimal design strategy and the assumed cascade configuration may not be attainable in practice. An alternative is to adopt cascades of *embedded stages* where each stage is the starting point for the design of the next [99, 8, 98, 81, 50, 64]. The main advantage of this structure is that the whole cascade can be designed with a single Boosting run, and adding exit points to a standard classifier ensemble. This also minimizes the convergence rate problems of individual stage design. [81] derived a method for learning embedded stages from Wald's theory of sequential decision making. While attempting to optimize the whole cascade, these approaches do not fully address the configuration problem. Some simply add an exit point per weak learner [50, 98, 81], while others use post-processing [8, 99] or pre-specified detection and false-positive rates [64] to

determine exit point locations. More recently, [42, 66] propose to learn all stages simultaneously, by modeling a cascade as the product, or logical "AND", of its stages.

Overall, despite substantial progress, no method addresses all problems of cascade learning. Since few approaches explicitly optimize the cascade configuration, fewer among these rely on cost-sensitive learning, and no method optimizes detection speed explicitly, cascade learning can require extensive trial and error. This can be quite expensive from a computational point of view and leads to a tedious design procedure, which can produce sub-optimal cascades. In the following sections we propose an alternative framework, which is fully automated and jointly determines 1) the number of cascade stages, 2) the number of weak learners per stage, and 3) the predictor of each stage, by minimizing a Lagrangian risk that is cost-sensitive and explicitly accounts for detection speed.

III.C A Boosting algorithm for the design of classifier cascades

A detailed overview of Boosting and algorithm for designing new Boosting algorithms are presented in Chapter II. In this section we start with a brief review the main concepts of that chapter and use it for training a classifier cascade.

III.C.1 Boosting

A binary classifier $h : \mathcal{X} \rightarrow \{-1, 1\}$ maps an example x into a class label $y(x)$. A learning algorithm seeks the classifier of minimum probability of error, $P_X(h(x) \neq y(x))$, in the space of binary mappings

$$\mathbf{H} = \{h | h : \mathcal{X} \rightarrow \{-1, 1\}\}.$$

Since $\mathbf{H}$ is not convex and $h \in \mathbf{H}$ not necessarily differentiable, this is usually done by restricting the search to mappings of the form

$$h(x) = sign[f(x)],$$

where $f : \mathcal{X} \to \mathbb{R}$, is a *predictor*. The goal is then to learn the optimal $f(x)$ in a set of predictors

$$\mathbf{F} = \{f | f : \mathcal{X} \to \mathbb{R}\}.$$

This is the predictor which minimizes the classification risk, $\mathcal{R}_E : \mathbf{F} \to \mathbb{R}$,

$$\mathcal{R}_E[f] = E_{X,Y}\{L(y(x), f(x))\} \simeq \frac{1}{|S_t|} \sum_i L(y_i, f(x_i)), \tag{III.1}$$

where $L : \{+1, -1\} \times \mathbb{R} \to \mathbb{R}$ is a loss function, and $S_t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a set of training examples x_i of labels y_i.

Boosting algorithms are iterative procedures that learn f as a combination of simple predictors, known as *weak learners,* from a set $\mathbf{G} = \{g_1(x), \ldots, g_n(x)\} \subset \mathbf{F}$. The optimal combination is the solution of

$$\begin{cases} \min_{f(x)} & \mathcal{R}_E[f] \\ s.t: & f(x) \in span(\mathbf{G}). \end{cases} \tag{III.2}$$

Each Boosting iteration reweights the training set and adds the weak learner of lowest weighted error rate to the weak learner ensemble. When $\mathbf{G}$ is rich enough, i.e. contains a predictor with better than chance-level weighted error rate for any distribution over training examples, the Boosted classifier can be arbitrarily close to the minimum probability of error classifier [26]. For most problems of practical interest, $\mathbf{G}$ is an overcomplete set and the solution of (III.2) can have many decompositions in $span(\mathbf{G})$. In this case, sparser decompositions are likely to have better performance, i.e. faster computation and better generalization. Boosting can be interpreted as a greedy forward feature selection procedure to find such sparse solutions.

Although the ideas proposed in this work can be combined with most Boosting algorithms, we limit the discussion to AdaBoost [26]. This is an algorithm

that learns a predictor f by minimizing the risk of (III.1) when L is the negative exponential of the *margin* $y(x)f(x)$

$$L(y(x), f(x)) = e^{-y(x)f(x)}. \tag{III.3}$$

This is known as the exponential loss function [73].

The Boosting algorithms proposed in this section are inspired by the statistical view of AdaBoost, introduced in [53, 28]. Under this view, each iteration of Boosting computes the functional derivatives of the risk along the directions of the weak learners $g_k(x)$, at the current solution $f(x)$. This can be written as

$$\begin{aligned} < \delta\mathcal{R}_E[f], g > &= \frac{d}{d\epsilon}\mathcal{R}_E[f + \epsilon g]\Big|_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left[\frac{d}{d\epsilon}e^{-y_i(f(x_i)+\epsilon g(x_i))}\right]_{\epsilon=0} \\ &= -\frac{1}{|S_t|}\sum_i y_i w_i g(x_i), \end{aligned} \tag{III.4}$$

where $y_i = y(x_i)$ and

$$w_i = w(x_i) = e^{-y_i f(x_i)}, \tag{III.5}$$

is the weight of example x_i. The latter measures how well x_i is classified by the current predictor $f(x)$. The predictor is then updated by selecting the direction (weak learner) of steepest descent

$$\begin{aligned} g^*(x) &= \arg\max_{g\in\mathbf{G}} < -\delta\mathcal{R}_E[f], g > \\ &= \arg\max_{g\in\mathbf{G}} \frac{1}{|S_t|}\sum_i y_i w_i g(x_i), \end{aligned} \tag{III.6}$$

and computing the optimal step size along this direction

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \mathcal{R}_E[f + \alpha g^*]. \tag{III.7}$$

While the optimal step size has a closed form for AdaBoost [26], it can also be found by a line search. The predictor is finally updated according to

$$f(x) = f(x) + \alpha^* g^*(x), \tag{III.8}$$

and the procedure iterated, as summarized in Algorithm 3.

Algorithm 3 adaboost

Input: Training set $S_t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y_i \in \{1, -1\}$ is the class label of example $\mathbf{x}_i$, and number of iterations N.

Initialization: Set $f(x) = 0$.

for $t = 1$ to N **do**

Compute $< -\delta R_E[f], g >$ for all weak learners using (III.4).

Select the best weak learner $g^*(x)$ using (III.6).

Find the optimal step size α^* along $g^*(x)$ using (III.7).

Update $f(x) = f(x) + \alpha^* g^*(x)$.

end for

Output: decision rule: $sign[f(x)]$

III.C.2 Cascade Boosting

In this work, we consider the question of whether Boosting can be extended to learn a detector cascade. We start by introducing some notation. As shown in Figure III.1-a), a classifier cascade is a binary classifier $H(x) \in \mathbf{H}$ implemented as a sequence of classifiers

$$h_i(x) = sgn[f_i(x)] \quad i = 1, \ldots, m, \tag{III.9}$$

where the predictors $f_i(x)$ can be any real functions, e.g. linear combinations of weak learners. The cascade implements the mapping $H : \mathcal{X} \to \{-1, 1\}$ where

$$H(x) = \mathcal{H}^m[h_1, \ldots, h_m](x) \quad = \quad \begin{cases} -1 & \text{if } \exists\, k : h_k(x) < 0 \\ +1 & \text{otherwise,} \end{cases} \tag{III.10}$$

and $\mathcal{H}^m[h_1, \ldots, h_m]$ is a *classifier cascading (CC) operator,* i.e. a functional mapping $\mathcal{H}^m : \mathbf{H}^m \to \mathbf{H}$ of the stage classifiers $h_1, \ldots, h_m$ into the cascaded classifier[1] H. Similarly, it is possible to define a cascade predictor $F(x)$ for $H(x)$, i.e. a mapping $F : \mathcal{X} \to \mathbb{R}$ such that

$$H(x) = sign[F(x)], \tag{III.11}$$

[1]The notation $\mathcal{H}^m[h_1, \ldots, h_m](x)$ should be read as: the value at x of the image of $(h_1, \ldots, h_m)$ under operator $\mathcal{H}^m$.

where

$$F(x) = \mathcal{F}^m[f_1, \ldots, f_m](x), \tag{III.12}$$

and $\mathcal{F}^m : \mathbf{F}^m \to \mathbf{F}$ is a *predictor cascading (PC) operator*, i.e. a functional mapping of the stage predictors $f_1, \ldots, f_m$ into the cascade predictor F. We will study the structure of this operator in Section III.D. For now, we consider the problem of learning a cascade, given that the operator $\mathcal{F}^m$ is known.

To generalize AdaBoost to this problem it suffices to use the predictor $F(x)$ in the exponential loss of (III.3) and solve the optimization problem

$$\begin{cases} \min_{m, f_1, \ldots f_m} & \mathcal{R}_E[F] = \frac{1}{|S_t|} \sum_i e^{-y_i F(x_i)} \\ \\ s.t: & F(x) = \mathcal{F}^m[f_1, \ldots, f_m](x) \\ & \forall i \quad f_i(x) \in span(\mathbf{G}) \end{cases} \tag{III.13}$$

by gradient descent in $span(\mathbf{G})$. The main difference with respect to AdaBoost is that, since any of the cascade stages can be updated, multiple gradient steps are possible per iteration. The directional gradient for updating the predictor of the kth stage is

$$\begin{aligned} < \delta\mathcal{R}_E[F], g >_k &= \frac{d}{d\epsilon}\mathcal{R}_E[\mathcal{F}^m[f_1, \ldots f_k + \epsilon g, \ldots f_m]]\Big|_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left[\frac{d}{d\epsilon} e^{-y_i \mathcal{F}^m[f_1, \ldots f_k + \epsilon g, \ldots f_m](x_i)}\right]_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left\{(-y_i) e^{-y_i \mathcal{F}^m[f_1, \ldots f_m](x_i)} \left[\frac{d}{d\epsilon}\mathcal{F}^m[f_1, \ldots f_k + \epsilon g, \ldots f_m]\right]_{\epsilon=0}(x_i)\right\} \\ &= -\frac{1}{|S_t|}\sum_i y_i w(x_i) b_k(x_i) g(x_i), \end{aligned} \tag{III.14}$$

with

$$w(x_i) = e^{-y_i \mathcal{F}^m[f_1, \ldots f_m](x_i)} = e^{-y_i F(x_i)} \tag{III.15}$$

$$b_k(x_i) = \frac{d}{d\epsilon}\mathcal{F}^m[f_1, \ldots f_k + \epsilon g, \ldots f_m]\Big|_{\epsilon=0}(x_i). \tag{III.16}$$

The optimal descent direction for the kth stage is then

$$\begin{aligned} g_k^* &= \arg\max_{g \in \mathbf{G}} < -\delta \mathcal{R}_E[F], g >_k \\ &= \arg\max_{g \in \mathbf{G}} \frac{1}{|S_t|} \sum_i y_i w(x_i) b_k(x_i) g(x_i), \end{aligned} \tag{III.17}$$

the optimal step size along this direction is

$$\alpha_k^* = \arg\min_{\alpha \in \mathbb{R}} \mathcal{R}_E[\mathcal{F}^m[f_1, .., f_k + \alpha g_k^*, ..f_m]], \tag{III.18}$$

and the optimal stage update is

$$f_k(x) = f_k(x) + \alpha^* g^*(x). \tag{III.19}$$

The steps of (III.15), (III.17), and (III.19) constitute a functional gradient descent algorithm for learning a detector cascade, which generalizes AdaBoost. In particular, the weight of (III.15) generalizes that of (III.5), reweighing examples by how well the current cascade classifies them. The weak learner selection rule of (III.17) differs from that of (III.6) only in that this weight is multiplied by coefficient $b_k(x_i)$. Finally, (III.19) is an additive update, similar to that of (III.8). If the structure of the optimal cascade were known, namely how many stages it contains, these steps could be used to generalize Algorithm 3. It would suffice to, at each iteration t, select the stage k such that g_k^* achieves the smallest risk in (III.18) and update the predictor of that stage. This only has a fundamental difference with respect to AdaBoost: the introduction of the coefficients $b_k(x_i)$ in the weak learner selection. We will see that the procedure above can also be extended into an algorithm that learns the cascade configuration. Since these extensions depend on the PC operator $\mathcal{F}$ of (III.12), we start by studying its structure.

III.D The structure of cascade predictors

In this section, we derive a general form for $\mathcal{F}^m$. We show that any cascade is compatible with an infinite set of predictors and that these can be computed recursively. This turns out to be important for the efficient implementation

of the learning algorithm of the previous section. We next consider a class of PC operators synthesized by recursive application of a two-stage PC operator, denoted the *generator* of the cascade. Two generators are then proposed, from which we derive two new cascade predictor families that generalize the two most common cascade structures in the literature.

III.D.1 Cascade predictors

From (III.10), a classifier cascade implements the logical-AND of the outputs of its stage classifiers, i.e. $\mathcal{H}^m$ is the pointwise logical-AND of $h_1, \ldots, h_m$,

$$\mathcal{H}^m[h_1, \ldots, h_m](x) = h_1(x) \wedge \ldots \wedge h_m(x), \tag{III.20}$$

where $\wedge$ is the logical-AND operation. Since, from (III.10)-(III.12),

$$\mathcal{H}^m[h_1, \ldots, h_m](x) = sgn[\mathcal{F}^m[f_1, \ldots, f_m](x)], \tag{III.21}$$

it follows from (III.9) that

$$sign[\mathcal{F}^m[f_1, \ldots, f_m](x)] = sgn[f_1(x)] \wedge \ldots \wedge sgn[f_m(x)]. \tag{III.22}$$

This holds if and only if

$$\begin{cases} \mathcal{F}^m[f_1, \ldots, f_m](x) < 0 & \text{if } \exists\, k : f_k(x) < 0 \\ \mathcal{F}^m[f_1, \ldots, f_m](x) > 0 & \text{otherwise.} \end{cases} \tag{III.23}$$

Since (III.22) holds for any operator with this property, any such $\mathcal{F}^m$ is denoted a pointwise *soft-AND* of its arguments. In summary, while a cascade implements the logical-AND of its stage decisions, the cascade predictor implements a soft-AND of the corresponding stage predictions. Note that there is an infinite number of soft-AND operators which will implement the same logical-AND operator, once thresholded according to (III.21). This makes the set of cascade predictors much richer than that of cascades.

III.D.2 Recursive implementation

For any m, it follows from (III.20) and the associative property of the logical-AND that

$$\mathcal{H}^m[h_1,\ldots,h_m] \;=\; \begin{cases} \mathcal{H}^2[h_1,h_2], & m=2 \\ \mathcal{H}^2\left[h_1,\mathcal{H}^{m-1}[h_2,\ldots,h_m]\right] & m>2. \end{cases} \tag{III.24}$$

A similar decomposition holds for the soft-AND operator of (III.23), since

$$sgn\left[\mathcal{F}^m[f_1,\ldots,f_m](x)\right] \;=\; \begin{cases} sgn\left[\mathcal{F}^2[f_1,f_2](x)\right], & m=2 \\ sgn\left[\mathcal{F}^2\left[f_1,\mathcal{F}^{m-1}[f_2,\ldots,f_m]\right](x)\right] & m>2. \end{cases} \tag{III.25}$$

The main difference between the two recursions is that, while there is only one logical-AND $\mathcal{H}^2[f_1,f_2]$, an infinite set of soft-AND operators $\mathcal{F}^2[f_1,f_2]$ can be used in (III.25). In fact, it is possible to use a different operator $\mathcal{F}^2$ at each level of the recursion, i.e. replace $\mathcal{F}^2$ by $\mathcal{F}^2_m$, to synthesize all possible sequences of soft-AND operators $\{\mathcal{F}^i\}_{i=2}^m$ for which the left-hand side of (III.25) is the same. For simplicity, we only consider soft-AND operators of the form of (III.25) in this work.

The recursions above make it possible to derive a recursive decomposition of both the cascade and the sign of its predictor. In particular, defining

$$H_k(x) \;=\; \mathcal{H}^{m-k+1}[h_k,\ldots,h_m](x),$$

(III.24) leads to the *cascade recursion*

$$H_k(x) \;=\; \begin{cases} h_m(x), & k=m \\ \mathcal{H}^2\left[h_k,H_{k+1}\right](x), & 1\le k<m, \end{cases}$$

with $H_1(x)=H(x)$. Similarly, for any sequence of soft-AND operators $\{\mathcal{F}^i\}_{i=2}^m$ compatible with (III.25), defining

$$F_k(x) \;=\; \mathcal{F}^{m-k+1}[f_k,\ldots,f_m](x),$$

leads to the *predictor recursion*

$$sgn[F_k(x)] \;=\; \begin{cases} sgn[f_m(x)], & k=m \\ sgn\left[\mathcal{F}^2\left[f_k,F_{k+1}\right](x)\right], & 1\le k<m, \end{cases} \tag{III.26}$$

with $sgn[F_1(x)] = sgn[F(x)]$. Simplifying (III.26), in the remainder of this work we consider predictors of the form

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ \mathcal{F}^2\left[f_k, F_{k+1}\right](x), & 1 \leq k < m. \end{cases} \tag{III.27}$$

Since the core of this recursion is the two-stage predictor

$$\mathcal{G}[f_1, f_2] = \mathcal{F}^2[f_1, f_2], \tag{III.28}$$

this is denoted the *generator* of the cascade. We will show that the two most popular cascade architectures can be derived from two such generators. For each, we will then derive the cascade predictors $F_k(x)$, the cascade Boosting weights $w(x_i)$ of (III.15), and the coefficients $b_k(x_i)$ of (III.16). We start by defining some notation to be used in these derivations.

III.D.3 Some definitions

Some of the computations of the following sections involve derivatives of Heaviside step functions $u(.)$, which are not differentiable. As is common in the neural network literature, this problem is addressed with the sigmoidal approximation

$$u(x) \approx \sigma(x) = \frac{1}{2}(\tanh(\mu x) + 1). \tag{III.29}$$

The parameter μ controls the sharpness of the sigmoid. This approximation is well known to have the symmetry $\sigma(-x) = 1 - \sigma(x)$ and derivative $\sigma'(x) = 2\mu\sigma(x)\sigma(-x)$. We also introduce the sequence of *cascaded Heaviside functions*

$$\gamma_k(x) = \begin{cases} 1, & k = 1 \\ \prod_{j<k} u[f_j(x)], & k > 1, \end{cases} \tag{III.30}$$

and *cascaded rectification functions*

$$\xi_k(x) = \begin{cases} 1, & k = 1 \\ \prod_{j<k} f_j(x) u[f_j(x)], & k > 1, \end{cases} \tag{III.31}$$

where $u(.)$ is the Heaviside step. The former generalize the Heaviside step, in the sense that $\gamma_k(x) = 1$ if $f_j(x) > 0$ for all $j < k$ and $\gamma_k(x) = 0$ otherwise. The latter generalize the half-wave rectifier, in the sense that $\gamma_k(x) = \prod_{j<k} f_j(x)$ if $f_j(x) > 0$ for all $j < k$ and $\gamma_k(x) = 0$ otherwise.

III.D.4 Last stage cascades

The first family of cascade predictors that we consider is derived from the generator

$$\begin{aligned} \mathcal{G}_1[f_1, f_2](x) &= f_1(x)u[-f_1(x)] + u[f_1(x)]f_2(x) \\ &= \begin{cases} f_1(x) & \text{if } \ f_1(x) < 0 \\ f_2(x) & \text{if } \ f_1(x) \geq 0, \end{cases} \end{aligned} \tag{III.32}$$

Using (III.27), the associated predictor recursion is

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ f_k(x)u[-f_k(x)] + u[f_k(x)]F_{k+1}(x), & 1 \leq k < m. \end{cases} \tag{III.33}$$

The kth stage of the associated cascade passes example x to stage $k+1$ if $f_k(x) \geq 0$. Otherwise, the example is rejected with prediction $f_k(x)$. Hence,

$$\mathcal{F}^m[f_1, \ldots, f_m](x) = \begin{cases} f_j(x) & \text{if } \ f_j(x) < 0 \text{ and} \\ & f_i(x) \geq 0 \ \ i = 1, \ldots, j-1 \\ f_m(x) & \text{if } \ f_i(x) \geq 0 \ \ i = 1 \ldots, m-1, \end{cases}$$

i.e. the cascade prediction is that of the last stage visited by the example. For this reason, the cascade is denoted a *last-stage cascade*.

This property makes it trivial to compute the weights $w(x)$ of the cascade Boosting algorithm, using (III.15). It suffices to evaluate

$$w(x_i) = e^{-y_i f_{j^*}(x_i)}, \tag{III.34}$$

where j^* is the smallest k for which $f_k(x_i)$ is negative and $j^* = m$ if there is no such k. The computation of $b_k(x)$ with (III.16) requires a differentiable form of

$\mathcal{F}^m[f_1, \ldots, f_m]$ with respect to f_k. This can be obtained by recursive application of (III.33), since

$$
\begin{aligned}
&\mathcal{F}^m[f_1, \ldots, f_m](x) = F_1(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]F_2(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]\left\{f_2(x)u[-f_2(x)] + u[f_2(x)]F_3(x)\right\} \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\prod_{j<i} u[f_j(x)]\right] + F_k(x)\prod_{j<k} u[f_j(x)] \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + F_k(x)\gamma_k(x) \qquad k = 1 \ldots m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)\left\{f_k(x)u[-f_k(x)] + u[f_k(x)]F_{k+1}(x)\right\} \qquad k < m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)\left\{f_k(x) + u[f_k(x)][F_{k+1}(x) - f_k(x)]\right\} \\
&\approx \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)f_k(x) + \gamma_k(x)\sigma[f_k(x)][F_{k+1}(x) - f_k(x)] \qquad \text{(III.35)}
\end{aligned}
$$

where $\gamma_k(x)$ are the cascaded Heaviside functions of (III.30) and we used the differentiable approximation of (III.29) in (III.35). Note that neither the first term on the right-hand side of (III.35) nor γ_k or F_{k+1} depend on f_k. It follows from (III.16) that

$$
b_k(x) = \begin{cases} \gamma_k(x), & k = m \\ \gamma_k(x)\{1 + 2\mu\sigma[f_k(x)][F_{k+1}(x) - f_k(x)]\}\sigma[-f_k(x)] & 1 \le k < m, \end{cases} \qquad \text{(III.36)}
$$

where $\sigma(.)$ is defined in (III.29). Given x, all these quantities can be computed with a sequence of a forward, a backward, and a forward pass through the cascade. The initial forward pass computes $\gamma_k(x)$ for all k according to (III.30). The backward pass then computes $F_{k+1}(x)$ using (III.33). The final forward pass computes the weight $w(x)$ and coefficients $b_k(x)$ using (III.34) and (III.36). These steps are summarized in Algorithm 4. The procedure resembles the back-propagation algorithm for neural network training [67].

Algorithm 4 Last-stage cascade

Input: Training example (x, y), stage predictors $f_k(x), k = 1, \ldots, m$, sigmoid parameter μ.

Evaluation:

Set $\gamma_1(x) = 1$.

for $k = 2$ to m **do**

  Set $\gamma_k(x) = \gamma_{k-1}(x)u[f_k(x)]$.

end for

Set $F_m(x) = f_m(x)$.

for $k = m - 1$ to 1 **do**

  Set $F_k(x) = f_k(x)u[-f_k(x)] + u[f_k(x)]F_{k+1}(x)$.

end for

Learning:

Set $w(x) = e^{-yf_{j^*}(x)}$ where j^* is the smallest k for which $f_k(x_i) < 0$ and $j^* = m$ if there is no such k.

for $k = 1$ to $m - 1$ **do**

  Set $b_k(x) = \gamma_k(x)\{1 + 2\mu\sigma[f_k(x)][F_{k+1}(x) - f_k(x)]\}\sigma[-f_k(x)]$.

end for

Set $b_k(x) = \gamma_m(x)$.

Output: $w(x), \{F_k(x), b_k(x)\}_{k=1}^m$.

III.D.5 Multiplicative cascades

The second family of cascade predictors has generator

$$\begin{aligned} \mathcal{G}_2[f_1, f_2](x) &= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)f_2(x) \\ &= \begin{cases} f_1(x) & \text{if } f_1(x) < 0 \\ f_1(x)f_2(x) & \text{if } f_1(x) \geq 0. \end{cases} \end{aligned} \tag{III.37}$$

Using (III.27), the associated predictor recursion is

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ f_k(x)u[-f_k(x)] + u[f_k(x)]f_k(x)F_{k+1}(x), & 1 \leq k < m \end{cases} \tag{III.38}$$

and

$$\mathcal{F}^m[f_1,\ldots,f_m](x) = \begin{cases} \prod_{i\le j} f_i(x) & \text{if } \; f_j(x) < 0 \text{ and} \\ & f_i(x) \ge 0 \;\; i = 1..j-1 \\ \prod_{i=1}^{m} f_i(x) & \text{if } \; f_i(x) \ge 0 \;\; i = 1..m-1. \end{cases}$$

Hence, the cascade predictor is the product of all stage predictions up-to and including that where the example is rejected. This is denoted a *multiplicative cascade*.

The weights $w(x)$ of the cascade Boosting algorithm are

$$w(x_i) = e^{-y_i \prod_{k\le j^*} f_k(x_i)},$$

where j^* is the smallest k for which $f_k(x_i)$ is negative and $j^* = m$ if there is no such k. The computation of $b_k(x)$ with (III.16) requires a differentiable form of $\mathcal{F}^m[f_1,\ldots,f_m]$ with respect to f_k. This can be obtained by recursive application of (III.38), since

$$\begin{aligned}
&\mathcal{F}^m[f_1,\ldots,f_m](x) = F_1(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)F_2(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)\{f_2(x)u[-f_2(x)] + u[f_2(x)]f_2(x)F_3(x)\} \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\prod_{j<i} f_j(x)u[f_j(x)]\right] + F_k(x)\prod_{j<k} f_j(x)u[f_j(x)] \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + F_k(x)\xi_k(x) \qquad k = 1\ldots m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)\left\{f_k(x)u[-f_k(x)] + u[f_k(x)]f_k(x)F_{k+1}(x)\right\} \qquad k < m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)f_k(x)\left\{1 + u[f_k(x)][F_{k+1}(x) - 1]\right\} \\
&\approx \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)f_k(x)\left\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\right\} \qquad \text{(III.39)}
\end{aligned}$$

where $\xi_i(x)$ are the rectification functions of (III.31) and we used (III.38) and the differentiable approximation of (III.29) in (III.39). Since neither the first term on

Algorithm 5 multiplicative cascade

Input: Training example (x, y), stage predictors $f_k(x), k = 1, \ldots, m$, sigmoid parameter μ.

Evaluation:

Set $\xi_1 = 1$.

for $k = 2$ to m **do**

  Set $\xi_k(x) = \xi_{k-1}(x) f_k(x) u[f_k(x)]$.

end for

Set $F_m(x) = f_m(x)$.

for $k = m - 1$ to 1 **do**

  Set $F_k(x) = f_k(x) u[-f_k(x)] + u[f_k(x)] f_k(x)\ F_{k+1}(x)$.

end for

Learning:

Set $w(x) = e^{-y \prod_{k \leq j^*} f_k(x)}$ where j^* is the smallest k for which $f_k(x_i) < 0$ and $j^* = m$ if there is no such k.

for $k = 1$ to $m - 1$ **do**

  Set $b_k(x) = \xi_k(x)\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\}\{1 + 2\mu f_k(x)\sigma[-f_k(x)]\}$.

end for

Set $b_m(x) = \xi_m(x)$.

Output: $w(x), \{F_k(x), b_k(x)\}_{k=1}^m$.

the right hand side, ξ_k, or F_{k+1} depend on f_k, it follows from (III.16) that

$$b_k(x) = \begin{cases} \xi_m(x), & k = m \\ \xi_k(x)\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\}\{1 + 2\mu f_k(x)\sigma[-f_k(x)]\} & 1 \leq k < m, \end{cases} \tag{III.40}$$

where $\sigma(.)$ is defined in (III.29). Again, these coefficients can be computed with a forward, a backward, and a forward pass through the cascade, which resembles back-propagation, as summarized in Algorithm 5.

III.E Learning the cascade configuration

Given a cascade configuration, Algorithms 4 or 5, could be combined with the algorithm of section III.C.2 to extend AdaBoost to the design of last-stage or multiplicative cascades, respectively. However, the cascade configuration is usually not known and must be learned. This consists of determining the number of cascade stages and the number of weak learners per stage.

III.E.1 Complexity Loss

We start by assuming that the number of cascade stages is known and concentrate on the composition of these stages. So far, we have proposed to simply update, at each Boosting iteration, the stage k with the weak learner g_k^* that achieves the smallest risk in (III.18). While this will produce cascades with good detection accuracy, there is no incentive for the cascade configuration to be efficient, i.e. achieve an optimal trade-off between detection accuracy and classification speed. To guarantee such a trade-off it is necessary to search for the most accurate detector under a complexity constraint. This can be done by minimizing the Lagrangian

$$\mathcal{L}[F] \;=\; \mathcal{R}_E[F] + \eta \mathcal{R}_C[F], \tag{III.41}$$

where $F(x)$ and $R_E[F]$ are the cascade predictor and classification risk of (III.13), respectively,

$$\mathcal{R}_C[F] \;=\; E_{X|Y}\{L_C(F,x)|y(x)=-1\} \simeq \frac{1}{|S_t^-|}\sum_{x_i \in S_t^-} L_C(F,x_i),$$

is a complexity risk and η a Lagrange multiplier that determines the trade-off between accuracy and computational complexity. $\mathcal{R}_C[F]$ is the empirical average of a computational loss $L_C(F,x)$, which reflects the number of machine operations required to evaluate $F(x) = \mathcal{F}^m[f_1,\ldots,f_m](x)$, over the set S_t^- of negatives in S_t. The restriction to negative examples is not necessary but common in the

classifier cascade literature, where computational complexity is usually defined as the average computation required to reject negative examples. This is mostly because positives are rare and contribute little to the overall computation.

As is the case for the classification risk, where the loss of (III.3) is an upper bound on the margin and not the margin itself, the computational loss $L_C[F]$ is a surrogate for the computational cost $\mathcal{C}(F, x)$ of evaluating the cascade prediction $F(x)$ for example x. Using the predictor recursions of Section III.D.2, this cost can itself be computed recursively. Since, by definition of cascade, example x is either rejected by the predictor f_k of stage k or passed to the remaining stages,

$$\mathcal{C}(F_k, x) = \begin{cases} \Omega(f_k) + u[f_k(x)]\mathcal{C}(F_{k+1}, x), & k < m \\ \Omega(f_m), & k = m, \end{cases} \tag{III.42}$$

where $F_k(x)$ is as defined in (III.27) and $\Omega(f_k)$ is the computational cost of evaluating stage k. Defining $\mathcal{C}(F_{m+1}, x) = 0$, it follows that

$$\begin{aligned} \mathcal{C}(F, x) &= \Omega(f_1) + u[f_1(x)]\mathcal{C}(F_2, x) \\ &= \Omega(f_1) + u[f_1(x)][\Omega(f_2) + u[f_2(x)]\mathcal{C}(F_3, x)] \\ &= \left[\sum_{i=1}^{k-1} \Omega(f_i) \prod_{j<i} u[f_j(x)]\right] + \mathcal{C}(F_k, x) \prod_{j<k} u[f_j(x)] \\ &= \left[\sum_{i=1}^{k-1} \Omega(f_i)\gamma_i(x)\right] + \Omega(f_k)\gamma_k(x) + u[f_k(x)]\mathcal{C}(F_{k+1}, x)\gamma_k(x) \\ &= \delta_k(x) + \Omega(f_k)\gamma_k(x) + \theta_k(x)u[f_k(x)], \end{aligned} \tag{III.43}$$

where $\gamma_i(x)$ are the cascaded Heaviside functions of (III.30) and

$$\begin{aligned} \delta_k(x) &= \sum_{i=1}^{k-1} \Omega(f_i)\gamma_i(x), \\ \theta_k(x) &= \mathcal{C}(F_{k+1}, x)\gamma_k(x). \end{aligned} \tag{III.44}$$

This relates the cascade complexity to the complexity of the kth stage, $\Omega(f_k)$. The surrogate computational loss $L_C[F, x]$ is inspired by the surrogate classification loss of AdaBoost, which upper bounds the zero-one loss $u[-yf(x)]$ by the exponential

$e^{-yf(x)}$. Using the bound $u[f(x)] \leq e^{f(x)}$ on (III.43) leads to

$$L_C[F, x] \;=\; \delta_k(x) + \Omega(f_k)\gamma_k(x) + \theta_k(x)e^{f_k(x)},$$

and the computational risk

$$R_C[F] \;=\; \frac{1}{|S_t^-|} \sum_{x_i \in S_t^-} \delta_k(x_i) + \Omega(f_k)\gamma_k(x_i) + \theta_k(x_i)e^{f_k(x_i)}. \qquad \text{(III.45)}$$

To evaluate this risk, it remains to determine the computational cost $\Omega(f_k)$ of the predictor of the kth cascade stage. Since $f_k(x) = \sum_l \alpha_l g_l(x), g_l \in \mathbf{G}$, is a linear combination of weak learners, we define

$$\Omega(f_k) = \sum_l \Omega(g_l). \qquad \text{(III.46)}$$

Let $\mathcal{W}(f_k) \subset \mathbf{G}$ be the set of weak learners, g_l, that appear in (III.46). In this work, we restrict our attention to the case where all g_l have the same complexity and $\Omega(f_k)$ is proportional to $|\mathcal{W}(f_k)|$. This is the most common scenario in computer vision problems, such as face detection, where all weak learners are thresholded Haar wavelet features [91] and have similar computational cost. We will, however, account for the fact that there is no cost in the repeated evaluation of a weak learner. For this, $\mathcal{W}(f_k)$ is split into two sets. The first, $\mathcal{O}(f_k)$, contains the weak learners used in some earlier cascade stage $f_j, j \leq k$. Since the outputs of these learners can be kept in memory, they require minimal computation (multiplication by α_l and addition to cumulative sum). The second is the set $\mathcal{N}(f_k)$ of weak learners unused in prior stages. The computational cost of f_k is then

$$\Omega(f_k) = |\mathcal{N}(f_k)| + \lambda|\mathcal{O}(f_k)|, \qquad \text{(III.47)}$$

where $\lambda < 1$ is the ratio of computation required to evaluate a used vs. new weak learner. This implies that when updating the kth stage predictor

$$\Omega(f_k + \epsilon g) = \Omega(f_k) + \rho(g, f_k),$$

with

$$\rho(g, f_k) = \begin{cases} \lambda & \text{if } g \in \mathcal{O}(f_k) \\ 1 & \text{if } g \in \mathcal{N}(f_k). \end{cases} \qquad \text{(III.48)}$$

III.E.2 Boosting with complexity constraints

Given the computational risk of (III.45), it is possible to derive a Boosting algorithm that accounts for cascade complexity. We start by deriving the steepest descent direction of the Lagrangian of (III.41), with respect to stage k

$$\begin{aligned} < -\delta\mathcal{L}[F], g >_k &= < -\delta \left(R_E[F] + \eta R_C[F] \right), g >_k \\ &= < -\delta R_E[F], g >_k + \eta < -\delta R_C[F], g >_k . \end{aligned}$$

The first term is given by (III.14), the second requires the descent direction with respect to the complexity risk $R_C[F]$. Using (III.45),

$$\begin{aligned} < \delta R_C[F], g >_k &= \frac{d}{d\epsilon} R_C(\mathcal{F}^m[f_1, .., f_k + \epsilon g, ..f_m]) \Big|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \frac{d}{d\epsilon} L_C \left[\mathcal{F}^m[f_1, .., f_k + \epsilon g, ..f_m], x_i \right] \Big|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \frac{d}{d\epsilon} \left[\delta_k(x_i) + [\Omega(f_k) + \rho(f_k, g)]\gamma_k(x_i) + \theta_k(x_i) e^{f_k(x_i) + \epsilon g(x_i)} \right] \Big|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \psi_k(x_i) \theta_k(x_i) g(x_i), \end{aligned} \tag{III.49}$$

where $y_i^s = I(y_i = -1)$, $I(x)$ is the indicator function, $\theta_k(x_i)$ as in (III.44) and

$$\psi_k(x_i) = e^{f_k(x_i)}. \tag{III.50}$$

Finally, combining (III.14) and (III.49),

$$< -\delta\mathcal{L}[F], g >_k = \sum_i \left(\frac{y_i w(x_i) b_k(x_i)}{|S_t|} - \eta \frac{y_i^s \psi_k(x_i) \theta_k(x_i)}{|S_t^-|} \right) g(x_i), \tag{III.51}$$

where $w(x_i) = e^{-y_i F(x_i)}$ and $b_k(x_i)$ is given by (III.36) for last-stage and by (III.40) for multiplicative cascades.

It should be noted that, although (III.51) does not depend on $\rho(f_k, g)$, the complexity of the optimal weak learner g^* affects the computational risk in (III.45) and thus the magnitude of the steepest descent step. To account for this, we find the best update for f_k in two steps. The first step searches for the best update

Algorithm 6 BestStageUpdate

Input: Training set S_t, trade-off parameter η, cascade $[f_1, \ldots, f_m]$, index k of the stage to update, sigmoid parameter μ.

for each pair (x_i, y_i) in S_t **do**

Compute $w(x_i)$, $b_k(x_i)$, $F_k(x_i)$ e.g. using Algorithm 4 for last-stage or Algorithm 5 for multiplicative cascades.

Compute $\theta_k(x_i)$, $\psi_k(x_i)$ with (III.44) and (III.50).

end for

Find the best update $(\alpha_k^*, g_k^*(x))$ for the kth stage using (III.51)-(III.54).

Output: $\alpha_k^*, g_k^*(x)$

within $\mathcal{O}(f_k)$ and $\mathcal{N}(f_k)$

$$g_{1,k}^* = \arg\max_{g \in \mathcal{O}(f_k)} < -\delta\mathcal{L}[F], g >_k \tag{III.52}$$

$$g_{2,k}^* = \arg\max_{g \in \mathcal{N}(f_k)} < -\delta\mathcal{L}[F], g >_k, \tag{III.53}$$

and computes the corresponding optimal steps sizes

$$\alpha_{j,k}^* = \arg\min_{\alpha \in \mathbb{R}} \mathcal{L}[\mathcal{F}^m[f_1, ..f_k + \alpha g_{j,k}, ..f_m]], \tag{III.54}$$

for $j = 1, 2$. The second step chooses the update that most reduces $\mathcal{L}[F]$ as the best update for the kth stage. The overall procedure is summarized in Algorithm 6. Using this procedure to cycle through all cascade stage updates within each iteration of the algorithm of section III.C.2 and selecting the one that most reduces $\mathcal{L}[F]$ produces an extension of AdaBoost for cascade learning that optimizes the trade-off between detection accuracy and complexity.

III.E.3 Growing a detector cascade

So far, we have assumed that the number of cascade stages is known. Since this is usually not the case, there is a need for a procedure that learns this component of the cascade configuration. In this work, we adopt a greedy strategy, where cascade stages are added by the Boosting algorithm itself, whenever this

leads to a reduction of the risk. It is assumed that a new stage, or predictor g, can only be added at the end of the existing cascade, i.e. transforming a m-stage predictor $\mathcal{F}^m[f_1, \ldots, f_m](x)$ into a $m+1$-stage predictor $\mathcal{F}^{m+1}[f_1, \ldots, f_m, g](x)$. This is consistent with current cascade design practices, where stages are appended to the cascade when certain heuristics are met.

The challenge of a risk-minimizing formulation of this process is to pose the addition of a new stage as a possible gradient step. Recall that, at each iteration of a gradient descent algorithm, the current solution, v^t, is updated by

$$v^{t+1} = v^t + \alpha \overline{v},$$

where $\overline{v}$ is the gradient update and α is step size found by a line search. An immediate consequence is that, if no update is taken in an iteration, i.e. $\alpha = 0$ or $\overline{v} = 0$, the value of the objective function should remain unaltered. For the proposed cascade Boosting algorithms this condition is not trivial to guarantee when a new stage is appended to the current cascade. For example, choosing $g(x) = 0$ may change the current solution since, in general,

$$\mathcal{F}^{m+1}[f_1, \ldots, f_m, 0](x) \neq \mathcal{F}^m[f_1, \ldots, f_m](x).$$

To address this problem, we introduce the concept of *neutral predictors*. A stage predictor $n(x) : \mathcal{X} \to \mathbb{R}$ is neutral for a cascade of predictor $\mathcal{F}^m[f_1, \ldots, f_m]$ if and only if

$$\mathcal{F}^{m+1}[f_1, \ldots, f_m, n](x) = \mathcal{F}^m[f_1, \ldots, f_m](x). \tag{III.55}$$

If such a neutral predictor exists, then it is possible to grow a cascade by defining the new stage as

$$f_{m+1}(x) = n(x) + g(x),$$

where $g(x)$ is the best update found by gradient descent. In this case, it follows from (III.55) that a step of $g(x) = 0$ will leave the cascade risk unaltered. Given a cascade generator, a predictor n that satisfies (III.55) can usually be found

with (III.28), i.e. it suffices that n satisfies

$$f_m(x) = \mathcal{G}[f_m, n](x), \tag{III.56}$$

where $\mathcal{G}$ is the generator that defines the PC operator $\mathcal{F}^m$. For example, from (III.32), the neutral predictor of a last-stage cascade must satisfy

$$f_m(x) = f_m(x)u[-f_m(x)] + u[f_m(x)]n(x),$$

a condition met by

$$n(x) = f_m(x). \tag{III.57}$$

Similarly, from (III.37), the neutral predictor of a multiplicative cascade must satisfy

$$f_m(x) = f_m(x)u[-f_m(x)] + u[f_m(x)]f_m(x)n(x),$$

which is met by

$$n(x) = 1. \tag{III.58}$$

These neutral predictors are also computationally efficient. In fact, (III.57) and (III.58) add no computation to the evaluation of predictor $f_{m+1}(x)$, i.e. to the computation of $g(x)$ itself. This is obvious for (III.58) which is a constant, and follows from the fact that $f_m(x)$ has already been computed in stage m for (III.57). This computation can simply be reused at stage $m+1$ with no additional cost. Hence, for both models

$$\mathcal{C}(\mathcal{F}^{m+1}[f_1, \ldots, f_m, n], x) = \mathcal{C}(\mathcal{F}^m[f_1, \ldots, f_m], x),$$

and

$$\mathcal{L}[\mathcal{F}^{m+1}[f_1, \ldots, f_m, n]] = \mathcal{L}[\mathcal{F}^m[f_1, \ldots, f_m]].$$

In summary, the addition of stages *does not* require special treatment in the proposed cascade learning framework. It suffices to append a neutral predictor to the cascade and find the best update for this new stage. If this reduces the objective function of (III.41) further than updating other stages, the new stage

is *automatically* created and appended to the cascade. In this way, the cascade grows organically, as a side effect of the risk optimization, and there is no need for heuristics.

III.F The FCBoost cascade learning algorithm

In this section, we combine the contributions from the previous sections into the *Fast Cascade Boosting* (FCBoost) algorithm, discuss its connections with the previous literature and some interesting properties.

III.F.1 FCBoost

FCBoost is initialized with a neutral predictor. At each iteration, it finds the best update $g_k^*(x)$ for each of the cascade stages and the best stage to add at the end of the cascade. It then selects the stage k^* whose update $g_{k^*}^*(x)$ most reduces the Lagrangian $\mathcal{L}[F]$. If k^* is the newly added stage, a new stage is created and appended to the cascade. The procedure is summarized in Algorithm 7. Note that the only parameters are the multiplier η of (III.41), which encodes the relative importance of cascade speed vs. accuracy for the cascade designer, and the sigmoid parameter μ that controls the smoothness of the Heaviside approximation. In our implementation we always use $\mu = 5$. Given these parameters, FCBoost will automatically determine both the cascade configuration (number of stages and number of weak learners per stage) and the predictor of each stage, so as to optimize the trade-off between detection accuracy and complexity which is specified by η.

III.F.2 Connections to the previous cascade learning literature

FCBoost supports a large variety of cascade structures. The cascade structure is defined by the generator $\mathcal{G}$ of (III.28), since this determines the neutral predictor $n(x)$, according to (III.56), and consequently how the cascade grows as Boosting progresses.

Algorithm 7 FCBoost

Input: Training set $S = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_n, y_n)\}$, trade-off parameter η, sigmoid parameter μ, and number of iterations N.

Initialization: Set $m = 0$ and $f_1(x) = n(x)$, e.g. using (III.57) for last-stage and (III.58) for multiplicative cascade.

for $t = 1$ to N **do**

  for $k = 1$ to m **do**

    $(\alpha_k^*, g_k^*) = \text{BestStageUpdate}(S, \eta, [f_1, ...f_m], k, \mu)$.

  end for

  $(\alpha_{m+1}^*, g_{m+1}^*) = \text{BestStageUpdate}(S, \eta, [f_1, ...f_{m+1}], m+1, \mu)$.

  for $k = 1$ to m **do**

    Set $\hat{\mathcal{L}}(k) = \mathcal{L}\left[\mathcal{F}^m(f_1, .., f_k + \alpha_k^* g_k^*, .., f_m)\right]$ using (III.41).

  end for

  Set $\hat{\mathcal{L}}(m+1) = \mathcal{L}\left[\mathcal{F}^{m+1}(f_1, .., f_m, f_{m+1} + \alpha_{m+1}^* g_{m+1}^*(x))\right]$ using (III.41).

  Find $k^* = \arg\min_{k \in \{1,\ldots,m+1\}} \hat{\mathcal{L}}(k)$.

  Set $f_{k^*} = f_{k^*} + \alpha_{k^*}^* g_{k^*}^*$.

  if $k^* = m+1$ **then**

    Set $m = m + 1$.

    Set $f_{m+1}(x) = n(x)$.

  end if

end for

Output: decision rule: $sgn[\mathcal{F}^m(f_1, \ldots, f_m)]$.

The two cascade predictors used in this work, last-stage and multiplicative, cover the two predominant cascade structures in the literature. The first, introduced in [91], is the *independent stage* (IS) structure. In this structure stage predictors are designed independently[2], in the sense that the learning of f_k starts from an empty predictor which is irrespective of the composition of the previous

[2]Note that the predictors are always *statistically* dependent, since the role of h_{i+1} is to classify examples not rejected by h_i.

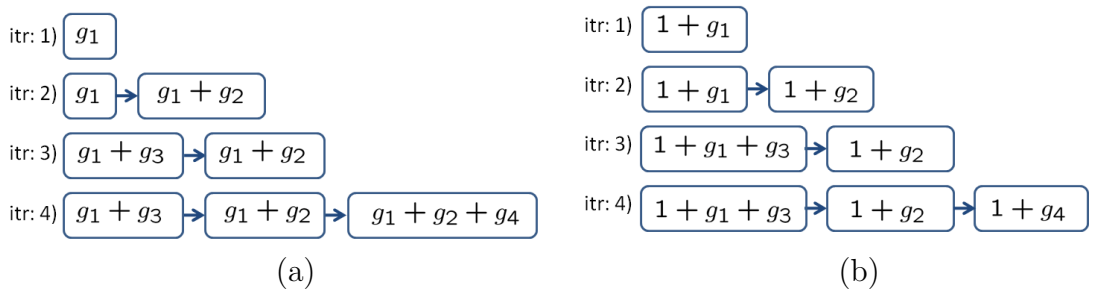


Figure III.2: Illustration of the different configurations produced by identical steps of (a) last-stage and (b) multiplicative cascade learning.

stages, $f_j, j < k$. The second structure is the *embedded stage* (ES) structure of [99] where predictors of consecutive stages are related by

$$f_{k+1}(x) = f_k(x) + \mathbf{w}(x),$$

and $\mathbf{w}(x)$ is a single or linear combination of weak learners. Under this structure, each stage predictor contains the predictor of the previous stage, which is augmented with some weak learners.

The connection between these structures and the models proposed in this dissertation can be understood by considering the neutral predictors of the latter. For multiplicative cascades, it follows from (III.58) that

$$f_{m+1}(x) = 1 + \alpha g(x),$$

and there is no dependence between consecutive stages. Hence, multiplicative cascades have the IS structure. For last stage cascades, it follows from (III.57) that

$$f_{m+1}(x) = f_m(x) + \alpha g(x).$$

If FCBoost always updates the last two stages, this produces a cascade with the ES structure. Since FCBoost is free to update any stage, it can produce more general cascades, i.e. a superset of the set of cascades with the ES structure.

It is interesting that two predictors with the very similar generators of (III.32) and (III.37) produce very different cascade structures. This is illustrated in

Figure III.2, where we consider the cascades resulting from the following sequence of operations:

- **iteration 1:** start form an empty classifier, create first stage.
- **iteration 2:** add a new stage.
- **iteration 3:** update first stage.
- **iteration 4:** add a new stage.

Note that while the last-stage cascade of a) has substantial weak learner sharing across stages, this is not true for the multiplicative cascade of b), which is similar to the cascades of [91].

III.F.3 Properties

Beyond these connections to the literature, FCBoost has various interesting properties as a cascade Boosting algorithm. First, its example weighing is very similar to that of AdaBoost [26]. A comparison of (III.5) and (III.15) shows that FCBoost reweights examples by how well they are classified by the current cascade. As in AdaBoost, this is measured by the classification margin, but now with respect to the cascade predictor, F, (margin yF) rather than a simple predictor f (margin yf). Second, the weak learner selection rule of FCBoost is very similar to that of AdaBoost. While in (III.6) AdaBoost selects the weak learner g that maximizes

$$\frac{1}{|S_t|}\sum_i y_i w_i g(x_i),$$

in (III.52)-(III.53) FCBoost selects the stage k and weak learner g that maximize

$$\sum_i \left(\frac{y_i w(x_i) b_k(x_i)}{|S_t|} - \eta \frac{y_i^s \psi_k(x_i)\theta_k(x_i)}{|S_t^-|}\right) g(x_i). \tag{III.59}$$

When $\eta = 0$, the only significant difference is the inclusion of $b_k(x_i)$ in (III.59). To understand the role of this term note that, from (III.36) and (III.40), $b_k(x_i) = 0$ whenever $\gamma_k(x_i) = 0$ in (III.30), and $\xi_k(x_i) = 0$ in (III.31). This implies that

there is at least one stage $j < k$ such that $f_j(x_i) < 0$, i.e. where x_i is rejected. When this holds, $b_k(x_i) = 0$ prevents x_i from influencing the update of $f_k(x)$. This is sensible: since x_i will not reach the kth stage, it should not affect its learning. Hence, the coefficients $b_k(x_i)$ can be seen as *gating coefficients*, which prevent examples rejected by earlier stages from affecting the learning of stage k. If $\eta \neq 0$, a similar role is played by $\theta_k(x_i)$ in the second term of (III.59) since, from (III.44), $\theta_k(x_i) = 0$ whenever $\gamma_k(x_i) = 0$. Thus, if x_i is rejected by a stage $j < k$, its processing complexity is not considered for any stage posterior to j. Due to the gating coefficients $b_k(x_i)$ and $\theta_k(x_i)$, FCBoost emulates the bootstrapping procedure commonly used in cascade design. This is a procedure that eliminates the examples rejected by each stage from the training set of subsequent stages. These examples are replaced with new false positives [91, 84]. While FCBoost emulates "example discarding" with the gating coefficients $b_k(x_i)$ and $\theta_k(x_i)$, it does not seek new false positives. This still requires the "training set augmentation" of bootstrapping.

A third interesting property of FCBoost is the complexity penalty (second term) of (III.59). From (III.44) and (III.50) this is, up to constants,

$$-y_i^s \gamma_k(x_i) e^{f_k(x_i)} \mathcal{C}(F_{k+1}, x_i) g(x_i).$$

Given example x_i and cascade stage k, all factors in this product have a meaningful interpretation. First, since $y_i^s \gamma_k(x_i)$ is non-zero only for negative examples which have not been rejected by earlier cascade stages ($j < k$), it acts as a selector of the false-positives that reach stage k. Second, since $f_k(x_i)$ measures how deeply x_i penetrates the positive side of the stage k classification boundary, $e^{f_k(x_i)}$ is large for the false-positives that stage k confidently assigns to the positive class. Third, since $\mathcal{C}(F_{k+1}, x_i)$ is the complexity of processing x_i by the stages beyond k, it measures how deeply x_i penetrates the cascade, if not rejected by stage k. Finally, $g(x_i)$ is the label given to x_i by weak learner $g(x)$. Since only $g(x_i)$ can be negative, the product is maximized when $g(x_i) = -1$, $\gamma_k(x_i) = 1$ and $f_k(x_i)$ and $\mathcal{C}(F_{k+1}, x_i)$ are as large as possible. Hence, the best weak learner is that which, on average,

declares as negatives the examples which 1) are false-positives of the earlier stages, 2) are most confidently accepted as false-positives by the current stage, and 3) penetrate the cascade most deeply beyond this stage. This is intuitive, in the sense that it encourages the selection of the weak learner that most contradicts the current cascade on its most costly mistakes.

In summary, FCBoost is a generalization of AdaBoost with similar example weighting, gating coefficients that guarantee consistency with the cascade structure, and a cost function that accounts for classifier complexity. This encourages the selection of weak learners that correct the false-positives of greatest computational cost. It should be mentioned that while we have used AdaBoost to derive FCBoost, similar algorithms could be derived from other forms of Boosting, e.g. logitBoost, gentle Boost [27], KLBoost [46] or float Boost [44]. This would amount to replacing the exponential loss, (III.3), with other loss functions. While the resulting algorithms would be different, the fundamental properties (example reweighing, additive updates, gating coefficients) would not. We next exploit this to develop a cost-sensitive extension of FCBoost.

III.F.4 Cost-sensitive FCBoost

While positive examples rejected by a cascade stage cannot be recovered by subsequent stages, the cascade false positive rate can always be reduced through addition of stages. Hence, in cascade learning, maintaining a high detection rate across stages is more critical than maintaining a low false positive rate. This is difficult to guarantee with the risk of (III.1), which is an upper bound on the error rate, treating misses and false positives equally. Several approaches have been proposed to enforce asymmetry during cascade learning. One possibility is to manipulate the thresholds of the various detector stages to guarantee the desired detection rate [91, 81, 47]. This is usually sub-optimal, since Boosting predictors are not well calibrated outside a small neighborhood of the classification boundary [54]. Threshold tuning merely changes the location of the boundary

and can perform poorly [49]. An alternative is to use cost sensitive Boosting algorithms [92, 50], derived from asymmetric losses that weigh miss-detections more than false-positives, optimizing the cost-sensitive boundary directly. This usually outperforms threshold tuning.

In this work we adopt the cost sensitive risk of [92],

$$\begin{aligned} R_E^c(f) &= \frac{C}{|S_t^+|}\sum_{x_i\in S_t^+} e^{-y_i f(x_i)} + \frac{1-C}{|S_t^-|}\sum_{x_i\in S_t^-} e^{-y_i f(x_i)} \\ &= \sum_{x_i\in S_t} y_i^c e^{-y_i f(x_i)}, \end{aligned} \tag{III.60}$$

where $C \in [0,1]$ is a cost factor,

$$y_i^c = \frac{C}{|S_t^+|} I(y_i = 1) + \frac{1-C}{|S_t^-|} I(y_i = -1),$$

$I(.)$ the indicator function, and the relative importance of positive vs. negative examples is determined by the ratio $\frac{C}{1-C}$. This leads to the cost-sensitive Lagrangian

$$\mathcal{L}^c[F] = R_E^c[F] + R_C[F]. \tag{III.61}$$

A derivation similar to that of (III.14) can be used to show that

$$< \delta R_E^c[F], g >_k = -\sum_i y_i y_i^c w(x_i) b_k(x_i) g(x_i), \tag{III.62}$$

where $w(x_i) = e^{-y_i F(x_i)}$ and $b_k(x_i)$ is given by (III.36) for last-stage and by (III.40) for multiplicative cascades. Finally, combining (III.61), (III.62), and (III.49),

$$< -\delta\mathcal{L}^c[F], g >_k = \sum_i \left(y_i y_i^c w(x_i) b_k(x_i) - \eta \frac{y_i^s \psi_k(x_i)\theta_k(x_i)}{|S_t^-|} \right) g(x_i) \tag{III.63}$$

The cost-sensitive version of FCBoost replaces (III.51) with (III.63) in (III.52)-(III.53) and $\mathcal{L}$ by $\mathcal{L}^c$ in (III.54).

III.F.5 Open issues

One subtle difference between AdaBoost and FCBoost, with $\eta = 0$, is the feasible set of the underlying optimization problems. Rewriting the FCBoost

problem of (III.13) as

$$\begin{cases} \min_f & R_E[f] \\ s.t: & f \in \Omega_{\mathbf{G}}, \end{cases} \tag{III.64}$$

where

$$\Omega_{\mathbf{G}} = \{f | \exists f_1, ...f_m \in \mathbf{G} \quad \text{such that} \quad f(x) = \mathcal{F}^m[f_1, \ldots, f_m](x) \quad \forall x\}.$$

and comparing (III.64) to (III.2), the two problems differ in their feasible sets, $span(\mathbf{G})$ for AdaBoost vs. $\Omega_{\mathbf{G}}$ for FCBoost. Since any $\hat{f} \in span(\mathbf{G})$ is equivalent to a one-stage cascaded predictor, it follows that $\hat{f} \in \Omega_{\mathbf{G}}$ and

$$span(\mathbf{G}) \subset \Omega_{\mathbf{G}}.$$

Hence, the feasible set of FCBoost is larger than that of AdaBoost, and FCBoost can, in principle, find detectors of lower risk. Hence, all generalization guarantees of AdaBoost hold, in principle, for cascades learned with FCBoost. There is, however, one significant difference. Since $span(\mathbf{G})$ is a convex set, the optimization problem of (III.2) is convex whenever $R_E(f)$ is a convex function of f. This is the case for the AdaBoost risk, and AdaBoost is thus guaranteed to converge to a global minimum. However, since $\Omega_{\mathbf{G}}$ can be a non-convex set, no such guarantees exist for FCBoost. Hence, FCBoost can converge to a local minimum. We illustrate this with an example in section III.G.1. In general, the convexity of $\Omega_{\mathbf{G}}$ depends on the PC operator $\mathcal{F}^m$ and the set of weak learners $\mathbf{G}$. There is currently little understanding on what conditions are necessary to guarantee convexity.

III.G Evaluation

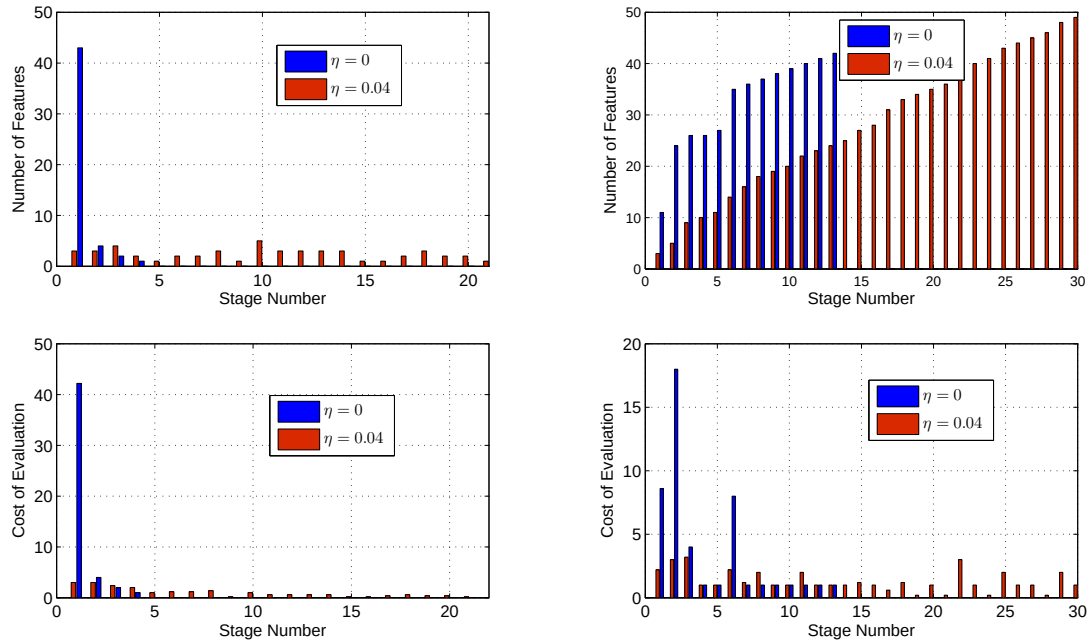
In this section, we report on several experiments conducted to evaluate FCBoost. We start with a set of experiments designed to illustrate the properties of the algorithm. We then report results on its use to build face and pedestrian detectors with state-of-the-art performance in terms of detection accuracy and complexity. In all cases, the training set for face detection contained $4,500$ faces

(along with their flipped replicas) and $9,000$ negative examples, of size 24×24 pixels, while pedestrian detection relied on a training set of $2,347$ positive and $2,000$ negatives examples, of size 72×30, from the Caltech Pedestrian data set [18]. All weak learners were decision-stumps on Haar wavelets [91].

III.G.1 Effect of η

We started by studying the impact of the Lagrange multiplier η, of (III.41), on the accuracy vs. complexity performance of FCBoost cascades. The test set consisted of 832 faces (along with their flipped replicas) and $1,664$ negatives. All detectors were trained for 50 iterations. The unit computational cost was set to the cost of evaluating a new Haar feature. This resulted in a cost of $\frac{1}{5}$ units for feature recycling, i.e. $\lambda = \frac{1}{5}$ in (III.47). Figure III.3 quantifies the structure of the cascades learned by FCBoost with $\eta = 0$ and $\eta = 0.04$: multiplicative in a) and last-stage in b). The top plots summarize the number of features assigned to each cascade stage, and those at the bottom the computational cost per stage. Note that since, from (III.57), the neutral predictor of the last-stage cascade is its last stage, each of the last-stage cascade stages benefits from the features evaluated in the previous stages. Hence, as shown in the top plot of Figure III.3-b, the number of weak learners per stage is monotonically increasing. However, because most features are recycled, the cost is still dominated by the early stages, when $\eta = 0$. With respect to the impact of η, its is clear that, for both structures, a small η produces short cascades whose early stages contain many weak learners. On the other hand, a large η leads to much deeper cascades, and a more uniform distribution of weak learners and computation. This is sensible, since larger η place more emphasis on computational efficiency and this requires that the early stages, which tend to be evaluated for most examples, be very efficient. Hence, long cascades with a few weak learners per stage tend to be computationally more efficient than short cascades with many learners per stage.

The accuracy vs. complexity trade-off of these cascades was compared



(a) (b)

Figure III.3: Number of features (top) and computational cost (bottom) per stage of an FCBoost cascade: (a) multiplicative, (b) last-stage.

to those of a non-cascaded AdaBoost detector and a cascade of embedded stages derived from this detector, by the procedure of [50]. This converts the detector into a cascade by inserting a rejection point per weak learner. The resulting cascade has embedded stages which add a single weak learner to their predecessors and is equivalent to the chain Boost cascade [99]. Figure III.4 depicts the trade-off between computation and accuracy of AdaBoost, chain Boost, and FCBoost cascades with $\eta \in [0, 0.04]$. The left-most (right-most) point on the FCBoost curves corresponds to $\eta = 0$ ($\eta = 0.04$). AdaBoost and ChainBoost points were obtained by limiting the number of weak learners, with a single weak learner (full detector) for the right-most (left-most) point. Several observations can be made from the figure. First, as expected, increasing the trade-off parameter η produces FCBoost cascades with less computation and higher error. Second, FCBoost has a better trade-off between complexity and accuracy (curves closer to the origin). Third, among FCBoost models, last-stage cascades have uniformly better trade-off

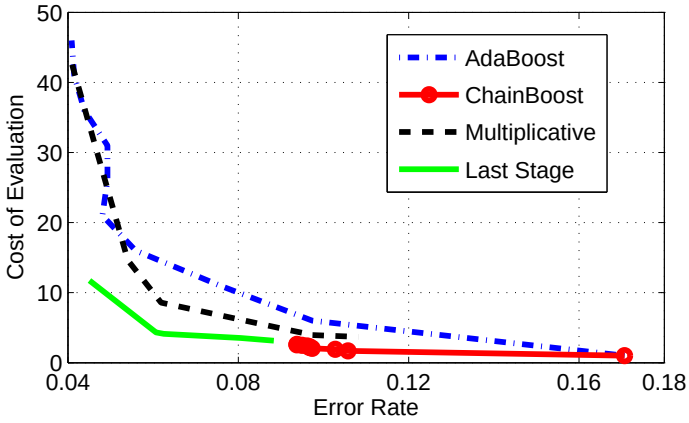


Figure III.4: Computational cost vs. error rate of the detectors learned with AdaBoost, chain Boost, and FCBoost with the last-stage and multiplicative structures.

than their multiplicative counterparts. Since last-stage are generalized embedded cascades, this confirms previous reports on the advantages of embedded over independent stages [64, 99]. Finally, it is interesting to note that, when $\eta = 0$, the Lagrangian of (III.41) is equivalent to the AdaBoost risk, i.e. FCBoost and AdaBoost minimize the same objective. However, due to their different feasible sets, they can learn very different detectors (see section III.F.5). While the larger feasible set of FCBoost suggests that it should produce detectors of smaller risk than AdaBoost, this did not happen in our experiments.

Table III.1 summarizes the error and cost of AdaBoost and the two FCBoost methods for $\eta = 0$. Note that the AdaBoost detector has a slightly lower error. The weaker accuracy of the FCBoost detectors suggests that the latter does get trapped in local minima. This is, in fact, intuitive as the decision to add a cascade stage makes it impossible for the gradient descent procedure to revert back to a non-cascaded detector. By making such a decision, FCBoost can compromise the global optimality of its solution, if the global optimum is a non-cascaded detector. Interestingly, FCBoost sometimes decides to add stages even when $\eta = 0$ (see Figure III.3). As shown in Table III.1, this leads to a slightly more error-prone but much more efficient detector than AdaBoost. In summary, even without pres-

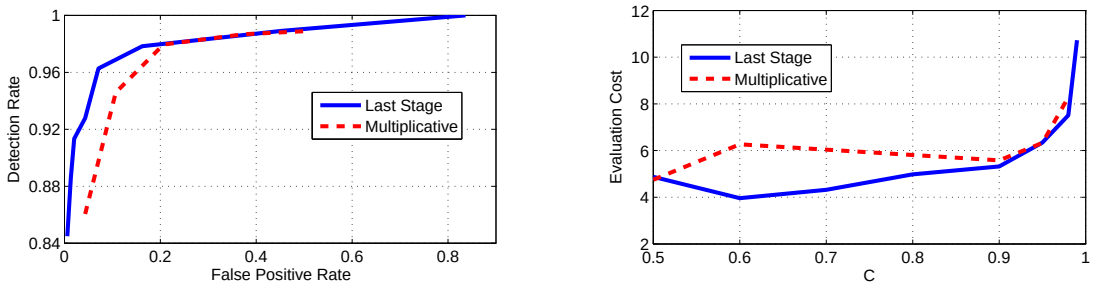
Table III.1 Performance comparison between AdaBoost and FCBoost, for $\eta = 0$.

	AdaBoost	FCBoost+last-stage	FCBoost+multiplicative
Err. rate	4.03%	4.51%	4.15%
Eval. cost	50	11.74	42.54

sure to minimize complexity ($\eta = 0$), FCBoost may trade-off error for complexity. This may be desirable or not, depending on the application. In the experiment of Table III.1, FCBoost seems to make sensible choices. For the last-stage structure, it trades a small increase in error (0.48%) for a large decrease in computation (76.5%). For the multiplicative structure, it trades-off a very small increase in error (0.12%) for a moderate (16%) decrease in computation.

III.G.2 Cost-Sensitive FCBoost

We next consider the combination of FCBoost and the cost sensitive risk of (III.60). Since the advantages of cost-sensitive Boosting over threshold tuning are now well established [92, 49], we limit the discussion to the effect of the cost factor C on the behavior of FCBoost cascades. Cascaded face detectors were learned for cost factors $C \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$. Figure III.5 a) presents the trade-off between detection and false positive rate for last-stage and multiplicative cascades. In both cases, the leftmost (rightmost) point corresponds to $C = 0.5$ ($C = 0.99$). Figure III.5 b) presents the equivalent plot for computational cost. Several observations can be made. First, as expected, larger cost factors C produce detectors of higher detection and higher false-positive rate. Second, they lead to cascades of higher complexity. This is intuitive since, for large cost factors, FCBoost aims for a high detection rate and is very conservative about rejecting examples. Hence, many negatives penetrate deep into the cascade, and computation increases. Third, comparing the curves of the last-stage and multiplicative cascades, the former again has better performance. In particular, last-stage cascades combine higher ROC curves in Figure III.5 a) with lower computational cost



(a) (b)

Figure III.5: Performance of cascades learned with cost-sensitive FCBoost, using different cost factors C. (a) ROC curves, (b) computational complexity.

in Figure III.5 b).

III.G.3 Face and pedestrian detection

Over the last decade, there has been significant interest in the problem of real-time object detection from video streams. In particular, the sub-problems of face and pedestrian detection have been the focus of extensive research, due to the demand for face detection in low-power consumer electronics (e.g. cameras or smart-phones) and pedestrian detection in intelligent vehicles. In this section, we compare the performance of FCBoost cascades with those learned by several state of the art methods in the face and pedestrian detection literatures.

We start with face detection, where cascaded detectors have become predominant, comparing FCBoost to the method of Viola and Jones (VJ) [91], Wald Boost [81] and the multi-exit approach of [64]. Since extensive results on these and other methods are available on the MIT-CMU test set, all detectors were evaluated on this data set. The methods above have been shown to outperform a number of other cascade learning algorithms [64] and, to the best of our knowledge, hold the best results in this data set. In all cases, the target detection rate was set to $D_T = 95\%$. For Wald Boost, multi-exit, and VJ, the training set was bootstrapped when a new stage was added to the cascade, for FCBoost when the false positive rate dropped below 95%. For VJ and multi-exit cascades, which require

the specification of the number of cascade stages and a target false-positive and detection rate per stage, we used 20 stages, and the popular strategy of setting the false positive rate to 50% and the detection rate to $D_T^{\frac{1}{20}}$. For FCBoost we used a last-stage cascade, since this structure achieved the best balance between accuracy and speed in the previous experiment. We did not attempt to optimize η, simply using $\eta = 0.02$. The cost factor C was initialized with $C = 0.99$. If after a Boosting update the cascade did not meet the detection rate, C was increased to

$$C_{new} = \frac{C_{old} + 1}{2}. \tag{III.65}$$

This placed more emphasis on avoiding misses than false positives, and was repeated until the updated cascade satisfied the rate constraint. The final value of C was used as the initial value for the next Boosting update.

Figure III.6 show the ROCs of all detectors. The average evaluation cost, i.e.average number of features evaluated per sub-window, is shown in the legend for each method. Note that the FCBoost cascade is simultaneously more accurate and faster than those of all other methods. For example, at 100 false positives, FCBoost has a detection rate of 91% as opposed to 88% for multi-exit, 83% for VJ, and 80% for Wald Boost. With regards to computation, FCBoost is 7.1, 4, and 2.5 times faster than multi-exit, VJ, and Wald Boost, respectively. Overall, when compared to the FCBoost cascade, the closest cascade in terms of detection rate (multi-exit, 3% drop) is significantly slower (7 times) and the closest cascade in terms of detection speed (Wald Boost, 2.5 times slower) has a very poor detection rate (11% smaller).

We next considered the problem of pedestrian detection, comparing results to a large set of state-of-the-art pedestrian detectors on the Caltech Pedestrian data set [18]. In this literature, it is well known that a good representation for pedestrians must account for both edge orientation and color [15, 60]. Similarly to [60], we adopted an image representation based on a 10 channel decomposition. This included 3 color channels (YUV color space), 6 gradient orientation channels,

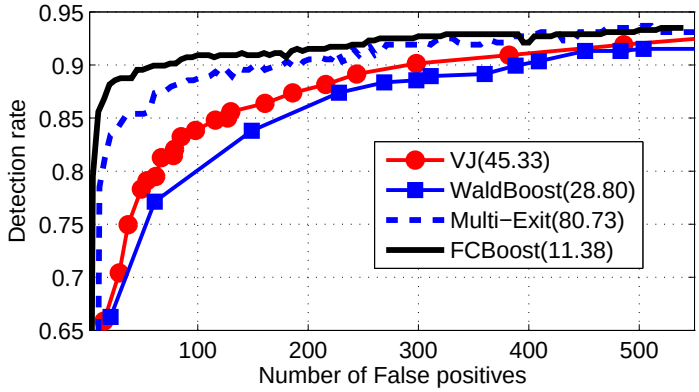


Figure III.6: ROCs of various face detectors on MIT-CMU. The number in the legend is the average evaluation cost, i.e.average number of features evaluated per sub-window.

and a gradient magnitude channel. In all other aspects, the cascade architecture was as before, e.g. using Haar wavelet features and decision stumps as weak learners, the previously used values for parameters D_T, and η, etc. When compared to the face detection experiments, the only difference is that the set of weak learners was replicated for each channel. At each iteration, FCBoost chose the best weak learner and the best channel to add to the cascade predictor. The performance of the FCBoost cascade was evaluated with the toolbox of [18]. Figure III.7 compares its complexity and curve of miss-detection rate vs number of false positives per image (FPPI) to those of a number of recent pedestrian detectors. The comparison was restricted to the popular near scale-large setting, which evaluates the detection of pedestrians with more than 100 pixels in height. The numbers shown in the left of the legend summarize the detection performance by the miss rate at 0.1 FPPI. The numbers shown in the right indicated the average time, in seconds, required for processing a 480×640 video frame. Note that the evaluation is *not* restricted to fast detectors, including the most popular architectures for object detection in computer vision, such as the HOG detector of [15] or the latent SVM of [24]. For more information on the curves and other methods the reader is referred to [18].

Two sets of conclusions can be derived from these results. First, they con-

firm the observation that the FCBoost cascade significantly outperforms previous cascaded detectors. A direct comparison is in fact possible against the ChnFtrs approach of [60]. This work introduced the multi channel features that we adopt but uses the SoftCascade algorithm [8] for cascade learning. The resulting detector is among the top methods on this data set, missing 30% of the pedestrians at 0.1 FPPI and using 0.85 seconds to process a frame. Nevertheless, the FCBoost cascade has substantially better accuracy, missing only 23% of the pedestrians at 0.1 FPPI, and requires less time (a 6% speed up). Second, the results of Figure III.7 show that the FCBoost cascade is one of the most accurate pedestrian detectors in the literature, and significantly faster than the detectors of comparable accuracy. In fact, only two detectors have been reported to achieve equivalent or lower miss rates. The Hog-Lbp detector [94] has the same miss rate (23% at 0.1 FPPI) but is 20 times slower. The MultiFtr+Motion [93] detector has a smaller miss rate of 16% (at 0.1 FPPI) but is 62 times slower (almost 1 minute per frame). The inclusion of this method in Figure III.7 is somewhat unfair, since it is the only approach that exploits motion features. All other detectors, including the FCBoost cascade, operate on single-frames. We did not investigate the impact of adding motion features to FCBoost. Finally, it should be noted that the FCBoost cascade could be enhanced with various computational speed ups proposed by [17] in the design of the FPDW detector. This is basically a fast version of the ChnFtrs detector, using several image processing speed-ups to reduce the time necessary to produce the image channels on which the classifier operates. These speed-ups lead to a significant increase in speed (0.15 vs 0.85 seconds) at a marginal cost in terms of detection accuracy (33% vs. 30% miss rate at 0.1 FPPI). Since these enhancements are due to image processing, not better cascade design, we have not considered them in our implementation. We would expect, however, to see similar computational gains in result of their application to the FCBoost cascade.

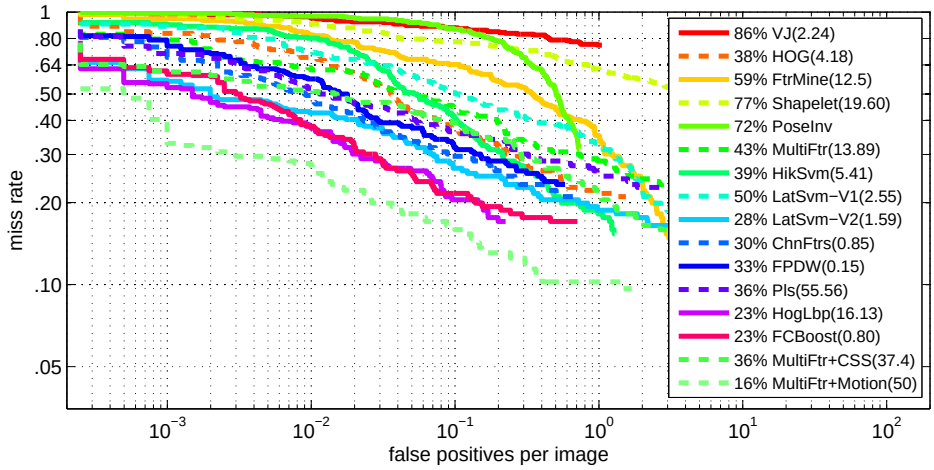


Figure III.7: Accuracy curves and complexity of various pedestrian detectors on the Caltech data set. Legend: (left) miss rates at 0.1 FPPI, (right) average time, in seconds, required to process 480×640 frame.

III.H Conclusions

In this work we have addressed the problem of detector cascade learning by introducing the FCBoost algorithm. This algorithm optimizes a Lagrangian risk that accounts for both detector speed and accuracy with respect to a predictor that complies with the sequential decision making structure of the cascade architecture. By exploiting recursive properties of the latter, it was shown that many cascade predictors can be derived from generator functions, which are cascade predictors of two stages. Variants of FCBoost were derived for two members of this family, last-stage and multiplicative cascades, which were shown to generalize the popular independent and embedded stage cascade architectures. The concept of neutral predictors was exploited to integrate the search for cascade configuration into the Boosting algorithm. In result, FCBoost can automatically determine 1) the number of cascade stages and 2) the number of weak learners per stage, by minimizing the Lagrangian risk. It was also shown that FCBoost generalizes AdaBoost, and is compatible with existing cost-sensitive extensions of Boosting. Hence, it can be used to learn cascades of high detection rate. Experimental evaluation has shown

that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

III.I Acknowledgments

The text of Chapter III, in full, is based on the material as it is going to appear in: Mohammad Saberian and Nuno Vasconcelos. Boosting Algorithm for Learning Detector Cascade, *to appear in Journal of Machine Learning Research (JMLR)* and the material as it is appeared in: Mohammad Saberian and Nuno Vasconcelos. Boosting Classifer Cascades. *Proceedings of Neural Information Processing Systems (NIPS)*, 2010. The dissertation author was a primary researcher and an author of the cited material.

Chapter IV

Multiclass Boosting

IV.A Introduction

Boosting is a popular approach for classifier design in machine learning. It is a simple and effective procedure to combine many weak learners into a strong classifier. However, most existing Boosting methods were designed primarily for binary classification. In many cases, the extension to M-ary problems (of $M > 2$) is not straightforward. Nevertheless, the design of multiclass Boosting algorithms has been investigated since the introduction of AdaBoost in [26].

Two main approaches have been attempted. The first is to Boost multiclass weak learner, e.g. decision trees, such as AdaBoost-M1[25], [23], SAMME[102], and AdaBoost-Cost [56]. These methods may require strong weak learners, specially AdaBoost-M1, which substantially increase complexity and have high potential for over-fitting.

The second approach is to reduce the multiclass problem into a collection of binary sub-problems. Beyond the popular methods of "one vs all" [57] and "all pairs" [33] the rest of these methods fall into three branches. The first branch started with work of Sejnowski et al. [77] and error correcting output coding approach of [16], where a binary code is assigned to each class and *independent* binary classifiers are trained to predict the bits of those codes. Schapire [72] used this approach in Boosting and proposed AdaBoost.OC [72]. This method uses the pseudo loss definition of AdaBoost.M2 [25] which makes it possible to learn binary sub-classifiers jointly. This approach then followed and improved by many other algorithms including AdaBoost.ECC [32],[4], AdaBoost.SECC [83], AdaBoost.ERP [43], AdaBoost.SIP [100] and HingBoost [30]. The main difficulty for these methods is the lack of optimal code for each class which is NP-hard [14].

The Second branch of reduction to binary approaches includes AdaBoost-M2 [25], AdaBoost-MR and AdaBoost-MH [73]. In these approaches class numbers $c \in \{1 \dots M\}$ are coupled with the example $x \in \mathcal{X}$ and a binary predictors $f : \mathcal{X} \times \{1 \dots M\} \to \mathbb{R}$ is trained to predict responses of example x for all classes.

The class with largest response then is declared as the prediction. This will effectively converts the multiclass problem into a binary problem where examples are augmented with their class numbers [27]. The resulting binary classification problem is more complicated and requires weak learners with very high discriminative power and thus the resulting Boosting algorithms may not work well with the regular weak learners such as decision stumps.

The third branch of binary approaches started with multiclass LogitBoost [27] which jointly trains M regression classifiers for solving M-ary classification problem. This approach later followed by [37] to propose multiclass GentleBoost and by [104] to propose AdaBoost.ML.

In this work, we introduce a new formulation for multiclass Boosting based on 1) multi-dimensional predictor, 2) multi-dimensional real-value codewords and 3) a new family of multiclass loss functions. Using these definitions, we then formulate multiclass Boosting as an optimization in multi-dimensional function space and propose two Boosting algorithms to solve this problem. The first is CD-MCBoost which implements functional coordinate descent procedure. CD-MCBoost supports any type of weak learners, updating one component of the predictor per Boosting iteration. This method is similar to reduction to binary approach but in CD-MCBoost 1) codewords are real values and 2) predictor components are learned jointly. The second algorithm, GD-MCBoost, implements functional gradient descent, uses multiclass weak learners and updates all components of the predictor simultaneously. We also show that both MCBoost algorithms are maximizing the margin, however, the selected codewords will impose an upper bound on the maximum achievable margin. We then find the optimal set of codewords that maximized this upper bound. In addition we show that MCBoost algorithms converge to a Bayes consistent predictor and reduce to Binary Boosting for binary problems. Experiments show that the MCBoost algorithms outperform comparable prior methods on a number of data sets.

IV.B Previous works

The problem of multiclass classification has attracted significant attention since the early days of machine learning. The first and most popular methods for solving M-class classification problem is to build M classifiers each separating one class form the other classes [57]. At the test time score of each classifier is computed and the class with highest score is declared as the prediction. Later Sejnowski et al [77] extend this idea by assigning a binary string of length l to each class and training l classifiers for learning bits of those strings, similar to "one-vs-all", at the test time bits of the string are predicted and the class with lowest Hamming distance is selected. In 1995, Dietterich et al. [16] improve this method by using Error Correcting Output Codes, ECOC, which made it possible to extract the true class even if there are a few errors in the predicted bits. Similarly [33] suggested designing $\frac{M(M-1)}{2}$ classifiers discriminating all pairs of classes and make a vote among these classifiers at the test time. Allwein et al. [4] unified all these binary based classification and showed that they are all reducing multiclass problem into binary sub-problems for some specific coding matrix.

After introduction of AdaBoost [26] in 1995, there was a great effort to extend this simple and effective binary classification algorithm to the multiclass case. These efforts fall into two main categories 1) Boosting multiclass weak learners and 2) converting the multiclass problem into binary sub-problems.

IV.B.1 Boosting multiclass weak learners

The first method in this class is AdaBoost.M1 [26, 25] which is a direct extension of AdaBoost to multiclass case. However, unlike AdaBoost which only requires "better than random" base learners, AdaBoost.M1 needs more stronger base learners, i.e. error rate less than 50%, for Boosting. The weak learner slection criteria of this method was furthur relaxed by [23]. Recently [56] provided a game theoric framework for analyzing the required condition of Boostable base learners

and proposed a more relaxed base learner selection criteria. However, finding the optimal criteria for selection of base learners is still an open problem.

The other method in this class is SAMME [102] which defines an exponential loss for multiclass classification in multi-dimensional function space. SAMME then minimizes this loss using by gradient descent. However, as we will show in section IV.I, the defined loss function is not margin enforcing and thus SAMME is not a margin maximizer algorithm.

IV.B.2 Reduction to binary

The multiclass classification algorithms based on reduction to binary fall into three branches. The first branch follows the error correcting output coding approach [16]. Schapire [72] combined it with pseudo-loss definition of AdaBoost.M2 [25] and proposed AdaBoost.OC. He also proposed using binary random codes for finding codes with highest error correcting ability using "max-cut" algorithms. Later [32] modified the pseudo-loss definition of AdaBoost.OC and proposed AdaBoost.ECC with better generalization guarantees and performance. In 2000, [4] proposed to use loss-based distance instead of Hamming distance in prediction and training. Connecting these methods with margin framework, in 2005 [83] showed that AdaBoost.OC and AdaBoost.ECC were in fact maximizing multiclass definition of margin. In 2002 Crammer et al. [14] showed that the problem of finding the optimal coding matrix is NP-hard and suggested to use real-valued codes. The problem of finding optimal binary coding matrix was also studied in [43, 100, 30] where they proposed optimization methods to find a good set of binary codes in each iteration. Performance of these methods depends on two factors 1) The error correction quality of the coding matrix and weighting algorithms used in the training binary classifiers. However optimizing these two factors as pointed out by [14] is NP-hard and these method often require extensive computations.

The second branch of reduction to binary methods started with the introduction of AdaBoost-M2 [26, 25]. In this method class numbers $c \in \{1 \dots M\}$ are

coupled with the examples $x \in \mathcal{X}$ and a real-valued predictor $f : \mathcal{X} \times \{1 \dots M\} \rightarrow \mathbb{R}$ is trained to predict response of example x for the class numbers. This predictor is trained by minimizing a pseudo-loss function that is defined over all pairs of examples and their corresponding incorrect labels to penalize the errors. At the test time for a given example x, this predictor is evaluated over all pairs of x and class numbers, and the class with largest response is declared as the prediction. Schapire et al. [73], extend AdaBoost.M2 to AdaBoost.MR for multi-label problems. He also improved this algorithm by introducing AdaBoost-MH whose weights were updated using Hamming loss. As it shown by [27], coupling examples with their class numbers effectively converts the multiclass problem into a new binary problem for learning responses of each example to different class numbers. However, this binary problem is very complicated and requires weak learners with very high discriminative power which may result in over-fitting.

The third branch started with multiclass LogitBoost proposed by [27]. This method uses statistical view of Boosting as gradient descent in function space and learns additive logistic regression models for each class using LogitBoost. The key difference between multiclass LogitBoost and "one-vs-all" approach is that in this method binary predictors are learned jointly and sums up to zeros. Later [37] adapted this framework for GentleBoost and proposed GAMMBLE algorithm. This framework was further extend by [104] to be used with any Fisher consistent loss function.

IV.C Multiclass Boosting

We start by reviewing the fundamental ideas behind the classical use of Boosting for the design of *binary* classifiers, and then extend these ideas to the multiclass setting.

IV.C.1 Binary classification

A binary classifier, $F(x)$, implements a *decision rule* that maps examples $x \in \mathcal{X}$ to classes $c \in \{1, 2\}$. The classifier is optimal when this decision rule minimizes some *classification risk*. A classical risk is the probability of classification error, which is minimized by the Bayes decision rule

$$F(x) = \arg \min_{c \in \{1,2\}} P_{C|X}(c|x). \tag{IV.1}$$

This rule is not easy to implement, due to the difficulty of estimating the probabilities $P_{C|X}(c|x)$. Large margin methods, such as Boosting, avoid this difficulty by adopting alternative risks. They implement the classifier as

$$F(x) = \begin{cases} 1 & \text{if} \quad f^*(x) < 0 \\ 2 & \text{if} \quad f^*(x) > 0. \end{cases} \tag{IV.2}$$

where $f^*(x) : \mathcal{X} \to \mathbb{R}$ is the *continuous valued predictor* that minimizes the risk

$$f^*(x) = \arg \min_f R_L(f), \tag{IV.3}$$

where

$$R_L(f) = E_{X,C}\{L[y^c, f(x)]\}, \tag{IV.4}$$

defined by a *loss function* $L[.,.]$ and a set of *class labels* y^c, where y^c is the label of class $c \in \{1, 2\}$. If the loss $L[.,.]$ is Bayes consistent, the minimization of (IV.4) results in the Bayes decision rule, i.e. (IV.1) and (IV.2) are equivalent.

To learn the optimal classifier, the risk of (IV.4) is estimated by the empirical risk

$$\overline{R}_L(f) \approx \frac{1}{n} \sum_{i=1}^{n} L[y^{c_i}, f(x_i)] \tag{IV.5}$$

over a training sample $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$. Large margin methods use the *labels* $y^1 = -1$ and $y^2 = 1$ and a Bayes consistent loss function that only depends on the *classification margin* $y^c f(x)$, i.e.

$$L[y^c, f(x)] = L[y^c f(x)]. \tag{IV.6}$$

This guarantees that the classifier has good generalization for finite training samples [88]. Boosting learns the optimal predictor $f^*(x) : \mathcal{X} \to \mathbb{R}$ as the solution of

$$\begin{cases} \min_{f(x)} & \overline{R}_L(f) \\ s.t & f(x) \in span(\mathcal{H}), \end{cases} \tag{IV.7}$$

where $\overline{R}_L(f)$ is the empirical risk of (IV.5), and $\mathcal{H} = \{h_1(x), ...h_r(x)\}$ a set of weak learners $h_i(x) : \mathcal{X} \to \mathbb{R}$. The optimization is carried out by gradient descent in the function space $span(\mathcal{H})$ of linear combinations of $h_i(x)$ [27, 53, 70]. The extension of binary Boosting to the multiclass setting requires multiclass definitions of class labels, predictor, margin, decision rule, loss function and risk minimization procedure.

IV.D Multiclass class labels, predictors and margin

We start by introducing a set of multiclass definitions for class labels, predictor, and margin.

IV.D.1 Class labels and predictor

The definition of the classification labels as $y^c = \pm 1$ plays a significant role in the binary formulation. One of the difficulties of the multiclass extension is that these labels do not have an obvious generalization. For M-ary classification, $c \in \{1, \ldots, M\}$, each class c must be mapped into a distinct class label $y^c \in \mathcal{Y} = \{y^1 \ldots y^M\}$. This label can be thought of as a *codeword* that identifies the class. In the binary case, the predictor is a real-valued function, i.e. $f(x) \in \mathbb{R}$, and the codewords ± 1 are the two directions on the line. To generalize these concepts to the multiclass setting, we introduce a multi-dimensional predictor $f(x) \in \mathbb{R}^d$ and codewords y^k which are directions in this space

$$y^c \in \mathbb{R}^d, \quad \|y^c\| = 1. \tag{IV.8}$$

At this point, there is no restriction on the dimension d or the codeword directions, which can be any M distinct directions in $\mathbb{R}^d$. In Section IV.E we will discuss how the selection of codewords affects learning performance and procedures for determining optimal sets of codewords.

IV.D.2 Margin

Given a multi-dimensional predictor and codewords, we rely on the following definition of multiclass margin.

Definition 1. *Let $y^k \in \mathbb{R}^d, k \in \{1, \ldots, M\}$, be the set of codewords of an M-ary classification problem and $f : \mathcal{X} \to \mathbb{R}^d$. The margin of example x with respect to class k is*

$$\begin{aligned} \mathcal{M}(y^k, f(x)) &= \frac{1}{2} \min_{l \neq k} \left[\left\langle y^k, f(x) \right\rangle - \left\langle y^l, f(x) \right\rangle \right] \\ &= \frac{1}{2} \left[\left\langle y^k, f(x) \right\rangle - \max_{l \neq k} \left\langle y^l, f(x) \right\rangle \right] \end{aligned} \tag{IV.9}$$

where $< ., . >$ is the Euclidean dot-product.

This definition is closely related to previous definitions of multiclass margin. For example, it generalizes that of [31], where the codewords y^k are restricted to binary vectors in the canonical basis of $\mathbb{R}^d$, and is a special case of that of [4], where the dot products $\left\langle y^k, f(x) \right\rangle$ are replaced by a generic function of f, x, and k. Furthermore, when $M = 2$ $y^1 = -y^2 = 1$,

$$\begin{aligned} \mathcal{M}(y^k, f(x)) &= \frac{1}{2}[y^k f(x) - \max_{l \neq k} y^l f(x)] \\ &= \frac{1}{2}[y^k f(x) + y^k f(x)] = y^k f(x), \end{aligned} \tag{IV.10}$$

and (IV.9) is identical to the classic definition of margin. We next generalize the concept of the predictor margin.

Definition 2. *The margin of a predictor $f(.)$ with respect to a set of codewords $\mathcal{Y} = \{y^1, ..y^M\}$ and examples $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$ is*

$$\mathcal{M}_p(\mathcal{D}, \mathcal{Y}, f) = \min_{(x_i, y^{c_i}) \in \mathcal{D}} \mathcal{M}(y^{c_i}, f(x_i)). \tag{IV.11}$$

Similarly to the binary case, this can be seen as a measure of the distance between the classification boundary and the point closest to it.

IV.D.3 Decision Rule

For a binary classifier, when $y^1 = 1$, $y^2 = -1$ the decision rule of (IV.2) can be written as

$$F(x) = \arg\max_{k=\{1,2\}} y^k f^*(x), \tag{IV.12}$$

i.e. the classifier simply chooses the class of largest margin for example x. This has the following straightforward extension to the multiclass case.

Definition 3. *Consider a M-ary classification problem with codewords $y^k \in \mathbb{R}^d, k \in \{1, \ldots, M\}$. A maximum margin classifier of predictor $f : \mathcal{X} \to \mathbb{R}^d$ implements the decision rule*

$$F(x) = \arg\max_{k\in\{1,\ldots,M\}} \mathcal{M}(y^k, f(x)). \tag{IV.13}$$

This can be shown equivalent to selecting the class whose codeword has largest dot-product with the prediction $f(x)$.

Lemma 1. *The decision rule of the maximum margin classifier of (IV.13) is equivalent to*

$$F(x) = \arg\max_{k\in\{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle. \tag{IV.14}$$

Proof. Consider a prediction $f(x)$. Defining

$$k^* = \arg\max_{k\in\{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle, \tag{IV.15}$$

it follows from (IV.9) that

$$\mathcal{M}(y^{k^*}, f(x)) \geq 0 \geq \mathcal{M}(y^l, f(x)) \quad \forall\, l \neq k^* \tag{IV.16}$$

and thus

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{M}(y^k, f(x)). \tag{IV.17}$$

Conversely, if

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{M}(y^k, f(x)) \tag{IV.18}$$

it follows from (IV.9) that

$$\left\langle y^{k^*}, f(x) \right\rangle \geq \max_{l \neq k^*} \left\langle y^l, f(x) \right\rangle \tag{IV.19}$$

and thus

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle. \tag{IV.20}$$

■

As in binary classification, an example x of class c is correctly classified by the max margin classifier of predictor $f(x)$ if and if and only if the example margin of x with respect to class c is positive.

Corollary 1. *Let c be the class of example x and $f(x)$ the predictor of a maximum margin classifier $F(x)$. Then $F(x) = c$ if and only if*

$$\mathcal{M}(y^c, f(x)) > 0. \tag{IV.21}$$

Proof. If $F(x) = c$, it follows from Lemma 1 that

$$c = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x_i) \right\rangle \tag{IV.22}$$

and (IV.21) follows from (IV.9). Conversely, If $\mathcal{M}(y^c, f(x)) > 0$, it follows from (IV.9) that

$$\langle y^c, f(x) \rangle > \max_{k \neq c} \left\langle y^k, f(x) \right\rangle \tag{IV.23}$$

and, from (IV.14), $F(x) = c$. ■

Finally, a maximum margin classifier classifies all examples in a data set $\mathcal{D}$ correctly if and only if its predictor margin with respect to $\mathcal{D}$ is positive.

Corollary 2. *Let $f(.)$ be the predictor of a maximum margin classifier $F(x)$ and $\mathcal{D}$ a set of examples (x_i, c_i). $f(.)$ classifies all $x_i \in \mathcal{D}$ correctly if and only if*

$$\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) > 0. \tag{IV.24}$$

Proof. If $\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) > 0$, it follows from (IV.11) that

$$\mathcal{M}(y^{c_i}, f(x_i)) > 0 \quad \forall x_i \in \mathcal{D} \tag{IV.25}$$

and, from Corollary 1, all examples are classified correctly. Conversely, if all examples are classified correctly, then (IV.25) follows from Corollary 1, and

$$\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) = \min_{(x_i, y^{c_i}) \in \mathcal{D}} \mathcal{M}(y^{c_i}, f(x_i)) > 0. \tag{IV.26}$$

■

These corollaries extend the equivalent properties of binary large margin predictors to the multiclass case.

IV.E Optimal codewords

Since, from (IV.14), the score of example x_i under class k is the dot product between the prediction $f(x_i)$ and the class codeword y^k, the choice of codewords has an impact on classification results. If, for example, two classes were to share a codeword, it would impossible to distinguish them with (IV.14) or (IV.13). Hence, some codeword sets are better than others. In this section, we search for an optimal set of codewords.

IV.E.1 Optimality criterion

The proposed optimality criterion is based on the definition of margin of (IV.9)-(IV.9). We note, however, that margin optimization is not enough to

constrain the learning problem. Two issues arise. The first is that, because the addition of a constant to all codewords leaves (IV.9) unaltered, a margin-based criterion cannot fully constrain the codeword set $\mathcal{Y}$. This problem can be eliminated by complementing the unit norm constraint of (IV.8) with the constraint that the codewords be *centered*.

Definition 4. *A set of vectors* $\mathcal{Y} = \{y^1, ...y^M\} \in \mathbb{R}^d$ *is denoted an* (M, d) *codeword set if*

$$\sum_{k=1}^{M} y^k = 0, \quad \|y^k\| = 1 \quad \forall k = 1 \ldots M. \tag{IV.27}$$

The set of all (M, d) *codeword sets is denoted* $\mathcal{S}(M, d)$.

The second is that the margin of (IV.9) can be arbitrarily increased without fundamentally changing the associated decision rule, by simply rescaling the predictor $f(.)$. This can be avoided by introducing a predictor normalization.

Definition 5. *A predictor* $f(x)$ *is normalized if* $||f(x)|| = 1, \forall x$. $\mathcal{F}$ *is the set of normalized predictors.*

Under these conditions, which we will adopt in the reminder of this work, the margin achievable with a codeword set $\mathcal{Y}$ is bounded. This leads to the notion of the margin capacity of $\mathcal{Y}$.

Definition 6. *Consider a* M*-ary classification problem with a codeword set* $\mathcal{Y} \in \mathcal{S}(M, d)$. *The margin capacity of* $\mathcal{Y}$ *is*

$$\mathcal{C}[\mathcal{Y}] \quad = \quad \min_{k=1\ldots M} \mathcal{M}(y^k, \xi^k), \tag{IV.28}$$

where

$$\xi^k = \arg\max_{||v||=1} \mathcal{M}(y^k, v). \tag{IV.29}$$

is the predictor direction of largest margin for class k.

The margin capacity $\mathcal{C}[\mathcal{Y}]$ is the maximum margin achievable by any predictor in $\mathcal{F}$ using codewords $\mathcal{Y}$ on any data set $\mathcal{D}$. Note, from (IV.11), that it is

the margin, with respect to $\mathcal{Y}$ and $\mathcal{D}$, of a predictor which maps all examples from class k into the direction ξ^k of largest margin. A large capacity implies that the codeword set is such that a large margin can be achieved for all classes. A small capacity implies that there is at least one class for which the largest achievable margin is small. The optimal codeword set is that of largest capacity.

Definition 7. *$\mathcal{Y}^* \in \mathcal{S}(M,d)$ is a codeword set of maximum capacity if*

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y} \in \mathcal{S}(M,d)} \mathcal{C}[\mathcal{Y}]. \tag{IV.30}$$

IV.E.2 Maximum capacity codeword sets

In this section, we study the properties of the margin capacity. We start by deriving an upper bound on the margin achievable along any direction of largest margin, ξ^k, (IV.29).

Theorem 1. *Let $\mathcal{Y} \in \mathcal{S}(M,d)$ be a codeword set with directions of largest margin $\xi^k, k \in \{1,\ldots,M\}$. Then, $\forall k$*

$$\mathcal{M}(y^k, \xi^k) \leq \frac{M}{2(M-1)}, \tag{IV.31}$$

with equality if and only if

$$\langle y^l, y^k \rangle = -\frac{1}{M-1} \;\; \forall l \neq k. \tag{IV.32}$$

In this case, $\xi^k = y^k$.

Proof. Consider any direction v such that $||v|| = 1$. Using (IV.9), (IV.27) and the

fact that the minimum cannot be larger than the average

$$\begin{aligned}
\mathcal{M}(y^k, v) = \min_{l \neq k} \frac{1}{2} \left\langle y^k - y^l, v \right\rangle &\leq \frac{1}{2(M-1)} \sum_{l \neq k} \left\langle y^k - y^l, v \right\rangle \\
&= \frac{1}{2(M-1)} \left\langle \sum_{l \neq k} (y^k - y^l), v \right\rangle \\
&= \frac{1}{2(M-1)} \left\langle (M-1) y^k - \sum_{l \neq k} y^l, v \right\rangle \\
&= \frac{1}{2(M-1)} \left\langle M y^k - \sum_{l=1}^{M} y^l, v \right\rangle \\
&= \frac{M}{2(M-1)} \left\langle y^k, v \right\rangle . \qquad \text{(IV.33)}
\end{aligned}$$

Equality holds if and only if, for all $l \neq k$

$$\frac{1}{2} \left\langle y^k - y^l, v \right\rangle = \min_{l \neq k} \frac{1}{2} \left\langle y^k - y^l, v \right\rangle = \frac{M}{2(M-1)} \left\langle y^k, v \right\rangle , \qquad \text{(IV.34)}$$

i.e. if and only if, for all $l \neq k$,

$$\left\langle y^l, v \right\rangle = -\frac{1}{M-1} \left\langle y^k, v \right\rangle . \qquad \text{(IV.35)}$$

It follows from (IV.29) that

$$\mathcal{M}(y^k, \xi^k) \leq \frac{M}{2(M-1)} \left\langle y^k, \xi^k \right\rangle , \qquad \text{(IV.36)}$$

with equality if and only if, for all $l \neq k$

$$\left\langle y^l, \xi^k \right\rangle = -\frac{1}{M-1} \left\langle y^k, \xi^k \right\rangle . \qquad \text{(IV.37)}$$

Hence,

$$\mathcal{M}(y^k, \xi^k) = \frac{M}{2(M-1)}, \qquad \text{(IV.38)}$$

if and only if $\xi^k = y^k, \forall k$ and

$$\left\langle y^l, y^k \right\rangle = -\frac{1}{M-1} \forall l \neq k. \qquad \text{(IV.39)}$$

■

We next derive the necessary and sufficient conditions for optimality of a codeword set.

Theorem 2. *Let $\mathcal{Y}$ be any codeword set in $\mathcal{S}(M,d)$. Then*

$$\mathcal{C}[\mathcal{Y}] \leq \frac{M}{2(M-1)}. \tag{IV.40}$$

The left and side of (IV.40) is denoted the capacity bound of $\mathcal{S}(M,d)$. This bound is met with equality if and only if

$$\left\langle y^k, y^l \right\rangle = -\frac{1}{M-1} \quad \forall k,l,k \neq l. \tag{IV.41}$$

In this case, the directions of largest margin are $\xi^k = y^k, \forall k$.

Proof. The bound of (IV.40) is a straightforward consequence of (IV.28) and (IV.31). To prove the equality conditions assume that (IV.41) holds and $\xi^k = y^k$. Then, $\forall l \neq k$,

$$\begin{aligned} \left\langle y^l, \xi^k \right\rangle &= \left\langle y^l, y^k \right\rangle \\ &= -\frac{1}{M-1} \\ &= -\frac{1}{M-1} \left\langle y^k, y^k \right\rangle \\ &= -\frac{1}{M-1} \left\langle y^k, \xi^k \right\rangle, \end{aligned} \tag{IV.42}$$

where we have used the fact that $||y^k|| = 1$. It follows, from Theorem 1, that

$$\mathcal{M}(y^k, \xi^k) = \frac{M}{2(M-1)}, \tag{IV.43}$$

From (IV.28),

$$\mathcal{C}[\mathcal{Y}] = \frac{M}{2(M-1)}. \tag{IV.44}$$

Conversely, assume that

$$\mathcal{C}[\mathcal{Y}] = \frac{M}{2(M-1)}, \tag{IV.45}$$

holds. Then, from (IV.28) and (IV.31), there is a set of largest margin directions ξ^k such that

$$\mathcal{M}(y^k, \xi^k) = \frac{M}{2(M-1)} \ \forall k. \tag{IV.46}$$

From Theorem 1, these are the directions $\xi^k = y^k$ and

$$\langle y^l, y^k \rangle = -\frac{1}{M-1} \quad \forall k, l \neq k. \tag{IV.47}$$

■

The theorem shows that, if there is a codeword set that meets the capacity bound, the codewords in this set are also the directions of largest margin. We next derive the conditions under which such a set of codewords exists.

Theorem 3. *$\mathcal{S}(M,d)$ contains a set of codewords $\mathcal{Y}^c(M,d)$ that meets the capacity bound if and only if $d \geq M-1$. In this case, the codewords in $\mathcal{Y}^c(M,d)$ are the vertices of a regular simplex in $\mathbb{R}^d$.*

Proof. We start by recalling that $\mathcal{Y}^c(M,d)$ is a set of M centered, unit norm, d-dimensional vectors y^k, such that

$$\langle y^k, y^l \rangle = -\frac{1}{M-1}, \quad \forall k, l \neq k. \tag{IV.48}$$

The proof is by construction and uses a known method for the design of regular simplexes [12]. Let y^k be the codewords in $\mathcal{Y}^c(M,d)$. Without loss of generality we can set $y^1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^d$. From (IV.48) it follows that

$$y_1^k = -\frac{1}{M-1} \quad \forall k > 1, \tag{IV.49}$$

where y_i^k is the ith coordinate of vector y^k. Defining

$$\bar{y}^k = \frac{M-1}{\sqrt{M(M-2)}}[y_2^{k+1}, \ldots, y_d^{k+1}] \in \mathbb{R}^{d-1} \quad k = 1, \ldots, M-1, \tag{IV.50}$$

it follows that $\sum_{k=1}^{M-1} \bar{y}^k = 0$,

$$\begin{aligned} \|\bar{y}^k\|^2 &= \frac{(M-1)^2}{M(M-2)}[\|y^{k+1}\|^2 - [y_1^{k+1}]^2] \\ &= \frac{(M-1)^2}{M(M-2)}\left[1 - \frac{1}{(M-1)^2}\right] \\ &= 1 \quad \forall k, \end{aligned} \tag{IV.51}$$

and

$$\begin{aligned}\langle \bar{y}^k, \bar{y}^l \rangle &= \frac{(M-1)^2}{M(M-2)} \left[\langle y^{k+1}, y^{l+1} \rangle - y_1^{k+1} y_1^{l+1} \right] \\ &= \frac{(M-1)^2}{M(M-2)} \left[-\frac{1}{M-1} - \frac{1}{(M-1)^2} \right] \\ &= -\frac{(M-1)^2}{M(M-2)} \frac{M}{(M-1)^2} = -\frac{1}{M-2} \quad \forall k, l \neq k. \end{aligned} \tag{IV.52}$$

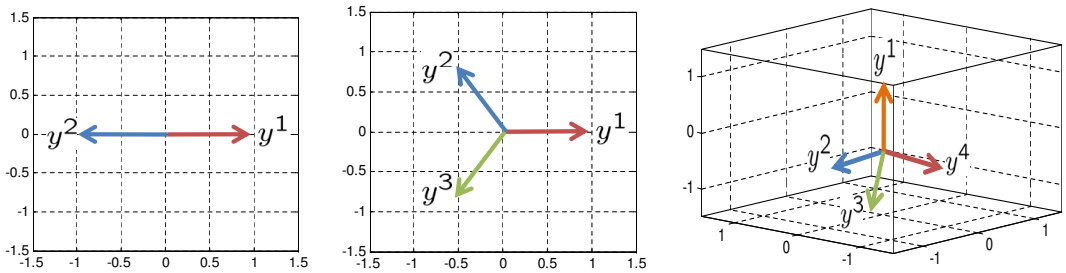
Hence, the codewords $\bar{y}^k$ are the elements of $\mathcal{Y}^c(M-1, d-1)$, the codeword set that meets the capacity bound of $\mathcal{S}(M-1, d-1)$. In summary, the application of the simplex design procedure of (IV.50) to a codeword set $\mathcal{Y}^c(M, d)$ produces a codeword set $\mathcal{Y}^c(M-1, d-1)$.

If $d \leq M-2$, the procedure can be applied $d-1$ times, to produce a codeword set $\mathcal{Y}^c(M-d+1, 1)$. This is a set of $M-d+1$ centered, unit norm, scalars that satisfy (IV.48). It follows from the unit norm constraint that $\bar{y}^k \in \{+1, -1\}$ and thus, for $k \neq l$, $\langle \bar{y}^k, \bar{y}^l \rangle = -1$. This, however, contradicts (IV.48), since $-\frac{1}{M-d+1-1} = -\frac{1}{M-d} \geq -\frac{1}{2}$. Hence, $\mathcal{S}(M, d)$ contains no set of codewords that meets the capacity bound, when $d \leq M-2$.

If $d \geq M-1$, the procedure can be applied $M-2$ times, to produce a codeword set $\mathcal{Y}^c(2, d-M+2)$. This is a set of 2 centered, unit norm, $(d-M+2)$-dimensional vectors that satisfy (IV.48), i.e. $\langle \bar{y}^1, \bar{y}^2 \rangle = -1$. Since $\bar{y}^1 = [1, 0...0]$ and $\bar{y}^2 = [-1, 0...0]$ in $\mathbb{R}^{d-M+2}$ satisfy these conditions, there exists a sequence $\mathcal{Y}^c(2, d-M+2), \ldots, \mathcal{Y}^c(M, d)$ of codeword sets that meet the capacity bounds of $\mathcal{S}(2, d-M+2), \ldots, \mathcal{S}(M, d)$, respectively. Since the procedure used to design this sequence is the regular simplex design procedure of (IV.50), the codewords in $\mathcal{Y}^c(2, d-M+2), \ldots, \mathcal{Y}^c(M, d)$ form a regular simplex in $\mathbb{R}^{d-M+2}, \ldots, \mathbb{R}^d$, respectively. ∎

The following corollary is a straightforward consequence of Theorems 1 and 3.

Corollary 3. *Let $\mathcal{Y}^*$ be a codeword set of maximum capacity in $\mathcal{S}(M, d)$. If $d \geq M-1$, the codewords $(y^*)^k$ of $\mathcal{Y}^*$ are the vertices of a regular simplex in $\mathbb{R}^d$. In*



$(M = 2, d = 1)$ $(M = 3, d = 2)$ $(M = 4, d = 3)$

Figure IV.1 Codewords of maximal capacity for $M = 2, 3, 4$.

this case, the directions of largest margin are $\xi^k = (y^*)^k$,

$$\left\langle (y^*)^k, (y^*)^l \right\rangle = -\frac{1}{M-1} \quad \forall k, l \neq k, \tag{IV.53}$$

and thus

$$\mathcal{C}[\mathcal{Y}^*] = \frac{M}{2(M-1)}. \tag{IV.54}$$

This corollary characterizes the optimal set of codewords of $\mathcal{S}(M, d)$, whenever $d \geq M - 1$. It also shows that, under this constraint, the capacity bound of (IV.40) is achieved. Since this maximal capacity only depends on the number of classes M, not in the dimension d, and the complexity of the maximum margin decision rule of (IV.13) is linear in d, there is no benefit in adopting a dimension larger than $M - 1$. Hence, when the classification problem has no further constraints, it is natural to rely on codeword sets of dimension

$$d = M - 1. \tag{IV.55}$$

Figure IV.1 presents the optimal codeword sets for various M. Note that in the binary case, $M = 2$, the optimal codewords are the classical $\{+1, -1\}$ labels. A Matlab script that determines the optimal codeword set for any M is available from [2].

IV.E.3 Low-dimensional predictors

When $d < M-1$, there is no guarantee that a codeword set of maximum capacity will achieve the capacity bound. Nevertheless, the choice of $d < M-1$ can be appealing for applications where it is critical to use low-dimensional predictors. For example, the design of a M-ary classifier with a two-dimensional predictor. In this case, the search for the codeword set of maximum capacity requires, according to (IV.28), the solution of

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y}\in\mathcal{S}(M,d)} \min_{k=1,\ldots,M} \max_{||v||=1} \min_{l\neq k} \left[\left\langle y^k, v\right\rangle - \left\langle y^l, v\right\rangle\right]. \tag{IV.56}$$

This, however, is a very non-trivial optimization for which, to the best of our knowledge, there are no efficient algorithms. Hence, it is of interest to consider alternative optimality criteria. One possibility is the maxmin codeword distance criterion.

Definition 8. *Let $\mathcal{Y}$ be a codeword set in $\mathcal{S}(M,d)$. The minimum distance of $\mathcal{Y}$ is*

$$d_{min}[\mathcal{Y}] = \min_{k,l\neq k} \|y^k - y^l\|^2. \tag{IV.57}$$

$\mathcal{Y}^*$ is a maxmin distance codeword set in $\mathcal{S}(M,d)$ if

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y}\in\mathcal{S}(M,d)} d_{min}[\mathcal{Y}]. \tag{IV.58}$$

The problem of (IV.58) is equivalent to determining the maximum diameter of M equal circles that can be placed on the surface of the unit sphere without overlap. This is known as the *Tammes* problem [85], and does not have closed-form solution in general. The solution is also not unique, since any rotation of a valid solution is a valid solution. However, its numerical solution is much simpler than that of (IV.56). The maximization of $d_{min}[\mathcal{Y}]$ is also intuitive. From (IV.14), the decision rule of the maximum margin classifier is based on the projections $\left\langle f, y^k\right\rangle$ of the predictor f along the codewords. If the codewords are similar, the same

will hold for the projections, resulting in lower margins. In fact, it can be shown that the optimal codeword sets under the optimality criteria of Definitons 7 and 8 are identical when $d > M - 1$. For this, we start by showing that there is a close relationship between the capacity and the minimum distance of any codeword set in $\mathcal{S}(M, d)$.

Lemma 2. *Let $\mathcal{Y}$ be a codeword set in $\mathcal{S}(M, d)$. Then*

$$\frac{1}{4} d_{min}[\mathcal{Y}] \leq \mathcal{C}[\mathcal{Y}] < \frac{1}{2} d_{min}[\mathcal{Y}] \tag{IV.59}$$

and

$$d_{min}[\mathcal{Y}] \leq \frac{2M}{M-1}. \tag{IV.60}$$

Proof. We start with (IV.59). The left inequality follows from (IV.28), (IV.29) and (IV.9), since

$$\begin{aligned}
\mathcal{C}[\mathcal{Y}] &= \min_{k=1\ldots M} \max_{||v||=1} \mathcal{M}(y^k, v) && \text{(IV.61)}\\
&\geq \min_{k=1\ldots M} \mathcal{M}(y^k, y^k) && \text{(IV.62)}\\
&= \frac{1}{2} \min_{k=1\ldots M} \min_{l\neq k} \left[\|y^k\| - \left\langle y^k, y^l \right\rangle\right] && \text{(IV.63)}\\
&= \frac{1}{4} \min_{k=1\ldots M} \min_{l\neq k} \left[2\|y^k\| - 2\left\langle y^k, y^l \right\rangle\right] && \text{(IV.64)}\\
&= \frac{1}{4} \min_{k=1\ldots M} \min_{l\neq k} \left[\|y^k\| + \|y^l\| - 2\left\langle y^k, y^l \right\rangle\right] && \text{(IV.65)}\\
&= \frac{1}{4} \min_{k,l\neq k} \|y^k - y^l\|^2. && \text{(IV.66)}
\end{aligned}$$

The right inequality follows from (IV.9) since, for any v such that $||v|| = 1$,

$$\begin{aligned}
\mathcal{M}(y^k, v) &= \frac{1}{2} \min_{l\neq k} \left\langle y^k - y^l, v \right\rangle && \text{(IV.67)}\\
&\leq \frac{1}{2} \min_{l\neq k} \|y^k - y^l\|^2, && \text{(IV.68)}
\end{aligned}$$

and thus

$$\begin{aligned}
\mathcal{C}[\mathcal{Y}] &= \min_{k=1\ldots M} \max_{||v||=1} \mathcal{M}(y^k, v) && \text{(IV.69)}\\
&\leq \min_{k=1\ldots M} \max_{||v||=1} \frac{1}{2} \min_{l\neq k} \|y^k - y^l\|^2 && \text{(IV.70)}\\
&= \frac{1}{2} \min_{k,l\neq k} \|y^k - y^l\|^2. && \text{(IV.71)}
\end{aligned}$$

(IV.60) follows from the left inequality of (IV.59) and (IV.40). ∎

We next use this result to show that, when $d \geq M-1$, the codeword sets of maximum capacity and maxmin distance are identical.

Theorem 4. *Let $\mathcal{Y}^*$ be a codeword set of maximum margin capacity in $\mathcal{S}(M,d)$. If $d \geq M-1$ then $\mathcal{Y}^*$ is a codeword set of maximum capacity, i.e.*

$$\mathcal{C}[\mathcal{Y}^*] = \frac{M}{2(M-1)}, \tag{IV.72}$$

if and only if $\mathcal{Y}^*$ is a codeword set of maxmin distance, i.e.

$$d_{min}[\mathcal{Y}^*] = \frac{2M}{M-1}. \tag{IV.73}$$

Proof. Let $\mathcal{Y}^*$ be a codeword set of maximum capacity. From (IV.57) and (IV.53)

$$\begin{aligned} d_{min}[\mathcal{Y}^*] &= \min_{k,l\neq k} \|(y^*)^k - (y^*)^l\|^2 && \text{(IV.74)} \\ &= \min_{k,l\neq k} \left[2 - 2\left\langle (y^*)^k, (y^*)^l \right\rangle\right] && \text{(IV.75)} \\ &= \min_{k,l\neq k} \left[2 + \frac{2}{M-1}\right] && \text{(IV.76)} \\ &= \frac{2M}{M-1} && \text{(IV.77)} \end{aligned}$$

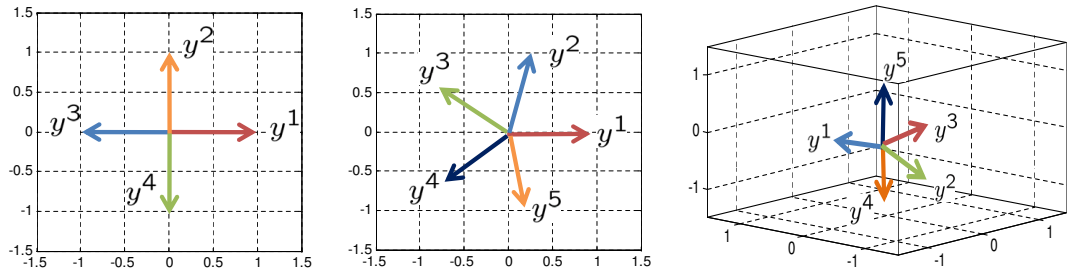
and $\mathcal{Y}^*$ meets the bound of (IV.60). Hence, it is a maxmin distance codeword set. Let $\mathcal{Y}^*$ be a codeword set of maxmin distance, i.e.

$$d_{min}[\mathcal{Y}^*] = \frac{2M}{M-1}. \tag{IV.78}$$

From (IV.59) it follows that

$$\mathcal{C}[\mathcal{Y}^*] \geq \frac{M}{2(M-1)}, \tag{IV.79}$$

and $\mathcal{Y}^*$ meets the capacity bound of (IV.40). Hence, it is a codeword set of maximum capacity. ∎



$(M = 4, d = 2)$ $(M = 5, d = 2)$ $(M = 5, d = 3)$

Figure IV.2 maxmin codeword sets for several values of d and M.

In summary, in the regime of $d \geq M - 1$, the vertices of a regular simplex in $\mathbb{R}^d$ are both a maximum capacity and a maxmin distance codeword sets of $\mathcal{S}(M, d)$. These codeword sets achieve both the capacity and maxmin distance bounds of $\mathcal{S}(M, d)$. The main difference between the two optimality criteria is the difficulty of finding an optimal solution for $d < M - 1$. While the optimization problem of (IV.56) is difficult, there are many algorithms for the solution of (IV.58). A Matlab implementation of one of these algorithms, based on a barrier method [58], is available from [2]. Figure IV.2 presents maxmin codewords sets for different values of M and d.

IV.F Multiclass losses

Given a set of codewords $\mathcal{Y}$, a multiclass predictor is learned by minimizing a risk derived from a loss function. In this section, we introduce a family of margin losses for multiclass classification.

IV.F.1 Risk

Let $\mathcal{Y}$ be a set of codewords, x an example from the class whose codeword is y, and $L_M[y, f(x)]$ the loss of prediction $f(x)$ for example x. A predictor is optimal if it minimizes the multiclass classification risk

$$R_M(f) \quad = \quad E_{X,C}\{L_M[y^c, f(x)]\}, \tag{IV.80}$$

For classifier design, this is approximated by an empirical estimate

$$\overline{R}_M(f) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} L_M[y^{c_i}, f(x_i)], \tag{IV.81}$$

derived from a training sample $\mathcal{D} = (x_i, c_i)_{i=1}^n$. If the minimization of (IV.81) encourages predictors for which the margin of (IV.11) is large, $L_M[.,.]$ is denoted a margin loss. This property guarantees that the optimal predictor $f^*(.)$ has good generalization, i.e. good small sample performance. If it it is possible to recover class posterior probabilities $P_{C|X}(c|x), \quad C = 1 \ldots M$, for all x from the predictions $f^*(x)$, $L_M[.,.]$ is denoted a proper loss. This property is important for applications that require a confidence score for the classification.

In this work, we seek multiclass proper margin losses. As in the binary case of (IV.6), it is natural to start from functions of the margin, now defined in (IV.9). However, in the M-ary case this leads to a very non-linear function of the predictor $f(.)$. To simplify the minimization of the empirical risk, we consider a family of losses that approximate the discontinuous min operator of (IV.9) by a continuous function.

Definition 9. *Let $\mathcal{Y} = \{y^1, \ldots, y^M\}$ be a set of codewords and $f(x) : \mathcal{X} \to \mathbb{R}^M$ a predictor. A loss*

$$L_M^{\phi}[y^c, f(x)] = \sum_{l=1, l\neq c}^{M} \phi\left(\frac{1}{2}\left[\langle f(x), y^c\rangle - \langle f(x), y^l\rangle\right]\right), \tag{IV.82}$$

where $\phi : \mathbb{R} \to \mathbb{R}^+$ is a strictly positive function, $\phi(x) > 0 \, \forall x$, is denoted a ϕ-loss.

We next investigate under which conditions these losses are margin enforcing and proper.

IV.F.2 Margin losses

We start by considering the relationship between empirical risk minimization and predictor margins.

Theorem 5. *Consider a training set $\mathcal{D}$, a codeword set $\mathcal{Y} = \{y^1, \ldots, y^M\}$, and a predictor $f(x)$. Let $\overline{R}_{L_M^\phi}(f)$ be the empirical risk of (IV.81) for the ϕ-loss of (IV.82). Then,*

$$\forall x \qquad L_M^\phi[y^c, f(x)] > \phi[\mathcal{M}(y^c, f(x))] \tag{IV.83}$$

$$\overline{R}_{L_M^\phi}(f) > \frac{1}{n}\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})], \tag{IV.84}$$

where $\mathcal{M}$ and $\mathcal{M}_p$ are defined in (IV.9), (IV.11) respectively.

Proof. To prove (IV.83), let

$$l^* = \arg\min_{l \neq c} \frac{1}{2}\left[\langle y^c, f(x)\rangle - \langle y^l, f(x)\rangle\right]. \tag{IV.85}$$

Then

$$\begin{aligned} L_M^\phi[y^c, f(x)] &= \phi\left(\frac{1}{2}\left[\langle y^c, f(x)\rangle - \langle y^{l^*}, f(x)\rangle\right]\right) \\ &+ \sum_{l \neq c, l^*} \phi\left(\frac{1}{2}\left[\langle y^c, f(x)\rangle - \langle y^l, f(x)\rangle\right]\right), \end{aligned} \tag{IV.86}$$

and it follows from (IV.9) that

$$L_M^\phi[y^c, f(x)] = \phi[\mathcal{M}(y^c, f(x))]\left[1 + \sum_{l \neq c, l^*} \frac{\phi(\frac{1}{2}[\langle y^c, f(x)\rangle - \langle y^l, f(x)\rangle])}{\phi[\mathcal{M}(y^c, f(x))]}\right].$$

The inequality of (IV.83) follows from the fact that $\phi(.)$ is strictly positive.

To prove (IV.84) we start by using (IV.83)

$$\begin{aligned} \overline{R}_{L_M^\phi}(f) &= \frac{1}{n}\sum_{i=1}^{n} L_M^\phi[y^{c_i}, f(x_i)] \\ &> \frac{1}{n}\sum_{i=1}^{n} \phi[\mathcal{M}(y^{c_i}, f(x_i))], \end{aligned} \tag{IV.87}$$

and let $i^* = \arg\min_i \mathcal{M}(y^{c_i}, f(x_i))$. From (IV.11) and the strict positivity of $\phi(.)$

$$\begin{aligned} \overline{R}_{L_M^\phi}(f) &> \frac{1}{n}\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]\left[1 + \sum_{i \neq i^*} \frac{\phi[\mathcal{M}(y^{c_i}, f(x_i))]}{\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]}\right] \\ &> \frac{1}{n}\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]. \end{aligned} \tag{IV.88}$$

■

It follows from (IV.84) that the minimization of the empirical risk encourages small values of $\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]$. If $\phi(.)$ is decreasing, i.e. $\phi'(x) \leq 0 \, \forall x$, this encourages large predictor margins and $L_M^\phi[.,.]$ is a margin loss. In summary, if $\phi(.)$ is a decreasing function, $L_M^\phi[.,.]$ is a margin loss.

IV.F.3 Convexity

We next investigate the conditions under which ϕ-losses lead to convex risks. For simplicity, we will adopt the notation

$$\eta_k(x) = P_{C|X}(k|x), \tag{IV.89}$$

for the posterior class probabilities and

$$u^k(x) = \frac{1}{2} \left\langle y^k, f(x) \right\rangle, \quad k = 1, \ldots, M. \tag{IV.90}$$

for associated codeword projections. We start by noting that, to minimize (IV.80), it suffices to determine the predictor $f^*(x)$ of minimum conditional risk

$$R_M(f|x) \;\; = \;\; E_{C|X}\{L_M[y^c, f(x)]|x\}, \tag{IV.91}$$

for all x. The following theoremm characterizes the convexity of the conditional risk of a ϕ-loss.

Theorem 6. *Let $\mathcal{Y} = \{y^1, \ldots, y^M\} \in \mathbb{R}^d$ be a codeword set, $f(x) : \mathcal{X} \to \mathbb{R}^d$ a predictor with the codeword projections of (IV.90), $\eta_k(x)$ the posterior probabilities of (IV.89), and $R_M^\phi(f|x)$ the conditional risk of (IV.91) when $L_M[.,.]$ is a ϕ-loss, as in (IV.82), and ϕ is twice differentiable. Then*

$$R_M^\phi(f|x) \;\; = \;\; \sum_{k, l \neq k} \eta_k \phi(u^k - u^l), \tag{IV.92}$$

is a convex function of f if ϕ is strictly convex. Its unique minimizer f^* is the solution of

$$\mathbf{Y}\mathbf{Q}_{f^*}^\phi \eta = 0, \tag{IV.93}$$

where $\mathbf{Y} \in \mathbb{R}^{d\times M}$ *is the matrix whose columns are the codewords* y^k, $\eta \in \mathbb{R}^M$ *is the posterior probability vector,* $\mathbf{Q}_f^\phi \in \mathbb{R}^{M\times M}$ *with*

$$\mathbf{Q}_f^\phi(k,l) = \begin{cases} -\phi'(u^l - u^k) & k \neq l \\ \sum_{j=1, j\neq k}^M \phi'(u^k - u^j) & k = l, \end{cases} \tag{IV.94}$$

and we have omitted the dependency of all terms on x *for notational simplicity.*

Proof. From (IV.91),

$$\begin{aligned} R_M^\phi(f|x) &= \sum_{k=1}^M \eta_k(x) L_M[y^k, f(x)] \\ &= \sum_{k=1}^M \eta_k(x) \sum_{l=1, l\neq k}^M \phi(u^k(x) - u^l(x)). \end{aligned} \tag{IV.95}$$

The derivative of $R_M^\phi(f|x)$ with respect to $f(x)$ is

$$\begin{aligned} \frac{\partial R_M^\phi(f|x)}{\partial f(x)} &= \frac{\partial}{\partial f(x)} \sum_{l,k|k\neq l} \eta_k \phi(u^k - u^l) \\ &= \frac{1}{2} \sum_{l,k|k\neq l} \eta_k \phi'(u^k - u^l)[y^k - y^l] \\ &= \frac{1}{2} \sum_{l,k|k\neq l} y^k \eta_k \phi'(u^k - u^l) - \frac{1}{2} \sum_{l,k|k\neq l} y^l \eta_k \phi'(u^k - u^l) \\ &= \frac{1}{2} \sum_{j,k|k\neq j} y^k \eta_k \phi'(u^k - u^j) - \frac{1}{2} \sum_{l,k|k\neq l} y^k \eta_l \phi'(u^l - u^k) \\ &= \frac{1}{2} \sum_{k=1}^M y^k \eta_k \sum_{j=1, j\neq k}^M \phi'(u^k - u^j) - \frac{1}{2} \sum_{k=1}^M y^k \sum_{l=1, l\neq k}^M \eta_l \phi'(u^l - u^k) \\ &= \frac{1}{2} \sum_{k=1}^M y^k \left[\eta_k \sum_{j=1, j\neq k}^M \phi'(u^k - u^j) - \sum_{l=1, l\neq k}^M \eta_l \phi'(u^l - u^k) \right] \\ &= \frac{1}{2} \mathbf{Y} \mathbf{Q}_f^\phi \eta. \end{aligned} \tag{IV.96}$$

Hence, (IV.93) holds for any minimizer of $R_M^\phi(f|x)$. The second order derivative of $R_M^\phi(f|x)$ with respect to $f(x)$ is

$$\begin{aligned} \frac{\partial^2 R_M^\phi(f|x)}{\partial f(x)^2} &= \frac{\partial}{\partial f(x)} \sum_{l,k|k\neq l} \eta_k \phi'(u^k - u^l)[y^k - y^l] \\ &= \sum_{l,k|k\neq l} \eta_k \phi''(u^k - u^l)[y^k - y^l][y^k - y^l]^T. \end{aligned} \tag{IV.97}$$

If ϕ is strictly convex, then $\phi'' > 0$. Since the codewords are different, i.e. $y^k - y^l \neq 0 \; \forall k, l$, the matrices $[y^k - y^l][y^k - y^l]^T$ are positive definite $\forall k, l$. Since $\eta_k \geq 0 \quad \forall k$ and $\sum_{k=1}^{M} \eta_k = 1$, it follows that $\eta_j > 0$ for at least one j. Hence, (IV.97) is strictly positive definite, $R_M^{\phi}(f|x)$ strictly convex, and has a unique minimum. ∎

IV.F.4 Proper ϕ-losses

The previous theorem shows that, as long as ϕ is strictly convex, $R_M^{\phi}(f|x)$ has a unique minimizer, f^*. We next consider the conditions under which the posterior probability vector η can be recovered from f^*, i.e. L_M^{ϕ} is proper. In this and the following results, we use $|\mathcal{A}|$ to denote the dimensionality of a space $\mathcal{A}$, and $Rank(\mathbf{A}), Null(\mathbf{A})$, and $Range(\mathbf{A})$ to denote the rank, null space, and column space of matrix $\mathbf{A}$, respectively. From (IV.93), it follows that η can be recovered uniquely only when $Null(YQ_{f^*}^{\phi})$ is a one-dimensional space that contains a probability vector. The following lemma gives sufficient conditions for the first property to hold.

Lemma 3. *Let $\mathbf{Y} \in \mathbb{R}^{d \times M}$ be a matrix whose columns are the codewords in a set $\mathcal{Y} \in \mathcal{S}(M, d)$, (IV.27), $\eta_k(x)$ the posterior probabilities of (IV.89), and $f^*(x)$ the solution of (IV.93) when $Q_{f^*}^{\phi}$ is as defined in (IV.94).*

1. *If $Rank(\mathbf{Y}) \leq M - 2$ then $|Null(YQ_{f^*}^{\phi})| \geq 2$.*
2. *If $Rank(\mathbf{Y}) = M - 1$ then $Null(YQ_{f^*}^{\phi}) = Null(Q_{f^*}^{\phi})$.*
3. *If ϕ is strictly decreasing then $|Null(Q_{f^*}^{\phi})| = 1$.*

Proof. We start by assuming that $Rank(\mathbf{Y}) < M-1$ and consider two possibilities.

1. The set $Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^{\phi})$ is empty. In this case, since

$$Null(\mathbf{Y}) \bigcup Range(\mathbf{Q}_f^{\phi}) \subset \mathbb{R}^M$$

and

$$\begin{aligned} M &\geq |Null(\mathbf{Y})| + |Range(\mathbf{Q}_f^\phi)| \\ &= M - Rank(\mathbf{Y}) + |Range(\mathbf{Q}_f^\phi)| \\ &\geq 2 + |Range(\mathbf{Q}_f^\phi)|. \end{aligned} \tag{IV.98}$$

Since $\mathbf{Q}_f^\phi \in \mathbb{R}^{M\times M}$, it follows that $|Null(\mathbf{Q}_f^\phi)| \geq 2$. Since $Null(\mathbf{Q}_f^\phi) \subset Null(\mathbf{Y}\mathbf{Q}_f^\phi)$, it follows that $|Null(\mathbf{Y}\mathbf{Q}_f^\phi)| \geq 2$.

2. The set $Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^\phi)$ is non-empty. Hence, there is at least one vector $v_1 \in Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^\phi)$. Since $v_1 \in Range(\mathbf{Q}_f^\phi)$, there exists a vector v_2 such that $\mathbf{Q}_f^\phi v_2 = v_1$, i.e. $v_2 \notin Null(\mathbf{Q}_f^\phi)$. Since $v_1 \in Null(\mathbf{Y})$, it follows that $\mathbf{Y}\mathbf{Q}_f^\phi v_2 = 0$ and $v_2 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. On the other hand, it follows from (IV.94) that the rows of $\mathbf{Q}_f^\phi$ sum to zero and

$$|Null(\mathbf{Q}_f^\phi)| \geq 1. \tag{IV.99}$$

Hence, there is at least a vector $v_0 \neq 0$ such that $\mathbf{Q}_f^\phi v_0 = 0$. It follows that $v_0 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. In summary, there is a vector $v_0 \in Null(\mathbf{Q}_f^\phi)$ and a vector $v_2 \notin Null(\mathbf{Q}_f^\phi)$ such that $v_0, v_2 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. Hence, $|Null(\mathbf{Y}\mathbf{Q}_f^\phi)| \geq 2$.

Statement 1. of the lemma follows from the combination of the two possibilities.

To prove statement 2. assume that $Rank(\mathbf{Y}) = M-1$. Since $\mathbf{Y} \in \mathbb{R}^{d\times M}$, $|Null(\mathbf{Y})| = 1$. Since the codewords are centered, as in (IV.27), $\mathbf{Y}\mathbf{1} = 0$ and $Null(\mathbf{Y}) = Range(\mathbf{1})$, i.e. the null space of $\mathbf{Y}$ is spanned by the vector $\mathbf{1}$. Hence, for any $\eta \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$ there is a scalar λ such that

$$\mathbf{Q}_f^\phi \eta = \lambda \mathbf{1}. \tag{IV.100}$$

Denoting by r_Q^i the ith row of $\mathbf{Q}_f^\phi$, it follows that $< r_Q^i, \eta >= \lambda$ and $< \sum_{i=1}^M r_Q^i, \eta >= M\lambda$. Since, from (IV.94), $\sum_{i=1}^M r_Q^i = 0$ it follows that $\lambda = 0$. Hence, $\eta \in Null(\mathbf{Q}_f^\phi)$ and $Null(\mathbf{Y}\mathbf{Q}_f^\phi) \subset Null(\mathbf{Q}_f^\phi)$. Statement 2. follows from the fact that $Null(\mathbf{Q}_f^\phi) \subset Null(\mathbf{Y}\mathbf{Q}_f^\phi)$ always holds.

To prove statement 3. let $\overline{\mathbf{Q}}_f^{\phi} \in \mathbb{R}^{M-1\times M-1}$ be the matrix obtained by eliminating the first row and column of $\mathbf{Q}_f^{\phi}$. From (IV.94),

$$|\mathbf{Q}_f^{\phi}(k,k)| = \left|\sum_{j=1,j\neq k}^{M} \phi'(u^k - u^j)\right|, \tag{IV.101}$$

and

$$\begin{aligned} |\overline{\mathbf{Q}}_f^{\phi}(k,k)| &= \sum_{j=1,j\neq k+1}^{M} |\phi'(u^{k+1} - u^j)| \\ &= \sum_{j=1,j\neq k+1}^{M} |\mathbf{Q}_f^{\phi}(j,k+1)| \\ &= |\mathbf{Q}_f^{\phi}(1,k+1)| + \sum_{j=1,j\neq k}^{M-1} |\overline{\mathbf{Q}}_f^{\phi}(j,k)| \end{aligned} \tag{IV.102}$$

If ϕ is strictly decreasing, it follows that

$$|\overline{\mathbf{Q}}_f^{\phi}(k,k)| > \sum_{j=1,j\neq k}^{M-1} |\overline{\mathbf{Q}}_f^{\phi}(j,k)|, \tag{IV.103}$$

i.e. $\overline{\mathbf{Q}}_f^{\phi}$ is strictly diagonally dominant and thus non-singular [34]. It follows that rows $r_Q^i, i = 2,\ldots,M$ of $\mathbf{Q}_f^{\phi}$ are linearly independent and $Rank(\mathbf{Q}_f^{\phi}) \geq M-1$. Since, from (IV.94),

$$r_Q^1 = -\sum_{i=2}^{M} r_Q^i \tag{IV.104}$$

it follows that $Rank(\mathbf{Q}_f^{\phi}) = M-1$. Hence, $|Null(\mathbf{Q}_f^{\phi})| = 1$. ∎

The lemma shows that, if $Rank(\mathbf{Y}) = M-1$ and ϕ is strictly decreasing, the set of vectors η for which (IV.93) holds is one-dimensional. It remains to verify if this set contains a probability vector, i.e. a vector of non-negative entries that sum to one. The following lemma provides the conditions under which this holds.

Lemma 4. *Let $\mathbf{Q}_f^{\phi}$ be as defined in (IV.94) and ϕ differentiable. Then $\mathbf{Q}_f^{\phi}\eta = 0$ if and only if*

$$\eta_k = \frac{\sum_{l=1}^{M} \eta_l \phi'(u^l - u^k)}{\sum_{l=1}^{M} \phi'(u^k - u^l)}, \quad \forall k. \tag{IV.105}$$

If ϕ is also twice differentiable, strictly decreasing, and strictly convex, then any solution η has the following properties

1. *if $u^k \geq u^j$ then $\eta_k \geq \eta_j$, with equality of the probabilities only if $u^k = u^j$.*

2. *if $\eta \neq 0$ then all $\eta_k \neq 0$ have the same sign and $\sum_k \eta_k \neq 0$.*

Proof. From (IV.94), $\mathbf{Q}_f^\phi \eta = 0$ when, for all k

$$\begin{aligned} 0 &= \eta_k \sum_{j=1, j\neq k}^{M} \phi'(u^k - u^j) - \sum_{l=1, l\neq k}^{M} \eta_l \phi'(u^l - u^k) \\ &= \eta_k \sum_{j=1, j\neq k}^{M} \phi'(u^k - u^j) + \eta_k \phi'(u^k - u^k) - \eta_k \phi'(u^k - u^k) - \sum_{l=1, l\neq k}^{M} \eta_l \phi'(u^l - u^k) \\ &= \eta_k \sum_{j=1}^{M} \phi'(u^k - u^j) - \sum_{l=1}^{M} \eta_l \phi'(u^l - u^k) \end{aligned} \tag{IV.106}$$

and (IV.105) follows. In this case, for all k, j

$$\frac{\eta_k}{\eta_j} = \frac{\sum_{l=1}^{M} \eta_l \phi'(u^l - u^k)}{\sum_{l=1}^{M} \eta_l \phi'(u^l - u^j)} \frac{\sum_{l=1}^{M} \phi'(u^j - u^l)}{\sum_{l=1}^{M} \phi'(u^k - u^l)} \tag{IV.107}$$

and, if ϕ is strictly decreasing,

$$\frac{\eta_k}{\eta_j} = \frac{\sum_{l=1}^{M} \eta_l |\phi'(u^l - u^k)|}{\sum_{l=1}^{M} \eta_l |\phi'(u^l - u^j)|} \frac{\sum_{l=1}^{M} |\phi'(u^j - u^l)|}{\sum_{l=1}^{M} |\phi'(u^k - u^l)|}. \tag{IV.108}$$

If ϕ is strictly decreasing and strictly convex then $|\phi'|$ is strictly decreasing. Hence, if $u^k > u^j$ for all $l = 1 \ldots M$

$$|\phi'(u^l - u^k)| > |\phi'(u^l - u^j)| \tag{IV.109}$$

$$|\phi'(u^j - u^l)| > |\phi'(u^k - u^l)|, \tag{IV.110}$$

and, from (IV.108), $\frac{\eta_k}{\eta_j} > 1$. On the other hand, if $u^k = u^j$ it follows from (IV.105) that $\eta_k = \eta_j$. Assume that $\eta \neq 0$ and let $k^* = \arg\max_k u^k$. Then $\frac{\eta_{k^*}}{\eta_k} \geq 1 > 0 \quad \forall k = 1 \ldots M$. Hence, all η_k are either zero or have the same sign as η_{k^*}. Since $\eta \neq 0$, it follows that $\sum_k \eta_k \neq 0$. ∎

The lemma provides a condition under which, for any $\eta \neq 0 \in Null(\mathbf{Q}_f^\phi)$, all $\eta_k \neq 0$ have the same sign and $\sum_k \eta_k \neq 0$. This implies that $\bar{\eta} = \frac{\eta}{\sum_k \eta_k}$ is a probability vector. Together, Lemmas 3 and 4 show that, if $Rank(\mathbf{Y}) = M-1$ and ϕ is strictly decreasing and strictly convex, the space of vectors η that satisfy (IV.93) is spanned by a probability vector. Under these conditions, the class posterior probabilities can be recovered from the optimal predictor f^* as follows.

Theorem 7. *Let $\mathbf{Y} \in \mathbb{R}^{d\times M}$ be a matrix whose columns are the codewords in a set $\mathcal{Y} \in \mathcal{S}(M,d)$, $R_M^\phi(f|x)$ the conditional risk of (IV.91), when $L_M[.,.]$ is a ϕ-loss, as defined in (IV.82) and ϕ twice differentiable, and η the vector of the posterior probabilities of (IV.89). If $Rank(\mathbf{Y}) = M-1$, ϕ is strictly decreasing and strictly convex, and f^* globally minimizes $R_M^\phi(f|x)$, then η can be recovered from f^* using*

$$\eta = {\mathcal{Q}_{f^*}^\phi}^{-1}\mathbf{e}_1, \tag{IV.111}$$

where $\mathbf{e}_1 = [1, 0 \ldots 0]^T \in \mathbb{R}^M$ and

$$\mathcal{Q}_f^\phi(k,l) = \begin{cases} 1 & k = 1 \\ \mathbf{Q}_f^\phi(k,l) & k > 1. \end{cases} \tag{IV.112}$$

Moreover, the codeword projections $u^{*k} = \frac{1}{2}\left\langle y^k, f^*\right\rangle$ have the property

$$\arg\max_k u^{*k} = \arg\max_k \eta_k. \tag{IV.113}$$

Proof. From Lemmas 3 and 4, if $Rank(\mathbf{Y}) = M-1$, ϕ is strictly decreasing and strictly convex, the solution of (IV.93) is identical to the solution of $\mathbf{Q}_f^\phi\eta = 0$ and, up to a normalization constant, a probability vector. It follows that It follows that the system of equations

$$\begin{cases} \mathbf{Q}_f^\phi\eta = 0 \\ \mathbf{1}^T\eta = 1, \end{cases} \tag{IV.114}$$

has a unique solution. Denoting by r_Q^i the ith row of $\mathbf{Q}_f^\phi$, it also follows from (IV.94) that $r_Q^1 = \sum_{i=2}^M r_Q^i$. Hence, the equality $r_Q^1\eta = 0$ follows from the equalities

$r_Q^i \eta = 0$, $\forall i = 2, \ldots, M$. This implies that removing the equation associated with r_Q^1 from (IV.114) has no effect on the solution. The system of equations of (IV.114) can thus be written as

$$\mathcal{Q}_f^\phi \eta = \mathbf{e}_1, \tag{IV.115}$$

with $\mathcal{Q}_f^\phi$ as defined in (IV.112). Since (IV.114) has a unique solution, $\mathcal{Q}_f^\phi$ can be inverted and $\eta = {\mathcal{Q}_{f^*}^\phi}^{-1} \mathbf{e}_1$. (IV.113) follows from statement 1. of Lemma 4. ■

IV.F.5 Discussion

Theorems 3 and 4 show that, for a M-ary classification problem, a codeword space of dimension $M-1$ is the smallest to contain a set of codewords that achieve either the capacity or maxmin distance bounds. In this case, the optimal codeword set consists of the M vertices of the regular simplex in $\mathbb{R}^{M-1}$ and can be obtained with the procedure of [12]. The associated matrix $\mathbf{Y}$ has rank $M-1$ and the codewors are the directions of largest margin for each of the M classes. Lemma 3 shows that this is also the smallest rank to support the design of a proper loss ϕ. Theorem 6 states that this choice of codewords leads to a convex risk if ϕ is strictly convex. Finally, Theorem 7 shows that, if ϕ is also strictly decreasing, then ϕ is a proper margin loss and the posterior probability vector η can be recovered from the unique minimizer f^* of the risk using (IV.111). In summary, the combination of the $M-1$ dimensional simplex codewords with a strictly decreasing and strictly convex loss ϕ are sufficient conditions for a proper margin ϕ-loss of maximum margin capacity and a convex risk.

IV.G Risk minimization

In this section we introduce two Boosting algorithms for the minimization of the empirical risk of (IV.81). Both are gradient descent procedures in the function space of weak learner linear combinations and can be seen as generalizations of GradientBoost [53]. The first is a functional coordinate descent

algorithm, which updates the components of the multi-dimensional predictor sequentially. The second is a functional gradient descent algorithm that updates all components simultaneously.

IV.G.1 Coordinate descent

The first formulation of Boosting searches for the predictor $f^*(x) = [f_1^*(x), ..f_d^*(x)]$ that solves the optimization problem

$$\begin{cases} \min_{f_1(x),...,f_d(x)} & \overline{R}_{L_M^\phi}([f_1(x), ..., f_d(x)]) \\ s.t & f_j(x) \in span(\mathcal{H}) \quad \forall j = 1 \dots d, \end{cases} \tag{IV.116}$$

where $\mathcal{H} = \{h_1(x), ...h_r(x)\}$ is a set of *scalar* weak learners $h_i(x) : \mathcal{X} \to \mathbb{R}$. These can be stumps, regression trees, or members of any other suitable model family. We denote by $f^t(x) = [f_1^t(x), ..., f_d^t(x)]$ the predictor available after t Boosting iterations. At iteration $t+1$ a single component $f_j(x)$ of $f(x)$ is updated with a step in the direction of the scalar functional g that most decreases the risk $\overline{R}_{L_M^\phi}[f_1^t, ..., f_j^t + \alpha_j^* g, ..., f_d^t]$. For this, we consider the functional derivative of $\overline{R}_{L_M^\phi}[f(x)]$ along the direction of the functional $g : \mathcal{X} \to \mathbb{R}$, at point $f(x) = f^t(x)$, with respect to the jth component $f_j(x)$ of $f(x)$ [29],

$$\delta\overline{R}_{L_M^\phi}[f^t; j, g] = \left.\frac{\partial \overline{R}_{L_M^\phi}[f^t + \epsilon g \mathbf{1}_j]}{\partial \epsilon}\right|_{\epsilon=0}, \tag{IV.117}$$

where $\mathbf{1}_j \in \mathbb{R}^d$ is a vector whose j^{th} element is one and the remaining zero, i.e. $f^t + \epsilon g \mathbf{1}_j = [f_1^t, .., f_j^t + \epsilon g, ..f_d^t]$. Using (IV.80), it is shown in Appendix IV.L.1 that

$$-\delta\overline{R}_{L_M^\phi}[f^t; j, g] \quad = \quad \sum_{i=1}^{n} g(x_i) w_i^j, \tag{IV.118}$$

with

$$w_i^j = -\frac{1}{2} \sum_{k=1, k\neq c_i}^{M} \left\langle \mathbf{1}_j, y^{c_i} - y^k \right\rangle \phi' \left(\frac{1}{2} [\left\langle y^{c_i}, f^t(x_i) \right\rangle - \left\langle y^k, f^t(x_i) \right\rangle] \right). \tag{IV.119}$$

The direction of largest descent is then

$$\begin{aligned} g_j^*(x) &= \arg\min_{g\in\mathcal{H}} \delta\overline{R}_{L_M^\phi}[f^t; j, g] \\ &= \arg\max_{g\in\mathcal{H}} \sum_{i=1}^{n} g(x_i) w_i^j, \end{aligned} \tag{IV.120}$$

and the optimal step size along this direction

$$\alpha_j^* = \arg\min_{\alpha\in\mathbb{R}} \overline{R}[f^t(x) + \alpha g_j^*(x)\mathbf{1}_j]. \tag{IV.121}$$

Note that α_j^* may not have a closed form and a line search might be required. The predictor is finally updated with

$$f^{t+1} = f^t(x) + \alpha_j^* g_j^*(x)\mathbf{1}_j = [f_1^t, ..., f_j^t + \alpha_j^* g_j^*, ..., f_d^t]. \tag{IV.122}$$

This procedure is summarized in Algorithm 8-left and denoted CD-MCBoost. It has initial condition $f^0(x) = 0 \in \mathbb{R}^d$ and updates the predictor components sequentially. Note that if L_M^ϕ is proper, then CD-MCBoost will converge to the optimal solution.

IV.G.2 Gradient descent

Alternatively, (IV.81) can be minimized by learning a linear combination of multiclass weak learners. In this case, the optimization problem is

$$\begin{cases} \min_{f(x)} & \overline{R}_{L_M^\phi}[f(x)] \\ s.t & f(x) \in span(\overline{\mathcal{H}}), \end{cases} \tag{IV.123}$$

where $\overline{\mathcal{H}} = \{\overline{h}_1(x), ..., \overline{h}_r(x)\}$ is a set of multiclass weak learners,

$$\overline{h}_i(x) : \mathcal{X} \to \mathbb{R}^d, \tag{IV.124}$$

such as decision trees. Note that, under this definition, the output of the multiclass weak classifiers (usually a class number) should be expressed as a codeword.

As before, let $f^t(x) \in \mathbb{R}^d$ be the predictor available after t Boosting iterations. At iteration $t+1$ a step is given along the direction $g(x) \in \overline{\mathcal{H}}$ of largest

decrease of the risk $\overline{R}_{L_M^\phi}[f(x)]$. For this, we consider the directional functional derivative of $\overline{R}_{L_M^\phi}[f(x)]$ along the direction of the functional $g : \mathcal{X} \to \mathbb{R}^d$, at point $f(x) = f^t(x)$

$$\delta\overline{R}_{L_M^\phi}[f^t; g] = \left.\frac{\partial \overline{R}_{L_M^\phi}[f^t + \epsilon g]}{\partial \epsilon}\right|_{\epsilon=0}. \tag{IV.125}$$

As shown in Appendix IV.L.2,

$$-\delta\overline{R}_{L_M^\phi}[f^t; g] = \sum_{i=1}^{n} \langle g(x_i), w_i \rangle, \tag{IV.126}$$

where $w_i \in \mathbb{R}^d$

$$w_i = -\frac{1}{2} \sum_{k=1, k\neq c_i}^{M} (y^{c_i} - y^k)\phi' \left(\frac{1}{2}[\langle y^{c_i}, f^t(x_i)\rangle - \langle y^k, f^t(x_i)\rangle] \right). \tag{IV.127}$$

The direction of steepest descent is the weak learner

$$\begin{aligned} g^*(x) &= \arg\min_{g\in\overline{\mathcal{H}}} \delta\overline{R}[f^t; g] \\ &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i=1}^{n} \langle g(x_i), w_i \rangle, \end{aligned} \tag{IV.128}$$

and the optimal step size along this direction

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \overline{R}_{L_M^\phi}[f^t(x) + \alpha g^*(x)]. \tag{IV.129}$$

Again, this step size may not have a closed form and a line search might be required. The predictor is finally updated according to

$$f^{t+1}(x) = f^t(x) + \alpha^* g^*(x). \tag{IV.130}$$

This procedure is summarized in Algorithm 8-right, and denoted GD-MCBoost. Similarly to CD-MCBoost, it converges to an optimal predictor whenever L_M^ϕ is proper.

IV.H Properties of MCBoost

MCBoost has a number of interesting and intuitive properties which we will discuss in this section.

Algorithm 8 CD-MCBoost and GD-MCBoost

Input: Number of classes M, dimension d, a set of codewords, $\mathcal{Y} = \{y^1, \ldots, y^M\} \in \mathbb{R}^d$ number of iterations N and data set $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$, where x_i are training examples and $c_i \in \{1 \ldots M\}$ their classes.
Initialization: set $t = 0$, and $f^t = 0 \in \mathbb{R}^d$.

CD-MCBoost

while $t < N$ **do**
  for $j = 1$ to d **do**
    Compute w_i^j with (IV.119).
    Find $g_j^*(x)$, α_j^* using (IV.120) and (IV.121).
    Set $f_j^{t+1}(x) = f_j^t(x) + \alpha_j^* g_j^*(x)$.
    Set $f_k^{t+1}(x) = f_k^t(x) \quad \forall k \neq j$.
    Set $t = t + 1$.
  end for
end while

GD-MCBoost

while $t < N$ **do**
  Compute w_i with (IV.127).
  Find $g^*(x)$, α^* by (IV.128) and (IV.129).
  Set $f^{t+1}(x) = f^t(x) + \alpha^* g^*(x)$.
  Set $t = t + 1$.
end while

Output: decision rule: $F(x) = \arg\max_{y^k} \left\langle f^N(x), y^k \right\rangle$

IV.H.1 Predictor

In MCBoost algorithms the predictor is initialized with $f(x) = 0 \quad \forall x$ i.e. all example are mapped into the origin. Minimizing (IV.81), MCBoost learns a predictor $f(x_i)$ to maximize margin. Using (IV.9), maximizing the margin of example x_i requires maximizing projection of $f(x_i)$ over y^{c_i} while minimizing its projection over other codewords. However, when optimal codewords are used, the predictor of the largest margin must be aligned with the direction of y^{c_i}, (see Theorem 1). In addition if margin of x_i is positive, increasing magnitude of $f(x)$ will the increase the margin. Therefore MCBoost learns a predictor $f(x_i)$ that 1) is as align as possible with y^{c_i} and 2) have the largest magnitude, i.e. as far as possible from the origin. Hence starting from the origin MCBoost *pushes example's predictions in the direction of their class codewords.* This mechanism results in higher margins and thus more robust and accurate classification when using decision rule of (IV.14).

IV.H.2 Weights

Similar to binary Boosting, weights of MCBoost algorithms are very intuitive. First note that weights in CD-MCBoost, (IV.119), and GD-MCBoost, (IV.127), are closely related as

$$w_i^j = \langle \mathbf{1}_j, w_i \rangle \,, \tag{IV.131}$$

where the left w represents CD-MCBoost weights and the right one is weights in GD-MCBoost. This is not surprising since these weights are derivatives of the same objective function in optimization problems of (IV.116) and (IV.123). At each iteration, GD-MCBoost updates all coordinates of the multi-dimensional predictor simultaneously and thus its weights (IV.127) contain information about all coordinates. On the other hand CD-MCBoost updates only one coordinate at a time and thus its weights (IV.119) contain information about that coordinate. Given this in the rest of this section we mainly focus on the weights for GD-MCBoost which also contain weights of CD-MCBoost.

According to (IV.127), the weight of example x_i belonging to class c_i at iteration t is

$$w_i \;=\; -\frac{1}{2} \sum_{k=1, k\neq c_i}^{M} (y^{c_i} - y^k)\lambda(u^k), \tag{IV.132}$$

where $u^k = \frac{1}{2}\left[\langle f^t(x_i), y^{c_i}\rangle - \left\langle f^t(x_i), y^k\right\rangle\right]$ and $\lambda(u^k) = -\frac{1}{2}\phi(u^k)$. This weight vector is consist of two parts 1) vectors of $y^{c_i} - y^k$ and 2) coefficients $\lambda(u^k)$. The vector portion of these weights contains the directions that the current predictor should follow and the coefficient $\lambda(u^k)$ indicates the importance of each direction. Those directions are then amplified with their importance and summed to form the final vector w_i. MCBoost selects a weak learner g^* that has the largest similarity, dot product, with this weight vector w_i (IV.128) and add it to the ensemble. Addition of g^* will push predictor f^t towards $y^{c_i} - y^k \quad k \neq c_i$ and therefore increases the projection of f^{t+1} on y^{c_i} and decreases its projection on other $y^k \quad k \neq c_i$. To illustrated this effect, assume that example x_i is mis-classified by the current

predictor and has projections

$$\left\langle y^{l_l}, f^t(x_i)\right\rangle < \left\langle y^{c_i}, f^t(x_i)\right\rangle < \left\langle y^{l_b}, f^t(x_i)\right\rangle . \tag{IV.133}$$

In this case $u^{l_b} < 0 < u^{l_l}$ and if ϕ is strictly convex (see Theorem 7), ϕ' will be an increasing function and thus

$$\lambda(u^{l_l}) < \lambda(u^{l_b}). \tag{IV.134}$$

Therefore the vector $y^{c_i} - y^{l_b}$ has larger impact in w_i than $y^{c_i} - y^{l_l}$ and thus w_i will have less projection on y^{l_b}.

The other property of (IV.132) is that magnitude of weights for example that are mis-classified are usually larger than the correctly classified example and thus have larger impact on the weak learner selection rule of (IV.128). Therefore in each iteration, MCBoost focuses more on mis-classified examples. To illustrate this effect assume that after adding sufficient number of weak learners, $\langle f(x_i), y^{c_i}\rangle$ is more than all other $\left\langle f(x_i), y^k\right\rangle \quad k \neq c_i$, and $f(x_i)$ correctly assigns example x_i to class c_i. In this case, all $u^k \quad k \neq c_i$ will be positive, therefore $\lambda(u^k)$ will be small and thus w_i will have smaller magnitude comparing to those examples that are not correctly classified.

Finally note that in the case of binary Boosting where $M = 2$, $y^1 = 1, y^2 = -1$ and $\phi(v) = e^{-v}$, the predictor of MCBoost would be one dimensional and the weights in CD-MCBoost and GD-MCBoost would be

$$w_i = -\frac{1}{2}[y^{c_i} - (-y^{c_i})][-e^{-[\frac{1}{2}[y^{c_i}-(-y^{c_i})]f(x_i)]}] = y^{c_i}e^{-y^{c_i}f(x_i)}, \tag{IV.135}$$

which is the same as the weights in AdaBoost [26] multiplied with the labels.

IV.H.3 Complexity of multiclass classifier

A major draw back in most of the multiclass Boosting algorithms is that their required number of evaluated classifiers, denoted *complexity*, increase with the number of classes. For example in the case of "one vs all" or "all pairs" complexity increase linearly or quadratically with the number of classes, respectively.

However in MCBoost the complexity of the final predictor is determined by the its dimension, d, not number of classes, M. In fact MCBoost is even able to solve any M-array classification problem by using only a two dimensional predictor, $d = 2$. However, while using smaller dimension will decrease the complexity, it also limits the margin capacity and reduces the accuracy of resulted classifier. This is a complexity-accuracy trade-off which can be tuned by selection of a proper dimension, d, according to the requirements of the problem.

IV.H.4 Weak learners

While the current multiclass Boosting method are mostly designed for specific type of weak learners, e.g regression or decision stumps, MCBoost frame work can Boost any type of weak learner. In fact in MCBoost weak learners can be any arbitrary function e.g. they are not necessary to be classifiers. More specifically, CD-Boost can be used with any set of real-valued functions $h : \mathcal{X} \to \mathbb{R}$ and GD-MCBoost can work with any set of multi-dimensional functions $\overline{h} : \mathcal{X} \to \mathbb{R}^d$.

IV.I Comparison to previous methods

According to the algorithms 8 the resulting multiclass Boosting algorithms are as simple as binary Boosting and consist of only *a few* lines of code. The main differences between these and the binary Boosting algorithm are 1) using multi-dimensional predictor and codewords and 2) definition of weights , w. In this section we will show that most of the current Boosting algorithms are special case of MCBoost and illustrate the relationship between MCBoost and SVM.

The concepts of multi-dimensional predictors and codewords have been used implicitly or explicitly in all previous multiclass Boosting methods. While methods such as SAMME [102], multiclass LogitBoost [27], AdaBoost.ECC [32] or "one vs. all" [57] explicitly use those concepts, methods such as AdaBoost-

(M1,M2) [25], AdaBoost-(MR,MH) [73] and AdaBoost-Cost [56] use these concepts implicitly. Next we will compare these methods with MCBoost framework in more details.

IV.I.1 Multiclass LogitBoost

Multiclass LogitBoost [27] is multiclass Boosting method that learns a M dimensional regression predictor $\bar{f}(x) = [\bar{f}_1(x), ..., \bar{f}_M(x)] \in \mathbb{R}^M$ by minimizing the negative log likelihood,

$$L_{Logit}[c_i, \bar{f}(x_i)] \;=\; -\log\left[\bar{P}_{C,X}(c_i|x_i)\right], \tag{IV.136}$$

where

$$\bar{P}_{C,X}(c_i|x_i) = \frac{e^{\bar{f}_{c_i}(x_i)}}{\sum_{j=1}^{M} e^{\bar{f}_j(x_i)}}, \tag{IV.137}$$

and

$$\sum_{k=1}^{M} \bar{f}_j(x) = 0. \tag{IV.138}$$

Comparing to CD-MCBoost, first note that using (IV.136), (IV.137) and monotonicity of logarithm function,

$$\begin{aligned} L_{Logit}[c_i, \bar{f}(x_i)] &= -\log\left[\frac{e^{\bar{f}_{c_i}(x_i)}}{\sum_{j=1}^{M} e^{\bar{f}_j(x_i)}}\right] = \log\left[\frac{\sum_{j=1}^{M} e^{\bar{f}_j(x_i)}}{e^{\bar{f}_{c_i}(x_i)}}\right] \\ &= \log\left[1 + \sum_{j=1, j\neq c_i}^{M} e^{\bar{f}_j(x_i) - \bar{f}_{c_i}(x_i)}\right] \\ &= \log\left[1 + \sum_{j\neq c_i} e^{-[\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle - \langle \bar{y}^{j}, \bar{f}(x_i)\rangle]}\right] \\ &\equiv \sum_{j=1, j\neq c_i}^{M} e^{-[\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle - \langle \bar{y}^{j}, \bar{f}(x_i)\rangle]}, \end{aligned} \tag{IV.139}$$

where $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$ and (IV.139) is special case of (IV.82) when $\phi(v) = e^{-2v}$. Therefore multiclass LogitBoost is a special case of CD-MCBoost that 1) uses regression weak learners and 2) the canonical basis, $\mathbf{1}_j$, are used as codewords.

Moreover the constraint of (IV.138) is not required since its violation will not have any impact on the decision rule of (IV.14). Finally note that the same analysis is true for those algorithms that follow multiclass LogitBoost strategy such as [37] and [104].

IV.I.2 AdaBoost-Cost

AdaBoost-Cost [56] is a multiclass Boosting algorithm that works with multiclass weak learners $\hat{g}(x) : \mathcal{X} \rightarrow \{1, 2, ...M\}$, to form an ensemble $\hat{f}(x) = \sum_t \alpha_t \hat{g}_t(x)$. The final classification rule is

$$\bar{F}(x) = \arg \max_{j \in \{1,2,..M\}} \hat{f}(x_i, j), \tag{IV.140}$$

where

$$\hat{f}(x_i, j) = \sum_t \alpha_t I(\hat{g}_t(x_i) = j), \tag{IV.141}$$

and $I(.)$ is the indicator function. In each round of AdaBoost-Cost, a cost for assigning example i to class j,

$$C_{i,j} = \begin{cases} e^{\hat{f}(x_i,j) - \hat{f}(x_i,c_i)} & \text{if } \ j \neq c_i \\ -\sum_{j \neq c_i} e^{\hat{f}(x_i,j) - \hat{f}(x_i,c_i)} & \text{if } \ j = c_i \end{cases} \tag{IV.142}$$

is computed and the weak learner with lowest prediction cost

$$g^* = \arg \min_{\hat{g}} \sum_i C_{i,\hat{g}(x_i)}, \tag{IV.143}$$

is added to the ensemble.

Comparing AdaBoost-Cost with GD-MCBoost first note that by defining

$$\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M \tag{IV.144}$$

$$\bar{g}(x_i) = \bar{y}^{\hat{g}(x_i)} \tag{IV.145}$$

$$\bar{f}(x) = \sum_t \alpha_t \bar{g}_t(x), \tag{IV.146}$$

the decision rule of (IV.140) is equivalent to

$$\bar{F}(x) = \arg \max_{j \in \{1,2,..M\}} \left\langle \bar{y}^j, \bar{f}(x) \right\rangle, \tag{IV.147}$$

and weak learner selection rule of (IV.143) is equivalent to

$$g^* = \arg \min_{\bar{g}} \sum_i \left\langle C_{i,:}, \bar{g}(x_i) \right\rangle, \tag{IV.148}$$

where $C_{i,:}$ is i^{th} row of the cost matrix C defined in (IV.142). Using (IV.142) and (IV.144)

$$\begin{aligned} C_{i,:} &= \sum_{j \neq c_i} \bar{y}^j e^{\left\langle \bar{y}^j, \bar{f}(x) \right\rangle - \left\langle \bar{y}^{c_i}, \bar{f}(x) \right\rangle} - \bar{y}^{c_i} \sum_{j \neq c_i} e^{\left\langle \bar{y}^j, \bar{f}(x) \right\rangle - \left\langle \bar{y}^{c_i}, \bar{f}(x) \right\rangle} \\ &= \sum_{j \neq c_i} (\bar{y}^j - \bar{y}^{c_i}) e^{-[\left\langle \bar{y}^{c_i}, \bar{f}(x) \right\rangle - \left\langle \bar{y}^j, \bar{f}(x) \right\rangle]}, \end{aligned} \tag{IV.149}$$

which is equivalent to weight vector for GD-MCBoost in (IV.127) when $\phi(v) = e^{-2v}$. Therefore AdaBoost-Cost is a special form of GD-MCBoost that implicitly uses the canonical basis, $\mathbf{1}_j$, as codewords.

IV.I.3 SAMME

SAMME [102] is a multiclass Boosting algorithm that learns a M-dimensional predictors $\bar{f} = [\bar{f}_1, ...\bar{f}_M] \in \mathbb{R}^M$ using codewords

$$\bar{y}^j = \frac{M\mathbf{1}_j - \mathbf{1}}{M-1} = \left[\frac{-1}{M-1}, \frac{-1}{M-1}, ..., 1, \frac{-1}{M-1}, \frac{-1}{M-1} \right] \in \mathbb{R}^M, \tag{IV.150}$$

and decision rule

$$\begin{aligned} \bar{F}(x) &= \arg \max_{j \in \{1,2,..M\}} \bar{f}_j(x) \\ &\equiv \arg \max_{j \in \{1,2,..M\}} \left\langle \bar{y}^j, \bar{f}(x) \right\rangle. \end{aligned} \tag{IV.151}$$

The loss function of this method is

$$L_{SAMME}[\bar{y}^{c_i}, f(x_i)] = e^{-\frac{1}{M} \left\langle \bar{y}^{c_i}, \bar{f}(x_i) \right\rangle}. \tag{IV.152}$$

However, note that minimization of this $L_{SAMME}[.,.]$, is equivalent to maximizing the term in the exponent

$$\begin{aligned}\mathcal{M}_{SAMME}(\bar{y}^{c_i}, f(x)) &= \left\langle \bar{y}^{c_i}, \bar{f}(x) \right\rangle \\ &= \bar{f}_{c_i}(x) - \frac{1}{M-1}\sum_{j\neq k} \bar{f}_j(x). \end{aligned} \tag{IV.153}$$

which is not equivalent to maximizing the multiclass margin of (IV.9). In fact $\mathcal{M}_{SAMME}$ is not a definition of multiclass margin since $\mathcal{M}_{SAMME}(y^{c_i}, \bar{f}(x)) > 0$ does not imply correct classification by (IV.151), i.e. $\bar{f}_{c_i}(x) > \bar{f}_j(x) \quad \forall j \neq k$. Therefore while SAMME is very similar to GD-MCBoost, minimizing $L_{SAMME}(.,.)$ 1) does not minimize the error rate and 2) does not guarantee a large margin predictor.

IV.I.4 AdaBoost.ECC

AdaBoost.ECC [32] is a multiclass Boosting algorithm that uses ECOC strategy [16] and the multiclass loss function of (IV.82) when $\phi(v) = e^{-2v}$. AdaBoost.ECC assigns a binary codewords, $\bar{y}^j$ to each class and trains a predictor for each bit of those codes. AdaBoost.ECC starts with an empty binary codeword for each class and in each iteration t, 1) a new bit is augmented to the codewords and 2) a new predictor dimension $\bar{f}_t$ is added to learn the new bit. After T iterations, the output of the algorithm will be a set of binary codewords $\bar{y}^j \in \{0,1\}^T$ and a T dimensional predictor $\bar{f}(x) = [\bar{f}_1(x), ...\bar{f}_T(x)] \in \mathbb{R}^T$. At the test time the predictor, $\bar{f}(x)$, is evaluated and the class with lowest Hamming distance

$$\begin{aligned}\bar{F}(x) &= \arg\min_{j\in\{1,2,..M\}} Hamming[\bar{f}(x_i), \bar{y}^j], \\ &\equiv \arg\max_{j\in\{1,2,..M\}} \left\langle \bar{f}(x), \bar{y}^j \right\rangle, \end{aligned} \tag{IV.154}$$

is selected.

Comparing AdaBoost.ECC with CD-MCBoost the only difference is in definition of the codewords. While in CD-MCBoost these codewords are real-values and selected prior to learning by maximizing the margin capacity of (IV.28), in

AdaBoost.ECC codewords are binary and are generated on-the-fly in each iteration either by random selection or by solving a "max-cut" problem. However note that 1) AdaBoost.ECC will result in a sub-optimal set of codewords that limits the margin maximizing ability of the algorithm, 2) increasing the length of codewords in each iteration increases the dimension of the predictor and thus increases the chance of over-fitting and complexity at the test time. The same is true for all other methods that follow ECOC strategy [16] such as [72, 43, 100, 30].

IV.I.5 AdaBoost-MR

AdaBoost-MR [73] uses weak learners of the form

$$h_t : \mathcal{X} \times \{1, \ldots, M\} \to \mathbb{R}, \tag{IV.155}$$

to create a predictor $\bar{f}(x) = [\bar{f}_1(x), ...\bar{f}_M(x)]$ where

$$\bar{f}_j(x) = \sum_t h_t(x, j), \tag{IV.156}$$

and j, t are class and iteration numbers respectively. The decision rule in this algorithm is

$$\begin{aligned} \hat{F}(x) &= \arg \max_{j \in \{1,2,..M\}} \bar{f}_j(x) \\ &\equiv \arg \max_{j \in \{1,2,..M\}} \left\langle \bar{y}^j, \bar{f}(x) \right\rangle, \end{aligned} \tag{IV.157}$$

where $\bar{y}^j = \mathbf{1}_{j.} \in \mathbb{R}^M$. AdaBoost.MR learns $\bar{f}(x)$ by minimizing

$$L_{MR}[c_i, \bar{f}(x_i)] = \sum_{j=1, j \neq c_i}^{M} e^{-[f_j(x_i) - f_{c_i}(x_i)]}, \tag{IV.158}$$

which is special case of (IV.82) when $\phi(v) = e^{-2v}$.

The main difference between AdaBoost.MR and GD-MCBoost is the type of used weak learners for Boosting i.e. (IV.155) vs. (IV.124). However note that in order to use weak learners of the form of (IV.155), training examples are usually augmented with the class numbers to form a new training examples

$\bar{x} = [x, j]\ \forall j$ and weak learners $h_t : \bar{\mathcal{X}} \rightarrow \mathbb{R}$ are used for building the ensemble. This trick is problematic since 1) it increase the size of training set and 2) it is hard to discriminate examples that differ only by one coordinate using simple weak learners such as decision stumps. These effects makes AdaBoost-MR algorithm to 1) become computationally intensive, 2) converge slowly and 3) require strong weak learners. The same problem arises for AdaBoost-MH [73] which also uses weak learners in the form of (IV.155).

IV.I.6 MCBoost and Kernelized SVM

Using the framework of MCBoost, it is possible to unify Boosting and support vector machines as margin maximizer learning algorithms. Given a training set $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$ kernelized support vector machines, K-SVM, first transforms training examples x_i to a new space using a mapping $\mathbf{\Phi}(x)$ introduced by the kernel $\mathbf{K}(.,.)$, i.e. $\mathbf{\Phi}(x) = \mathbf{K}(x,.)$. A linear classifier is then trained to separate examples of different classes in the new domain by maximizing the margin [88]. Similar to Boosting, there are several extensions for multiclass K-SVM such as 1) reducing multiclass to several binary sub-problems as in 'one vs all', 'all pairs' and 'ECOC' [75, 7, 4] or 2) building a direct formulation for multiclass problem [88, 95, 35, 13].

In the direct formulation, multiclass K-SVM solve a M-ary classification problem by learning M linear classifiers $\mathbf{w}_l,\ l = 1..M$, that maximize the multiclass margin of

$$\mathcal{M}_{K-SVM}(x_i, \mathbf{w}_{c_i)}) \quad = \quad \frac{1}{2}[\langle \Phi(x_i), \mathbf{w}_{c_i} \rangle - \max_{l \neq c_i} \langle \Phi(x_i), \mathbf{w}_l \rangle]. \qquad \text{(IV.159)}$$

Maximization of (IV.159) can be formulated in several ways [88, 95, 35, 13]. In this dissertation we resort to the original formulation of Vapnik [88] which finds the optimal linear classifiers, $\mathbf{w}^*$ by solving [1]

$$\begin{cases} \min_{\mathbf{w}_l..\mathbf{w}_M} & \sum_{k=1}^M \|\mathbf{w}_l\|_2^2 \\ s.t & \langle \Phi(x_i), \mathbf{w}_{c_i} \rangle - \langle \Phi(x_i), \mathbf{w}_l \rangle \geq 1 \quad \forall (x_i, c_i) \in \mathcal{D}, l \neq c_i, \end{cases} \qquad \text{(IV.160)}$$

[1] For simplicity we have omitted the slack variables and the bias terms.

and predicts

$$F_{SVM}(x) = \arg\max_{l=1..M} \langle \Phi(x), \mathbf{w}_l^* \rangle \,. \tag{IV.161}$$

Comparing (IV.159) with (IV.9) and (IV.161) with (IV.14), shows that

- The transformation $\Phi(x)$ in K-SVM is corresponding to the predictor, $f(x)$, in Boosting.
- The linear classifiers $\mathbf{w}_l$ in K-SVM are corresponding to class codewords y^l in Boosting.

These correspondences indicate that Boosting and K-SVM optimize similar concepts for maximizing the margin, but their optimization procedures are different. In K-SVM examples are first transformed into a new domain using a *pre-defined* transformation (predictor), then K-SVM finds a set of linear classifiers that maximizes the margin of examples in the transformed domain. On the other hand in Boosting the linear classifiers (codewords) are first defined to be fixed vectors and then Boosting finds a predictor (transformation) that maximizes the margin with respect to the codewords (pre-defined linear classifiers).

Comparing K-SVM and Boosting also reveals that unlike K-SVM for which the optimal kernel, and induced transformation, should be found by trail-and-error, Boosting builds this optimal transformation using the provided set of weak learners. In addition, in Boosting it is possible to design codewords with the maximum margin capacity (IV.28), (see Section IV.E). Therefore while SVM has only one tool for maximizing margin, i.e. finding the optimal linear classifiers, Boosting can maximize the margin by two means 1) *selection of codewords* (linear classifiers) with maximum margin capacity and 2) *learning an optimal transformation* (predictor) to maximize the margin.

IV.J Evaluation

A number of experiments were conducted to evaluate the performance of MCBoost algorithms. By default MCBoost algorithms were implemented using

Table IV.1 Characteristics of the used UCI data sets

#Id	Data Name	#Training	#Testing	#Attributes	# Classes
1	*Landsat Satellite*	$4,435$	$2,000$	36	6
2	*Letter*	$16,000$	$4,000$	16	26
3	*Pen Digit*	$7,494$	$3,498$	16	10
4	*Poker*	$25,010$	$1,000$	10	10
5	*Optical Digit*	$3,823$	$1,797$	64	10
6	*Shuttle*	$43,500$	$14,500$	9	7
7	*Isolet*	$6,238$	$1,559$	617	26
8	*Vehicle*	692	154	18	4
9	*Vertebral*	239	71	6	3
10	*Image Segmentation*	210	$2,100$	19	7
11	*Ecoli*	258	78	7	8
12	*Breast Tissue*	81	25	9	6

$d = M - 1$, vertices of a regular simplex in $\mathbb{R}^{M-1}$ as codewords and (IV.82) with $\phi(v) = e^{-v}$ as the loss function. An implementation of CD-MCBoost and GD-MCBoost is available from [2].

The first experiment is based on a syntethic data set, for which the optimal decision rule is known. This is a three class problem, with two-dimensional Gaussian classes of means

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \tag{IV.162}$$

and covariances

$$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}, \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.8 \end{bmatrix}, \tag{IV.163}$$

respectively. Training and test sets were each consist of $1,000$ random examples and the Bayes rule is computed in closed form [21]. The associated Bayes error rate was 11.67% in the training and 11.13% in the test set. Other experiments are based on the twelve UCI data sets of Table IV.1. The training/test set decomposition for these data sets were either provided by the data set or we randomly selected 20% of examples for testing.

IV.J.1 Synthetic data

We started with the synthetic example and trained a classifier with CD-MCBoost and decision stumps. Figure IV.3 and Figure IV.4 show predictions[2] of $f^t(x)$ and histogram of example margins $\mathcal{M}(y^{c_i}, f^t(x_i))$ for the test set after $t = 0, 10, 100$ iterations, respectively.

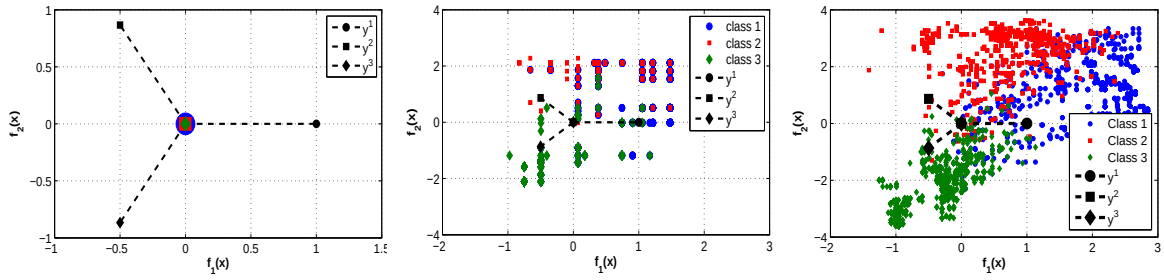
First note that in the begining the predictor maps all examples to the origin i.e. $\forall x_i \; f^0(x_i) = [0, 0]^T$. However, as algorithm proceed, CD-MCBoost produces predictions that are more aligned with the true class codewords, i.e. directions that are shown as dashed lines in Figure IV.3. This alignment of $f^t(x_i)$ with y^{c_i} implies that $f^t(x_i)$ will have more projections on y^{c_i} than other codewords and thus predictor f^t will correctly classify x_i using the decision rule of (IV.14).

Similarly since $\forall x_i \; f^0(x_i) = [0, 0]^T$, in begining margin of all examples are zero, i.e. $\forall x_i \; \mathcal{M}(y^{c_i}, f^0(x_i)) = 0$ and their histogram will be as of Figure IV.4 - a. However according to (IV.84) minimizing the loss function will increase margin of the examples, therefore as algorithm proceed, it learns a predictor under which examples have larger margins. This margin maximization is achieved by 1) making negative margins positive, which according to Corollary 1 results in more accurate classification, and 2) increases margins that are already positive, which improves the confidence of the classification. Figure IV.4- b and Figure IV.4-c illustrate these effects. Finally, error rate on the test set after 100 iterations was 11.30%, and very close to the Bayes error rate of 11.13%.

IV.J.2 Effect of codeword dimension, d

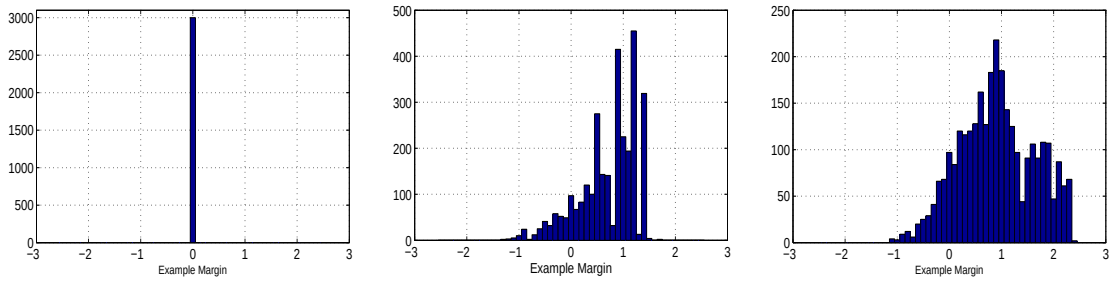
According to Section IV.E, the optimal number of dimensions for M-array classification problem is $d = M - 1$. In this section we illustrate effect of this parameter on the performance of the MCBoost algorithms and build classifier with 1) less than optimal dimensions, i.e. $d = 1 \ldots M - 2$, 2) optimal dimension, $d = M - 1$ and 3) more than optimal dimension, e.g. $d = M$. For each value of d we

[2] We emphasize that these are plots of $f^t(x) \in \mathbb{R}^2$, not $x \in \mathbb{R}^2$.



$a)t = 0$ $b)t = 10$ $c)t = 100$

Figure IV.3: Classifier predictions of CD-MCBoost, on the test set, after $t = 0, 10, 100$ Boosting iterations.



$a)t = 0$ $b)t = 10$ $c) = 100$

Figure IV.4: Histogram of example margins, $\mathcal{M}(y^{c_i}, f^t(x_i))$ using CD-MCBoost prediction, on the test set, after $t = 0, 10, 100$ Boosting iterations.

used codeword set of maximum distance note that in the case of $d \geq M - 1$ these codewords are the same as the optimal maximum margin codewords. Classifiers in this experiment were trained with 200 iterations. Moreover, decision stumps and trees of height 2 were used as weak learners for CD-MCBoost and GD-MCBoost respectively. Figure IV.5 shows accuracy of final classifiers as a function of number of dimensions d for the five data sets listed in the legends.

First, according to Figure IV.5 increasing d will increase the accuracy of the classifiers both in CD-MCBoost and GD-MCBoost. This is not surprising since larger d results in larger space for selecting codewords, i.e. $\mathbb{R}^d$, and make it possible to increase the minimum mutual distance between codewords d_{min}. According to Lemma 2, larger d_{min} yields larger margin capacity and thus MCBoost algorithms can produce predictors with larger margin that are more accurate. Second, note

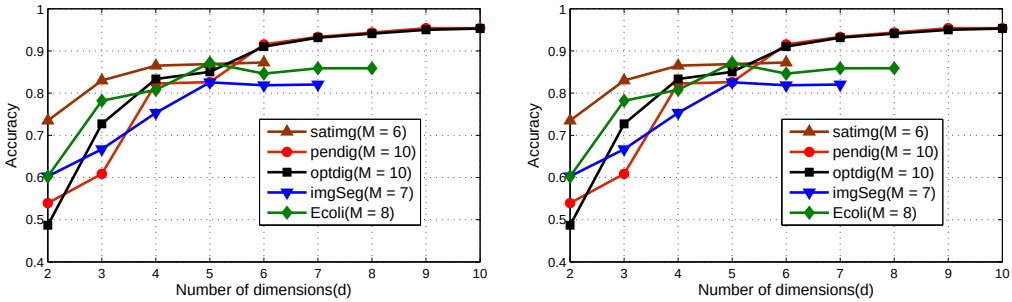


Figure IV.5: Effect of the dimension of codewords on the accuracy of the trained classifiers.

that increasing d will increase complexity of classifier since a d-dimensional predictor has to be evaluated at classification time. Therefore curves of Figure IV.5 also show the trade-off between accuracy and complexity of MCBoost classifier. Finally note that comparing case of $d = M$ and $d = M - 1$ in Figure IV.5, increasing d beyond $M - 1$ only increases complexity and has no significant effect on the accuracy. This is again not surprising since increasing d beyond $M - 1$ will not increase margin capacity.

IV.J.3 Comparison with other multiclass Boosting method

In Section IV.I we compared MCBoost algorithms with existing multiclass Boosting algorithms theoretically. In this section we compare their performance with CD-MCBoost and GD-MCBoost empirically.

CD-MCBoost

Among the methods identified as comparable in the previous section, we implemented "one vs all" (AdaBoost-OVA), AdaBoost-ECC [32] and multiclass LogitBoost [27].

Table IV.2 shows the performance of AdaBoost-OVS, AdaBoost-ECC and CD-MCBoost when decision stumps on example attributes were used as week learners. Comparing with AdaBoost-OVA, CD-MCBoost classifiers had better accuracy in six, same accuracy in four cases out of twelve. Comparison with

AdaBoost.ECC, CD-MCBoost has better performance in six and same performance in three cases out of twelve. Comparing all methods, CD-MCBoost produced the most accurate classifier in six out of twelve cases, the next best method was AdaBoost-ECC and AdaBoost-OVA resulted in the worst performance.

Table IV.3 compares the CD-MCBoost with multiclass LogitBoost. In order to have a fair comparison we implemented CD-MCBoost with regression weak learners and the loss function of (IV.139). Comparing with Multi.Logit, CD-MCBoost had better performance in four case, Multi.Logit was better in three cases and in five cases both algorithms resulted in the same accuracy. This close performance is not surprising since as discussed in Section IV.I.1 the only major difference between Multi.Logit and this implementation of CD-MCBoost is in codewords i.e. canonical basis in $\mathbb{R}^M$ for Multi.Logit vs. vertices of a simplex in $\mathbb{R}^{M-1}$ for the CD-MCBoost. This results again confirm results of Figure IV.5 that increasing d beyond $M-1$ will not improve the performance.

GD-MCBoost

Finally, the performance of GD-MCBoost was compared to AdaBoost-M1 [25], AdaBoost-Cost [56] and AdaBoost-SAMME [102]. The experiments were based on the UCI data sets of the previous section, but the weak learners were trees of depth 2 to show that unlike AdaBoost.M1 that requires strong weak learners for Boosting, GD-MCBoost is able to Boost very simple weak learners. These tree weak learners were built with a greedy procedure so as to 1) minimize the weighted error rate of AdaBoost-M1 [25] and AdaBoost-SAMME[102], 2) minimize the classification cost of AdaBoost-Cost [56], or 3) maximize (IV.128) for GD-MCBoost. Table IV.4 presents the classification accuracy of each method.

First, note that AdaBoost.M1 was not able to Boost the weak learners used in this experiment in half of the cases. Comparing GD-MCBoost with SAMME, in eleven cases of twelve cases GDMCBoost has better performance. The improvements were significant in some cases e.g. from 53% to 85% in #2 or from

85% to 95% in #7. The inferior performance of SAMME is likely due to its loss function definition which is not margin maximizer, see Section IV.I.3. Comparing to Ada.Cost, GD-MCBoost has better performance in eleven cases and same performance in one case. Again improvements over Ada.Cost can be significant e.g. from 64% to 85% in #2. This superior performance is not surprising since we showed in Section IV.I.2 that Ada.Cost is a sub-optimal special case of GD-MCBoost. Finally, when compared to all methods, GD-MCBoost achieved the best accuracy on ten out of twelve data sets. Among the remaining methods, AdaBoost-Cost has better performance followed by AdaBoost-SAMME. AdaBoost-M1 had the worst results. It should be noted that the results of Table IV.2, Table IV.3 and Table IV.4 are not directly comparable, since the classifiers are based on different types of weak learners and have different complexities.

IV.K Acknowledgments

The text of Chapter IV, in part, is based on the material as it is appeared in: Mohammad Saberian and Nuno Vasconcelos. Multiclass Boosting: Theory and Algorithms. *Proceedings of Neural Information Processing Systems (NIPS)*, 2011. A more comprehensive version of the material presented in this section is also going to be submitted to Journal of Machine Learning Research (JMLR). The dissertation author was a primary researcher and an author of the cited material.

Table IV.2 Comparison of accuracy (%) of CD-MCBoost with corresponding multiclass Boosting algorithms.

Data set id	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
AdaBoost-OVA	88.0	83.3	95.0	50.9	**95.0**	79.2	**95.3**	68.2	83.1	67.8	85.9	32.0
AdaBoost-ECC	**88.2**	77.8	94.9	**51.6**	94.8	79.2	93.6	69.5	83.1	**82.7**	84.6	**44.0**
CD-MCBoost	87.0	**84.0**	**95.4**	51.5	**95.0**	79.2	95.0	**71.4**	83.1	81.4	**85.9**	**44.0**

Table IV.3 Comparison of accuracy (%) of CD-MCBoost with multiclass LogitBoost.

Data set id	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
Multi.Logit	84.4	77.5	92.9	50.8	89.6	94.7	**96.3**	81.2	87.3	**93.0**	85.9	**76.0**
CD-MCBoost	**84.5**	77.5	92.9	50.8	**95.1**	**96.7**	95.6	81.2	87.3	92.3	**87.2**	72.0

Table IV.4 Comparison of accuracy (%) of GD-MCBoost with corresponding multiclass Boosting algorithms.

Data set id	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
Ada.M1	72.9	–	–	–	–	96.4	–	59.1	**88.7**	–	73.1	60.0
SAMME	82.0	53.3	90.4	52.0	91.8	99.7	85.7	76.6	84.5	93.1	87.2	**80.0**
Ada.Cost	89.0	64.4	95.2	59.2	95.4	99.8	91.2	81.2	87.3	94.9	84.6	68.0
GD-MCBoost	**89.1**	**85.2**	**96.5**	**60.8**	**96.8**	**99.9**	**95.2**	**81.8**	87.3	**95.3**	**89.7**	76.0

IV.L Appendix

IV.L.1 Derivation of CD-MCBoost

From (IV.80) and (IV.117)

$$
\begin{aligned}
-\delta\overline{R}[f^t; j, g] &= -\frac{\partial}{\partial\epsilon}\sum_{i=1}^{n} L_M^{\phi}[y^{c_i}, f^t(x_i) + \epsilon g(x_i)\mathbf{1}_j]\Big|_{\epsilon=0} \\
&= -\sum_{i=1}^{n} \frac{\partial L_M^{\phi}[y^{c_i}, f^t(x_i) + \epsilon g(x_i)\mathbf{1}_j]}{\partial\epsilon}\Bigg|_{\epsilon=0} \\
&= -\sum_{i=1}^{n} \frac{\partial}{\partial\epsilon}\sum_{k=1,k\neq c_i}^{M} \phi\left[\frac{1}{2}\left\langle f^t(x_i) + \epsilon g(x_i)\mathbf{1}_j, y^{c_i} - y^k\right\rangle\right]\Bigg|_{\epsilon=0} \\
&= -\sum_{i=1}^{n}\sum_{k\neq c_i} \frac{\partial}{\partial\epsilon}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k\right\rangle + \frac{1}{2}\epsilon g(x_i)\left\langle \mathbf{1}_j, y^{c_i} - y^k\right\rangle\right]\Bigg|_{\epsilon=0} \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{k\neq c_i} g(x_i)\left\langle \mathbf{1}_j, y^{c_i} - y^k\right\rangle \phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k\right\rangle\right] \\
&= -\frac{1}{2}\sum_{i=1}^{n} g(x_i)\sum_{k\neq c_i}\left\langle \mathbf{1}_j, y^{c_i} - y^k\right\rangle \phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k\right\rangle\right] \\
&= \sum_{i=1}^{n} g(x_i) w_i^j, \qquad \text{(IV.164)}
\end{aligned}
$$

where

$$
w_i^j = -\frac{1}{2}\sum_{k=1,k\neq c_i}^{M}\left\langle \mathbf{1}_j, y^{c_i} - y^k\right\rangle \phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k\right\rangle\right]. \qquad \text{(IV.165)}
$$

IV.L.2 Derivation of GD-MCBoost

Using (IV.80) and (IV.125)

$$
\begin{aligned}
-\delta\overline{R}[f^t;g] &= -\frac{\partial}{\partial\epsilon}\sum_{i=1}^{n} L_M^{\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)]\Big|_{\epsilon=0} \\
&= -\sum_{i=1}^{n}\frac{\partial L_M^{\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)]}{\partial\epsilon}\Bigg|_{\epsilon=0} \\
&= -\sum_{i=1}^{n}\frac{\partial}{\partial\epsilon}\sum_{k=1,k\neq c_i}^{M}\phi\left[\frac{1}{2}\left\langle f^t(x_i)+\epsilon g(x_i), y^{c_i}-y^k\right\rangle\right]\Bigg|_{\epsilon=0} \\
&= -\sum_{i=1}^{n}\sum_{k=1,k\neq c_i}^{M}\frac{\partial}{\partial\epsilon}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle+\frac{\epsilon}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\right]\Bigg|_{\epsilon=0} \\
&= -\sum_{i=1}^{n}\sum_{k\neq c_i}\frac{1}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right] \\
&= -\frac{1}{2}\sum_{i=1}^{n}\left\langle g(x_i), \frac{1}{2}\sum_{k\neq c_i}(y^{c_i}-y^k)\phi'\right\rangle\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right] \\
&= \sum_{i=1}^{n}\left\langle g(x_i), w_i\right\rangle, \qquad \text{(IV.166)}
\end{aligned}
$$

where

$$
w_i = -\frac{1}{2}\sum_{k=1,k\neq c_i}^{M}(y^{c_i}-y^k)\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]. \qquad \text{(IV.167)}
$$

Chapter V

Multi-Resolution Detector Cascade

V.A Introduction

In Chapter III we proposed Boosting algorithms for designing binary detector cascades, and in Chapter IV we proposed a framework for designing multiclass Boosting algorithm. In this chapter, we combine those two algorithms and design a multiclass detector cascade for detecting multiple objects. As discussed in Chapter III the design of cascades for real-time detector cascade, Figure V.1-a, for a single object class has been the subject of extensive research [92, 103, 60, 81, 8, 64, 69]. However, the simultaneous detection of multiple objects has received much less attention. In fact, most existing solutions simply decompose this problem into several binary (single class) detection sub-problems. They can be grouped as follows.

Parallel cascades [96]: These methods learn a cascaded detector per object class (e.g. view), as shown in Figure V.1-b, and rely on some post-processing to combine their decisions. This has two limitations. The first is the well known sub-optimality of one-vs.-all multiclass classification, since scores of independently trained detectors are not necessarily comparable [54]. Second, because there is no sharing of features across detectors, the overall classifier performs redundant computations and tends to be very slow. This has motivated some work in the feature sharing problem. Examples include JointBoost [86], which implements an exhaustive search for features to be shared between classes, and [61], which implicitly partitions examples of the positive class and performs a joint search for the best partition and features. However, these methods have large training complexity and questionable scalability.

Parallel cascades with pre-estimator [90]: The complexity of the parallel architecture can be reduced by first making a rough guess of the target class and running only one of the binary detectors, as illustrated in Figure V.1-c. While, for some applications (e.g. where classes are object poses), it is possible to obtain a reasonable pre-estimate of the target class, pre-estimation errors are

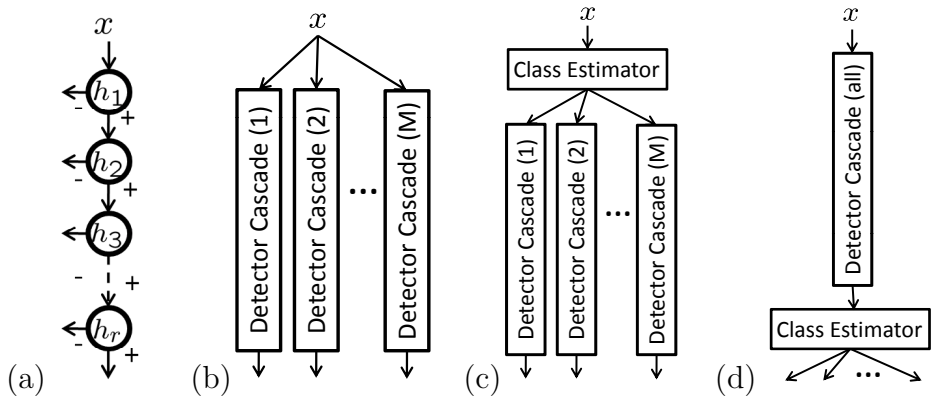


Figure V.1: a) detector cascade [91], b) parallel cascade [96], c) parallel cascade with pre-estimator [90] and d) all-class cascade with post-estimator.

difficult to undo. Hence, this classifier must be fairly accurate. Since it must also be fast, this approach boils down to real-time multiclass classification, i.e. the original problem. [36] proposed a variant of this method, where multiple detectors are run after the pre-estimate. This improves accuracy but increases complexity.

In this dissertation, we pursue an alternative strategy, inspired by Figure V.1-d. Under this architecture target classes are first grouped into an abstract class of *positive patches*. A detector cascade is then trained to distinguish these patches from everything else. A patch identified as positive is finally fed to a multiclass classifier, for assignment to one of the target classes. In comparison to parallel cascades, this method has the advantage of sharing features across all classes, eliminating redundant computation. When compared to the parallel cascade with pre-estimator, it has the advantage that the complexity of its class estimator has little weight in the overall computation, since it only processes a small percentage of the examples. This allows the use of very accurate/complex estimators. The main limitation is that the design of a cascade to detects all positive patches can be quite difficult, due to the large intra-class variability, e.g. Viola and Jones declared this strategy hopelessly inaccurate [90].

We argue, however, that this is due to the abrupt transition between the all-class and multiclass regimes. While it is difficult to build an all-class detector

with high detection and low false-positive rate, this is not true if the false-positive constraint is relaxed. This is, in fact, the essence of cascade design where it is always the case that 1) all stages must have high detection rate (rejected examples cannot be recovered), but 2) low false-positive rates are only required for the latest cascade stages. The second property is a necessary condition for the early stages to be simple, so that there can be computational savings. Hence, a defining property of any cascade, including binary cascades, is that false positive rates should be gradually decreasing for deeper cascade stages. This suggests that, rather than the abrupt all-class to multiclass transition of Figure V.1-d, a multiclass cascade should gradually progress from all-class to multiclass. Early stages would be all-class detectors of high simplicity and false-positive rate, late stages would be multiclass classifiers of high accuracy/complexity. In between, there would be stages of classifiers with intermediate numbers of classes, determined by the structure of the data itself, so as to guarantee decreasing false positive rates with cascade depth. Since cascades with this behavior represent the set of classes with different resolutions, we refer to them as multi-resolution (MRes) cascades.

The open question is then how to learn MRes cascades. We show that this is possible with resort to cost-sensitive learning. We consider a M-class classification problem and define a negative class $M+1$, which contains all non-target examples. We then rely on a cost-sensitive $M+1$ class Boosting algorithm to learn the cascade stages. By manipulating the costs of the associated risk function, it is possible to emphasize either 1) discrimination between the target classes and the negative class, or 2) discrimination among the target classes. This favors the learning of either 1) all-class or 2) multiclass cascade stages. By tying the costs of the risk function to the false positive rates of the cascade stages it is then possible to guarantee the MRes behavior. We use this strategy to design a Boosting algorithm for learning MRes cascades and show, through experiments in multi-view car detection and simultaneous detection of multiple traffic signs, that the resulting classifiers are faster and more accurate than those previously available.

V.B Multi-resolution cascades

In this section, we start by introducing the main ideas of the proposed solution. The detailed derivation of learning algorithms is left for the subsequent sections.

V.B.1 Multiclass detector cascade

The proposed multiclass detector cascade has the structure of Figure V.1-a. The fundamental difference is that, instead of a binary detector, each stage implements a multiclass classifier. The set of M target classes $\{1, \ldots M\}$, is first augmented with a class $M+1$, containing non-target examples. This is denoted the *negative class*. A multiclass detector cascade $\mathcal{H}$ is then defined as in Figure V.1-a. It assigns input x to class $z \in \{1, \ldots, M+1\}$ according to

$$\mathcal{H}(x) = \begin{cases} h_r(x) & \text{if } \; h_k(x) \neq M+1 \; \forall k \\ M+1 & \text{if } \exists k | \; h_k(x) = M+1, \end{cases} \tag{V.1}$$

where r is the number of detector stages and $h_1, \ldots h_r : \mathcal{X} \rightarrow \{1, \ldots, M+1\}$ the stage classifiers.

V.B.2 Cost-sensitive learning

A cost-sensitive $M+1$-class learning algorithm searches for the classifier $h(x)$ that minimizes a classification risk or its empirical estimate

$$\mathcal{R}[h] = E_{X,Z} \left\{ \sum_{k=1}^{M+1} C_{z,k} \mathbb{I}(h(x) = k), \right\} \tag{V.2}$$

$$\overline{\mathcal{R}}[h] \approx \frac{1}{n} \sum_{i=1}^{n} \left\{ \sum_{k=1}^{M+1} C_{z_i,k} \mathbb{I}(h(x_i) = k), \right\}, \tag{V.3}$$

where $\mathbb{I}(.)$ is the indicator function, C a cost matrix, and $\mathcal{D} = \{(x_i, z_i)\}_1^n$ a set of examples x_i and associated class labels z_i. The matrix C encodes the cost of different classification errors, assigning cost $C_{j,k}$ to the classification of an example from class j into class k. We also assume $C_{k,k} = 0 \; \forall k$ although we sometimes omit for simplicity.

V.B.3 Class transitions

The main question that we explore in this work is how to achieve class transitions. More precisely, how to design a sequence of cost matrices C_s so as to guarantee that, as examples progress through the cascade stages, s, the classification evolves from binary to M-ary.

We start by noting that, for the cascade to work at all, the cost matrix must satisfy the constraint

$$C_{k,M+1} \gg C_{M+1,k} \; \forall k = 1, \ldots, M. \tag{V.4}$$

This encodes the fact that the cost of assigning an example from class k to the negative class $M+1$ is much larger than the converse. It guarantees that the classifier $h(x)$ emphasizes the detection of target examples. This is a critical requirement to ensure a high detection rate and should be satisfied by *all cascade stages*.

Beyond this constraint, the cost-matrix can also control the relative costs of assigning targets to their classes. In the *early stages*, where the cascade should focus more on the rejection of negatives than on the correct classification of targets, the costs should satisfy the constraint

$$C_{k,l} \ll C_{k,M+1} \; \forall k, l = 1, \ldots, M. \tag{V.5}$$

This makes the cost of assigning an example from class k to another target class $l \neq M+1$ much smaller than that of assigning it to the negative class $M+1$. In result, the learning algorithm has small incentive to disambiguate between examples from classes k, l and the classifier $h(x)$ is an all-class detector, as in Figure V.1-d.

In the *late stages*, the cascade should focus on the precise assignment of targets to their individual classes. This can be encoded as

$$C_{k,l} = 1 \quad \forall k, l = 1, \ldots, M+1. \tag{V.6}$$

In this case, the accuracy of fine grained target classification becomes important and the cascade behaves as a multiclass classifier. This is similar to the class post-estimator of Figure V.1-d.

V.B.4 The cost schedule

To enforce the multi-resolution behavior, we need a *cost schedule,* i.e. a sequence of cost matrices $C^{(s)}$ that satisfies the constraints of (V.4) and (V.5) when s is small (early stages) and the constraints of (V.4) and (V.6) when s is large (late stages). To design such a schedule, we exploit a necessary property of any cascade, which follows from the following universal cascade requirements. First, since an example rejected by a stage cannot be recovered latter on, all stages of a cascade must have a high detection rate. Second, since the goal is to minimize the average computation, early stages (which are evaluated for all examples) must have low complexity, while late stages (evaluated only for "difficult" examples) can be complex. Third, since the goal is to produce a good classifier, late stages must have low false positive rate. These requirements are not independent, since to guarantee a cascade of constant detection rate, low early complexity, and low late false positive rate, it is necessary that the false positive rate (complexity) be gradually decreasing (increasing) for deeper cascade stages.

It is thus natural to tie the constraint of (V.5), which must hold for early stages, to high false positive rates and the constraint of (V.6) to low false-positive rates. To accomplish this goal we propose the following cost schedule

$$C^{(s)}_{k,l} = \begin{cases} 0 & \text{if } l = k \\ \frac{fp^{(S)}}{fp^{(s)}} & \forall k, l \in \{1, \ldots, M\} \\ \beta & \text{for } k \in \{1, \ldots, M\} \text{ and } l = M+1 \\ 1-\beta & \text{for } k = M+1 \text{ and } l \in \{1, \ldots, M\}. \end{cases} , \tag{V.7}$$

where $C^{(s)}$ is the sth cost matrix, $fp^{(s)}$ the associated false positive rate, S the number of schedule steps, and $\beta \in [0, 1]$. We will assume, for now, that the schedule steps are identical to the the stage numbers. In this case, stage $h_s(x)$ is learned using, in the risk of (V.3), the cost matrix $C^{(s-1)}$, determined by the false-positive rate $fp^{(s-1)}$ of stage $s-1$. We note, however, that this assumption is not strictly necessary and will discuss alternatives later on.

For now, consider the value of $C_{k,l}^{(s)}$ under the cost schedule of (V.7), for $k, l \in \{1, \ldots, M\}$. It follows from the decreasing dependence of the false positive rate on s that $C_{k,l}^{s}$ is an increasing sequence that converges to 1 for the late cascade stages. Hence, it satisfies the constraint of (V.6). Similarly, since the ratio of false positive rates tends to be small in the early stages, the constraint of (V.5) will hold as long as β is close to one. Finally, this also suffices to guarantee that the constraint of (V.4) holds. In summary, the cost schedule of (V.7) has all the desired properties as long as $\beta \approx 1$. We will later see that β remains a free parameter that can be used to tune the detection rate of the cascade. For now, it suffices to note that, since the ratio of false positives changes gradually, the learned cascade is a soft version of the detector of Figure V.1-d. Rather than an abrupt transition from binary to multiclass, the cascade makes a gradual transition.

V.C Boosting multi-resolution cascades

In this section, we introduce a Boosting algorithm to minimize the risk of (V.3) for a particular cost matrix C.

V.C.1 Optimization problem

The proposed Boosting algorithm is inspired by the MCBoost framework, proposed by [68] for learning cost-insensitive multiclass classifiers. Under this framework, the class labels $\{1, \ldots, M+1\}$ are first translated into a set of codewords $\{y_1, y_2, \ldots, y_{M+1}\} \in \mathbb{R}^M$ that form a simplex in $\mathbb{R}^M$, such that $\sum_{i=1}^{M+1} y_i = 0$. These codewords are then used to learn a M-dimensional predictor $F(x) = [f_1(x), f_2(x) \ldots f_M(x)] \in \mathbb{R}^M$, by solving the optimization problem

$$\begin{cases} \min_F & \overline{\mathcal{R}}[F] \\ s.t & F(x) = [f_1(x), f_2(x), \ldots, f_M(x)] \\ & f_j \in span(\mathcal{G}) \quad \forall j \in \{1, \ldots, M\} \end{cases}, \tag{V.8}$$

using coordinate descent in function space [28, 53], where $\overline{\mathcal{R}}[F]$ is a multiclass risk function and $\mathcal{G} = \{g_i\}$ a set of weak learners. Given predictor $F(x)$, the classifier implements the decision rule

$$h(x) = \arg \max_{k=1\ldots M+1} \langle y_k, F(x_i) \rangle. \tag{V.9}$$

V.C.2 Surrogate loss

This procedure cannot be applied directly to the risk of (V.3) because the discontinuity of the indicator function $\mathbb{I}(.)$ prevents the computation of the function gradient. As is common in Boosting algorithms [26], this problem can be avoided with recourse to a surrogate loss function, where the indicator function is replaced by a smooth upper bound. For this, we start by noting that, from the bound $e^{-\frac{1}{2}x} \geq 1 \, \forall x \leq 0$ and (V.9), it follows that if $(x_i, z_i) \in \mathcal{D}$

$$\begin{aligned} \mathbb{I}(h(x_i) = j) &= \mathbb{I}(j = \arg \max_{k=1,\ldots,M+1} \langle y_k, F(x_i) \rangle) \\ &\leq e^{-\frac{1}{2}[\langle y_{z_i}, F(x_i) \rangle - \langle y_j, F(x_i) \rangle]}. \end{aligned} \tag{V.10}$$

Using this bound in (V.2) results in the surrogate risk

$$\overline{\mathcal{R}}^c[F] = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{M+1} C_{z_i,j} e^{-\frac{1}{2}[\langle y_{z_i}, F(x_i) \rangle - \langle y_j, F(x_i) \rangle]}, \tag{V.11}$$

which upper bounds (V.2) for any choice of C. Note that this surrogate risk reduces to the objective function of [92] for binary cost-sensitive learning.

V.C.3 Boosting algorithm

The proposed cost-sensitive multiclass Boosting algorithm solves the optimization problem of (V.8), using the surrogate risk $\overline{\mathcal{R}}^c[F]$ of (V.11), by coordinate descent in the function space spanned by the set of weak learners $\mathcal{G}$. Each Boosting iteration identifies the best weak learner to add to each predictor component $f_k(x)$, using

$$g_k^* = \arg \min_{g \in \mathcal{G}} \delta \overline{\mathcal{R}}^c[F; k, g]. \tag{V.12}$$

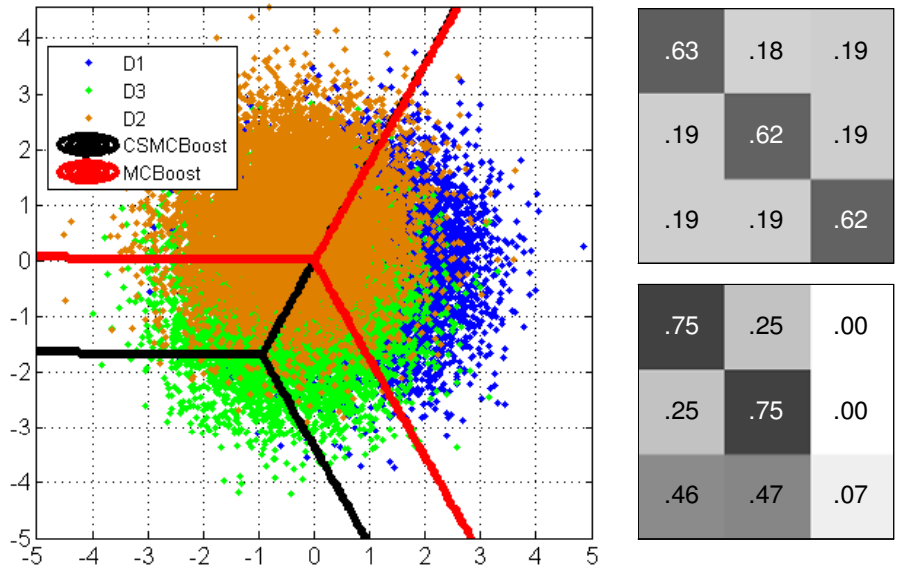


Figure V.2: Impact of the cost matrix C on the decision boundaries (left), and confusion matrix (right) of the proposed Boosting algorithm. Confusion matrices are shown for cost-sensitive (up right) and insensitive (down right) learning.

where is $\delta\overline{\mathcal{R}}^c[F;k,g]$ the directional derivative of $\overline{\mathcal{R}}^c[F]$ for updating $f_k(x)$ along direction $g(x)$. Defining $v_{i,j} = y_i - y_j$,

$$\begin{aligned}\delta\overline{\mathcal{R}}^c[F;k,g] &= \left.\frac{\partial\overline{\mathcal{R}}^c[F+\epsilon g\mathbf{1}_k]}{\partial\epsilon}\right|_{\epsilon=0} \\ &= -\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{M+1} g(x_i)\langle v_{z_i,j},\mathbf{1}_k\rangle C_{z_i,j}e^{-\frac{1}{2}\langle v_{z_i,j},F(x_i)\rangle},\end{aligned} \tag{V.13}$$

where $\mathbf{1}_j \in \mathbb{R}^M$ a vector whose j^{th} element is one and the remaining zero. The optimal step size along the update direction

$$\alpha_j^* = \arg\min_{\alpha\in\mathbb{R}} \overline{\mathcal{R}}^c[F+\alpha g_j^*\mathbf{1}_j], \tag{V.14}$$

then leads to the update

$$F = F + \alpha_{k^*}^* g_{k^*}^* \mathbf{1}_{k^*}, \tag{V.15}$$

where k^* is the coordinate whose update results in the lowest $\overline{\mathcal{R}}^c$.

V.C.4 Penalizing complexity

So far, we have considered the problem of Boosting one multiclass predictor. In this section, we consider the problem of learning a cascade $\mathcal{H}$ of stage predictors $F_1, \dots F_r$ and associated classifiers $h_1, \dots, h_r$, defined according to (V.9). Note that each h_k is a multiclass classifier and

$$F_k(x) = [f_{1,k}(x), \dots f_{M,k}(x)] \in \mathbb{R}^M. \tag{V.16}$$

To ensure consistency with both (V.1) and (V.9), we define a predictor for the whole cascaded detector as

$$\mathcal{F}[F_1, \dots, F_r](x) = \begin{cases} F_r(x) & \text{if } \; h_k(x) \neq M+1 \; \forall k, \\ F_k(x) & \text{if } h_k(x) = M+1 \text{ and } h_j(x) \neq M+1 \; \forall j < k. \end{cases} \tag{V.17}$$

This definition guarantees that (V.9) holds for classifier $\mathcal{H}$ and predictor $\mathcal{F}$.

When learning a classifier cascade, it is usually beneficial to introduce a mechanism that encourages solutions of low computational complexity [8, 69, 64]. In this work, this goal is achieved by learning the cascade predictor $\mathcal{F}[F_1, \dots F_r]$ that minimizes a Lagrangian

$$\mathcal{L}[\mathcal{F}] = \overline{\mathcal{R}}^c[\mathcal{F}] + \eta \overline{T}[\mathcal{F}], \tag{V.18}$$

which trades the surrogate classification risk $\overline{\mathcal{R}}^c[\mathcal{F}]$ of (V.11) with a complexity measure $\overline{T}[\mathcal{F}]$ (see section V.D) that penalizes cascades of high computational complexity. The Lagrange multiplier η controls the trade-off between accuracy and complexity. When $\eta = 0$ the learning algorithm encourages solutions that minimize the classification risk, for larger values of η this risk is weighed against the computational complexity of the associated cascade.

The cascade Boosting algorithm then solves

$$\begin{cases} \min_{F_1, \dots, F_r, r} \quad \mathcal{L}(\mathcal{F}[F_1, \dots, F_r]) \\ \\ s.t \qquad F_k(x) = [f_{k,1}(x), f_{k,2}(x) \dots f_{k,M}(x)], \\ \qquad\quad\; f_{k,j} \in span(\mathcal{G}) \quad \forall j = 1 \dots M, k = 1 \dots r \end{cases}, \tag{V.19}$$

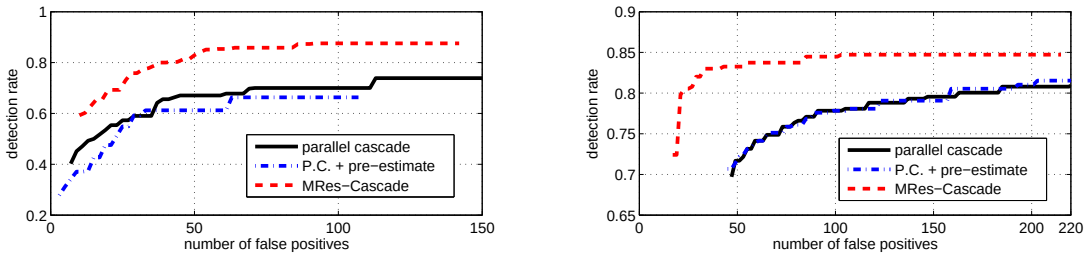


Figure V.3: ROCs of multi-view car detection(left) and traffic sign detection(right).

Table V.1 Multi-view car detection performance at 100 false positives.

	car detection			traffic sign detection		
Method	*cmp.*	*accu.*	det. rate	*cmp.*	*accu.*	det. rate
parallel cas. [96]	59.94	0.35	0.72	10.08	0.78	0.78
p.c. + pre-est. [90]	$15.15 + 6$	0.35	0.70	$2.32 + 4$	0.78	0.78
MRes cas.	**16.40**	**0.58**	**0.88**	**5.56**	**0.84**	**0.84**

using coordinate descent in the function space spanned by the weak learner set $\mathcal{G}$ [28, 53]. This is a simple extension of the Boosting algorithm of Section V.C.3. The main difference is that, in addition to Boosting the predictor stages F_k, there is a need to determine the cascade configuration, i.e. how many stages it contains and how many weak learners compose the classifier of each stage. This structure is determined with a greedy strategy that follows naturally from the Boosting procedure. The cascade is initialized as an empty classifier. At each Boosting iteration, two possibilities are considered: 1) updating the last cascade stage, or 2) introducing a new stage at the end of the cascade. This is done in two steps. First, the best update is determined for each coordinate of both the last cascade stage and the new additional stage, using (V.12) and (V.14). Second, among the $2M$ possible updates, the one that further decreases the Lagrangian of (V.18) is chosen, and the cascade updated accordingly, using (V.15).

V.C.5 Multi-resolution cascades

So far, we have assumed a single cost matrix C. This produces an M-class classifiers. In Section V.B.4, we have discussed how to extend this procedure to the design of MRes cascades, by introducing a sequence of cost matrices $C^{(s)}$. In that section, we have assumed a different cost matrix per cascade stage. This could easily be implemented in the cascade Boosting algorithm above, by changing the cost matrix every time a new stage is added to the cascade. In practice, however, the multi-resolution behavior is not the only goal that remains to be achieved and there are two additional problems that need to be addressed.

The first is to guarantee that the cascade maintains a target detection rate D throughout the learning process. The second is that the set of training examples is frequently not large enough to learn a cascade of low false positive rate. While it is possible to represent the target classes $1, \ldots, M$ with a few hundred training instances, the complexity of the negative class makes this impossible for class $M + 1$. Although after a small number of Boosting iterations the cascade rejects all negative training examples, there is no guarantee that it has a low false-positive rate outside the training set. The standard solution, in the cascade literature, is to rely on the bootstrapping procedure [84, 91]. This is an efficient strategy to sample examples from a large negative class. It consists of gradually replacing negative examples rejected by the current cascade with new false-positives, extracted from a validation set. Since the design of any cascade of practical interest requires bootstrapping, we tie the cost schedule to bootstrapping iterations, rather than cascade stages. That is, rather than using the stage number as the variable s of section V.B.4, we set this variable to the bootstrapping iteration. This provides some robustness against the greedy nature of the cascade growing process discussed above.

It remains to guarantee that the cascade maintains the target detection rate D throughout the training process. For this, we return to the cost matrix of (V.7). In section V.B.4, we saw that the constraints of (V.4)-(V.6) hold as long

as $\beta \approx 1$. To guarantee this, the cascade learning algorithm is initialized with a high value of β, typically $\beta = 0.9$. The resulting cost matrix is then used to determine the best cascade update. If the detection rate of the updated cascade is less than the target D this update is discarded, β is updated to $\beta = \frac{1+\beta}{2}$ and the procedure repeated. Since the update can only increase β, the constraints of (V.4)-(V.6) continue to hold. Since increasing β increases the cost $C_{k,M+1},\ k = 1, \ldots, M$ of target misses, the new update produces a cascade of higher detection rate. Hence, tuning β enables the cascade to eventually achieve the target detection rate. In summary, the procedure is guaranteed to produce a MRes cascade that holds the target detection rate throughout training.

The overall procedure for learning MRes cascades is summarized in algorithm 9. Note that S_L is the set of training examples that reach the last cascade stage. Similarly, S_A is the set of training examples not rejected by the current cascade stages, i.e. which will reach a new stage appended to the end of the cascade. Finally, the algorithm terminates when the cascade achieves a specified false-positive rate FP on the training set. Note that this is the value used for $fp^{(S)}$ in the cost schedule of (V.7).

V.D Experiments

The proposed learning algorithm was tested on a synthetic data set, and the tasks of multi-view car detection, and multiple traffic sign detection. The resulting detector, denoted *MRes cascade*, was compared to the detectors of Figure V.1. Since it has been established in the literature that the all-class detector with post-estimation has poor performance [90], the comparison was limited to parallel cascades [96] and parallel cascades with pre-estimation [90]. All binary cascade detectors were learned with a combination of the ECBoost algorithm of [69] and the cost-sensitive Boosting method of [92]. Following [60], all cascaded detectors used integral channel features and trees of depth two as weak learners. The train-

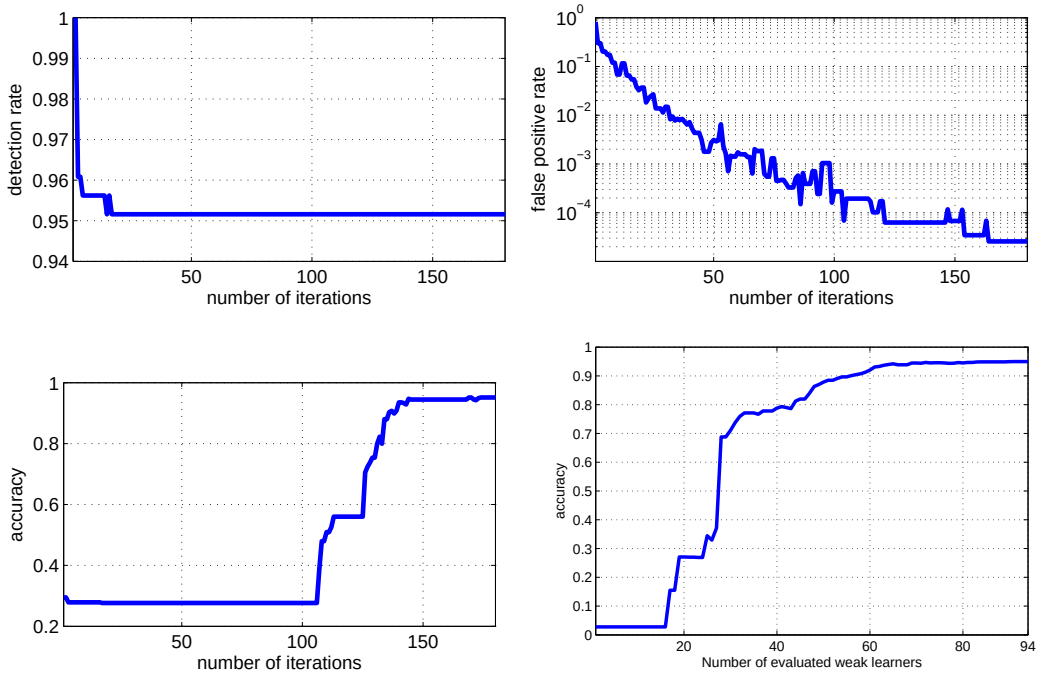


Figure V.4: Evolution of detection rate (top-left), false positive rate $fp^{(s)}$ (top-right), and accuracy (bottom-left) of the MRes cascade during learning for car detection problem. The (bottom-right) is evolution of the accuracy of the cascade during training a traffic sign detector for 17 traffic signs.

ing parameters were set to $\eta = 0.02$, $D = 0.95$, $FP = 10^{-6}$ and the training set was bootstrapped whenever the false positive rate dropped below 90%. Bootstrapping also produced an estimate of the false positive rate $fp^{(s)}$, which was used to define the cost matrix $C^{(s+1)}$. As in [90], detector cascade with pre-class estimation used tree classifiers for the pre-estimation. In the remainder of this section, detection rate is defined as the percentage of target examples, from all views or target classes, that were detected. Detector accuracy is the percentage of the target examples that were detected and assigned to the correct class. Finally, detector complexity is the average number of tree node classifiers evaluated per example.

Synthetic data: We start by illustrating the behavior of the algorithm of Section V.C.3 in the synthetic data set of Figure V.2-left. This contains $3 \times 10,000$ samples, randomly drawn from three Gaussian distributions of means $[0.8, 0]^T$, $[-0.4, 0.7]^T$, $[-0.4, -0.7]^T$ and variance $[1, 0; 0, 1]$. Multiclass classifiers

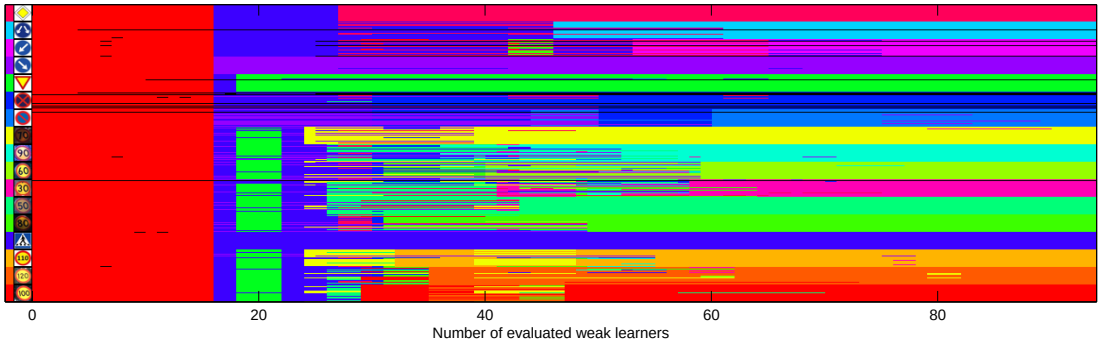


Figure V.5: Evolution of MRes cascade decisions for 20 randomly selected examples of various traffic signs. Each row illustrates the evolution of the label assigned to one example. The color of kth pixel signals the label after the evaluation of k weak learners. The traffic signs and their label colors are shown on the left.

were learned with both a cost-insensitive version of the Boosting algorithm (all non-diagonal entries of the cost matrix set to 1) and the cost sensitive version with costs $C = [0, 1, 10; 1, 0, 10; 1, 1, 0]$. The decision boundary of the classifiers, along with their confusion matrices, are shown in Figure V.2. As expected, the cost-sensitive classifier is more averse to assigning examples from the first two classes to the third. This can be seen by the shifted decision boundary and the confusion matrices. This experiment shows that the proposed Boosting method is able to enforce the cost structure encoded in the cost matrix C.

Multi-view Car Detection: To train a multi-view car detector, we collected images of 128 Frontal, 100 Rear, 103 Left, and 103 Right car views. These were resized to 41×70 pixels. The multi-view car detector was evaluated on the USC car data set [38], which consists of 197 color images of size 480×640, containing 410 instances of cars in different views.

The performance of the various cascades was compared through ROCs, which are shown in Figure V.3-a, as well as detection rate, accuracy and complexity, which are reported in Table V.1. The complexity of parallel cascades with pre-processing is broken up into the complexity of the cascade plus the complexity of the pre-estimator. Figure V.3-a, shows that the MRes cascade has significantly

Figure V.6: Examples of detections of cars from different views and several traffic signs.

better ROC performance than any of the other detectors. This is partially due to the fact that the detector is learned jointly across classes and thus has access to more training examples. In result, there is less overfitting and better generalization. Furthermore, as shown in Table V.1, the MRes cascade is much faster. The 3.5-fold reduction of complexity over the parallel cascade suggests that MRes cascades share features very efficiently across classes. The MRes cascade also detects 16% more cars and assigns 23% more cars to the true class. The parallel cascade with pre-processing was slightly less accurate than the parallel cascade but three times as fast. However, the complexity of the pre-estimator still makes it 20% slower than the MRes cascade.

Figure V.4 shows the detection rate, false positive rate, and accuracy of the MRes cascade during training. Note that the detection rate is above the specified $D = 95\%$ throughout the learning process. This is due to the updating of the β parameter of (V.7). It can also be seen that, while the false positive rate decreases gradually, accuracy remains low for many iterations. This shows that the early stages of the MRes cascade place more emphasis on rejecting negative examples (lowering the false positive rate) than making precise view assignments

for the car examples. This reflects the cost schedule $C^{(s)}$ of (V.7). Early on, (V.5) holds and confusion between classes has little cost. However, as the cascade grows and its false positive rate $fp^{(s)}$ decreases, the entries $C^{(s)}_{k,l}$ $k, l \neq M + 1$ become comparable to the remaining entries of the cost matrix and the detector starts to distinguish different car views. This happens soon after iteration 100, where the false-positive rate flattens but the accuracy starts to increase drastically. In this way, the MRes cascade behaves as a soft version of the all-class detector cascade with post-estimation, shown in Figure V.1-d.

Traffic Sign Detection: For the detection of traffic signs, we extracted $1,159$ training examples from the first set of the Summer traffic sign data set [41]. This produced 660 examples of "priority road", 145 of "pedestrian crossing", 232 of "give way" and 122 of "no stopping no standing" signs. For training, these images were resized to 40×40. For testing, we used 357 images from the second set of the Summer data set which contained at least one visible instance of the traffic signs, with more than 35 pixels of height. The performance of different traffic sign detectors is reported in Figure V.3-Right and Table V.1. Again, the MRes cascade was faster and more accurate than the others. In particular, it was faster than other methods, while detecting/recognizing 6% more traffic signs.

We next trained a MRes cascade for detection of the 17 traffic signs shown in the left end of Figure V.5-left. Figure V.5-right shows the evolution of the accuracy of this detector and Figure V.5-left the evolution of MRes cascade decisions for examples of different classes. In this figure, each row of color pixels illustrates the evolution of one example. The the color of kth pixel in a row signals the decision made by the cascade after k weak learners. The traffic signs and corresponding colors are shown in the left of the figure. For each class we randomly selected 20 examples and show the corresponding decision sequence next to the class picture.

Note that the early cascade stages assign most examples to the first class. Only a few examples were rejected by these stages. This corresponds to a high de-

tection rate but very low accuracy. However, as more weak learners are evaluated, the detector starts to create some intermediate categories. For example, after 20 weak learners, all traffic signs containing red and yellow colors are assigned to the "give way" class. Evaluating more weak learners further separates these classes and almost all examples are assigned correctly on the right side of the picture. This shows that besides being a soft version of the all-class detector cascade, the MRes cascade automatically groups the classes into an internal taxonomy based on visual similarity.

Finally, although we have not produced detection ground truth for this experiment, we have empirically observed that the final 17-traffic sign MRes cascade is accurate and has low complexity (5.15). This make it possible to use the detector in real-time on low complexity devices, such as smart-phones. A video illustrating the detection results is available in the supplementary material.

V.E Conclusion

We proposed that a multiclass detector cascade should have multi-resolution behavior where early stages should classify target vs. non-target patches with high detection rate and late stages are multiclass classifiers of high accuracy and complexity to distinguish between target classes. We showed that learning such cascade detector is possible by using a cost-sensitive multiclass Boosting algorithm and adjusting the cost factors dynamically. For these adjustments, we used false positive rate of the cascade detector during the training process and showed that it will result in cascade detectors with multi-resolution behavior. Using this strategy we derived a Boosting algorithm for learning multiclass detector cascades. Experiments on the problems of multi-view car detection and simultaneous detection of multiple traffic signs showed that the proposed detector is faster and more accurate than previous methods.

Algorithm 9 MRes Cascade Learning

Input: $M+1$-ary training set S_t containing target and negative examples, a validation set for bootstrapping, target detection and false positive rates (D, FP) for the cascade, Lagrange multiplier η.

Initialization:

Produce codewords $y^1 \dots y^M$ and translate example labels into codewords, using the procedure of [68].

Define the sets $S_L = S_A = S_t$, and set $s = 1$, and $fp^{(s)} = 1$.

while $fp^{(s)} > FP$ **do**

  for $j = 1$ to M **do**

    Find the best weak learner for updating the jth coordinate of the predictor of the last stage of the cascade and compute its Lagrangian, using (V.12), (V.14), (V.18), (V.7), an appropriate β and example set S_L.

    Find the best weak learner for updating the jth coordinate of a new stage, appended at the end of the cascade, and compute is Lagrangian, using example set S_A in (V.12), (V.14), (V.18), (V.7) and an appropriate β.

  end for

  Update the cascade, by updating the weak learner of smaller Lagrangian.

  if adding a new stage **then**

    Set $S_L = S_A$.

  end if

  Remove examples that are rejected, by current last stage of the cascade, from S_A.

  if more than 10% of negative examples in the current trainingf set are rejected **then**

    bootstrap, update the training set S_t, increase s, and estimate $fp^{(s)}$ using the validation set.

    Set $S_L = S_A = S_t$.

  end if

end while

Chapter VI

Conclusions

In this dissertation we considered the problem of designing real-time multiclass object detectors. Designing a fast and accurate multiclass object detector requires addressing several challenges. We started by proposing TaylorBoost. TaylorBoost explains Boosting algorithms as iterative descent algorithms for minimizing Taylor series expansion of the risk of classification in the function space. Using this framework, it is possible to derive first-order, equivalent to gradient descent, and second order, equivalent to Newton method, Boosting algorithms. We then used TaylorBoost in the rest of the dissertation to derive appropriate Boosting algorithms based on the requirements of the problems. We next considered the problem of learning optimal detector cascades and proposed FCBoost. This algorithm optimizes a Lagrangian risk that accounts for both detector speed and accuracy with respect to a predictor that complies with the sequential decision making structure of the cascade architecture. By exploiting recursive properties of the latter, it was shown that many cascade predictors can be derived from generator functions, which are cascade predictors of two stages. Variants of FCBoost were derived for two members of this family, last-stage and multiplicative cascades, which were shown to generalize the popular independent and embedded stage cascade architectures. The concept of neutral predictors was exploited to integrate the search for cascade configuration into the Boosting algorithm. In result, FCBoost can automatically determine 1) the number of cascade stages and 2) the number of weak learners per stage, by minimizing the Lagrangian risk. It was also shown that FCBoost generalizes AdaBoost, and is compatible with existing cost-sensitive extensions of Boosting. Hence, it can be used to learn cascades of high detection rate. Experimental evaluation has shown that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

We started transition for multiclass object detectors by considering the problem of multiclass Boosting. We proposed a new multiclass Boosting framework, based on multi-dimensional codewords and predictors. The optimal set of codewords is derived and a margin enforcing loss is proposed. The resulting loss is

then minimized by first order TaylorBoost on a multi-dimensional function space resulting in two algorithms: 1) CD-MCBoost, based on coordinate descent, which updates one predictor component at a time, 2) GD-MCBoost, based on gradient descent, which updates all components jointly. The algorithms differ in the weak learners that they support but are both shown to be 1) Bayes consistent, 2) margin enforcing, and 3) convergent to the global minimum of the risk. They also reduce to AdaBoost when there are only two classes. Experiments showed that both methods outperform previous multiclass Boosting approaches on a number of datasets.

Combining the proposed cascade learning algorithm, FCBoost, and multiclass Boosting method, MCBoost, made it possible to learn detector cascade for detecting multiple objects. The remaining challenge was to account for the cost-sensitive nature of sub-classifiers in the cascade sequence. We proposed that a multiclass detector cascade should have multi-resolution behavior where early stages should classify target vs. non-target patches with high detection rate and late stages are multiclass classifiers of high accuracy and complexity to distinguish between target classes. We showed that learning such cascade detector is possible by using a cost-sensitive multiclass Boosting algorithm and adjusting the cost factors dynamically. For these adjustments, we used false positive rate of the cascade detector during the training process and showed that it will result in cascade detectors with multi-resolution behavior. Using this strategy we derived a Boosting algorithm for learning multiclass detector cascades. Experiments on the problems of multi-view car detection and simultaneous detection of multiple traffic signs showed that the proposed detector is faster and more accurate than previous methods.

Bibliography

[1] "http://theoval.cmp.uea.ac.uk/ gcc/matlab/index.shtml."

[2] "http://www.svcl.ucsd.edu/projects/mcboost/."

[3] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Trans. on PAMI*, vol. 26, pp. 1475 –1490, 2004.

[4] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, September 2001. [Online]. Available: http://dx.doi.org/10.1162/15324430152733133

[5] S. Avidan, "Ensemble tracking," *IEEE PAMI*, vol. 29, no. 2, pp. 261–271, 2007.

[6] P. Bartlett and M. Traskin, "Adaboost is consistent," *Journal of Machine Learning Research*, vol. 8, pp. 2347–2368, Dec. 2007.

[7] V. Blanz, B. Scholkopf, H. Bultho, C. Burges, V. Vapnik, and T. Vetter, "Comparison of view-based object recognition algorithms using realistic 3d models," in *Artificial Neural Networks ICANN96*. Springer, 1996, pp. 251–256.

[8] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *CVPR*, 2005, pp. 236–243.

[9] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *International Journal Computer Vision*, vol. 77, pp. 65–86, 2008.

[10] G. Carneiro, B. Georgescu, S. Good, and D. Comaniciu, "Detection and measurement of fetal anatomies from ultrasound images using a constrained probabilistic boosting tree," *IEEE Transactions on Medical Imaging*, vol. 27, no. 9, pp. 1342 –1355, sept. 2008.

[11] M. Collins, R. Schapire, and Y. Singer, "Logistic regression, adaboost and bregman distances," *Machine Learning*, vol. 48, no. 1-3, pp. 253–285, 2002.

[12] H. Coxeter, *Regular Polytopes*. Dover Publications, 1973.

[13] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[14] ——, "On the learnability and design of output codes for multiclass problems," *Machine Learning*, vol. 47, pp. 201–233, May 2002.

[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[16] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[17] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.

[18] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[19] S. Du, N. Zheng, Q. You, Y. Wu, M. Yuan, and J. Wu, "Rotated haar-like features for face detection with in-plane rotation," in *Proceedings of international conference on Interactive Technologies and Sociotechnical Systems*, 2006, pp. 128–137.

[20] J. Duchi and Y. Singer, "Boosting with structural sparsity," *Proceedings of the International Conference on Machine Learning*, pp. 297–304, 2009.

[21] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[22] M. M. Dundar and J. Bi, "Joint optimization of cascaded classifiers for computer aided detection," in *CVPR*, 2007.

[23] G. Eibl and R. Schapire, "Multiclass boosting for weak classifiers," in *Journal of Machine Learning Research*, 2005, pp. 6–189.

[24] P. Felzenszwalb, R. Girshick, and D. M. andD. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[25] Y. Freund and R. Schapire., "Experiments with a new boosting algorithm," in *Proceedings of International Conference on Machine Learning*, 1996, pp. 148–156.

[26] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Comp. and Sys. Science*, 1997.

[27] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, pp. 337–407, 2000.

[28] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 1999.

[29] B. Frigyik, S. Srivastava, and M. Gupta, "An introduction to functional derivatives," *Technical Report(University of Washington)*, 2008.

[30] T. Gao and D. Koller, "Multiclass boosting with hinge loss based on output coding," in *Proceedings of International Conference on Machine Learning*, 2011.

[31] Y. Guermeur, "Vc theory of large margin multi-category classifiers," *Journal of Machine Learning Research*, vol. 8, pp. 2551–2594, December 2007.

[32] V. Guruswami and A. Sahai, "Multiclass learning, boosting, and error-correcting codes," in *Proceedings of Annual Conference on Computational Learning Theory*, 1999, pp. 145–155.

[33] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *NIPS*, 1998.

[34] R. A. Horn and C. R. Johnson, Eds., *Matrix Analysis*. New York, NY, USA: Cambridge University Press, 1986.

[35] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[36] C. Huang, H. Ai, Y. Li, and S. Lao, "High-performance rotation invariant multiview face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 671–686, 2007.

[37] J. Huang, S. Ertekin, Y. Song, H. Zha, and C. Giles, "Efficient multiclass boosting classification with active learning," in *SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2007.

[38] C.-H. Kuo and R. Nevatia, "Robust multi-view car detection using unsupervised sub-categorization," in *Applications of Computer Vision (WACV), 2009 Workshop on*, 2009, pp. 1–8.

[39] C. Lampert, "An efficient divide-and-conquer cascade for nonlinear object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1022 –1029.

[40] C. Lampert, M. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2129–2142, 2009.

[41] F. Larsson, M. Felsberg, and P.-E. Forssen, "Correlating Fourier descriptors of local patches for road sign recognition," *IET Computer Vision*, vol. 5, no. 4, pp. 244–254, 2011.

[42] L. Lefakis and F. Fleuret, "Joint cascade optimization using a product of boosted classifiers," in *Proceedings of the Neural Information Processing Systems Conference*, 2010.

[43] L. Li, "Multiclass boosting with repartitioning," in *Proceedings of International Conference on Machine Learning*, 2006, pp. 569–576.

[44] S. Li and Z. Zhang, "Floatboost learning and statistical face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112 –1123, 2004.

[45] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings of International Conference on Image Processing*, 2002, pp. I–900 – I–903 vol.1.

[46] C. Liu and H.-Y. Shum, "Kullback-leibler boosting," in *CVPR*, vol. 1, 2003, pp. 587 – 594.

[47] H. Luo, "Optimization design of cascaded classifiers," in *CVPR*, vol. 1, 2005, pp. 480 – 485.

[48] V. Mahadevan and N. Vasconcelos, "Saliency-based discriminant tracking," in *CVPR*, 2009.

[49] H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 294 –309, 2011.

[50] ——, "High detection-rate cascades for real-time object detection," in *ICCV*, vol. 2, 2007, pp. 1–6.

[51] ——, "Variable margin losses for classifier design," in *NIPS*, 2010.

[52] H. Masnadi-Shirazi, N. Vasconcelos, and V. Mahadevan, "On the design of robust classifiers for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[53] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *In Advances in Neural Information Processing Systems*, 2000, pp. 512–518.

[54] D. Mease and A. Wyner, "Evidence contrary to the statistical view of boosting," *Journal of Machine Learning Research*, vol. 9, pp. 131–156, 2008.

[55] C. Messom and A. Barczak, "Fast and efficient rotated haar-like features using rotated integral images," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2006.

[56] I. Mukherjee and R. Schapire., "A theory of multiclass boosting," in *NIPS*, 2010.

[57] N. Nilsson, *Learning machines*. New York: McGraw-Hill, 1965.

[58] J. Nocedal, S., and Wright, *Numerical Optimization*. New York: Springer Verlag, 1999.

[59] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *CVPR*, 1997, pp. 193–99.

[60] P. P. P. Dollar, Z. Tu and S. Belongie, "Integral channel features," in *BMVC*, 2009.

[61] X. Perrotton, M. Sturzel, and M. Roux, "Implicit hierarchical boosting for multi-view object detection," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 958–965.

[62] M. Pham and T. Cham, "Fast training and selection of haar features using statistics in boosting-based face detection," in *Proceedings of IEEE International Conference on Computer Vision*, 2007, pp. 1–7.

[63] M. Pham, Y. Gao, V. Hoang, and T. Cham, "Fast polygonal integration and its application in extending haar-like features to improve object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 942 –949.

[64] M.-T. Pham, V.-D. Hoang, and T.-J. Cham, "Detection with multi-exit asymmetric boosting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1 –8.

[65] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 829 – 836.

[66] V. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: Tradeoff between accuracy and cost," in *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 853–860.

[67] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, pp. 533–536, 1968.

[68] M. Saberian and N. Vasconcelos, "Multiclass boosting: Theory and algorithms," in *NIPS*, 2011.

[69] ——, "Learning optimal embedded cascades," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 32 005 –2018, 2012.

[70] M. J. Saberian, H. Masnadi-Shirazi, and N. Vasconcelos, "Taylorboost: First and second order boosting algorithms with explicit margin control," in *CVPR*, 2010.

[71] M. J. Saberian and N. Vasconcelos, "Boosting classifer cascades," in *NIPS*, 2010.

[72] R. Schapire, "Using output codes to boost multiclass learning problems," in *Proceedings of International Conference on Machine Learning*, 1997, pp. 313–321.

[73] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, pp. 297–336, December 1999.

[74] R. E. Schapire, "Drifting games," in *Proceedings of the twelfth annual conference on Computational learning theory*, 1999, pp. 114–124.

[75] B. Schlkopf, C. Burges, and V. Vapnik, "Extracting support data for a given task," in *Proceedings, First International Conference on Knowledge Discovery and Data Mining, Menlo Park*. AAAI Press, 1995, pp. 252–257.

[76] H. Schneiderman, "Feature-centric evaluation for efficient cascaded object detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 29–36, 2004.

[77] T. Sejnowski and C. Rosenberg, "Parallel networks that learn to pronounce english text," *Journal of Complex Systems*, vol. 1(1), pp. 145–168, 1987.

[78] C. Shen, S. Paisitkriangkrai, and J. Zhang, "Efficiently learning a detection cascade with sparse eigenvectors," *IEEE Transactions on Image Processing*, vol. 20, no. 1, pp. 22 –35, jan. 2011.

[79] C. Shen, P. Wang, and H. Li, "Lacboost and fisherboost: optimally building cascade classifiers," in *Proceedings of European Conference on Computer Vision*, 2010, pp. 608–621.

[80] D. R. Smith, *Variational Methods in Optimization*. Prentice-Hall, 1974.

[81] J. Sochman and J. Matas, "Waldboost-learning for time constrained sequential detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 150–156.

[82] J. Sun, J. M. Rehg, and A. Bobick, "Automatic cascade training with perturbation bias," in *Proceedings of Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 276–283.

[83] Y. Sun, S. Todorovic, J. Li, and D. Wu, "Unifying the error-correcting and output-code adaboost within the margin framework," in *Proceedings of International Conference on Machine Learning*, 2005, pp. 872–879.

[84] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. on PAMI*, vol. 20, pp. 39 –51, 1998.

[85] R. Tammes, "On the origin of number and arrangement of places of exit on the surface of pallen grains," *Rec. Trav. Bot. Neerl.*, vol. 27, pp. 1–84, 1930.

[86] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 5, pp. 854–869, 2007.

[87] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1713 –1727, 2008.

[88] V. Vapnik, *Statistical Learning Theory.* John Wiley Sons Inc, 1998.

[89] S. Vijayanarasimhan and K. Grauman, "Efficient region search for object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1401 –1408.

[90] M. Viola, M. J. Jones, and P. Viola, "Fast multi-view face detection," in *Proc. of Computer Vision and Pattern Recognition*, 2003.

[91] P. Viola and M. Jones, "Robust real-time object detection," *Workshop on Statistical and Computational Theories of Vision*, 2001.

[92] ——, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *NIPS*, 2002, pp. 1311–1318.

[93] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1030–1037.

[94] X. Wang, T. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *Proceedings of IEEE International Conference on Computer Vision*, 2009, pp. 32–39.

[95] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *Proceedings of European Symposium On Artificial Neural Networks*, 1999, pp. 219–224.

[96] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, 2004, pp. 79–84.

[97] J. Wu, S. Brubaker, M. Mullin, and J. Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 369–382, 2008.

[98] R. Xiao, H. Zhu, H. Sun, and X. Tang, "Dynamic cascades for face detection," *ICCV*, pp. 1–8, 2007.

[99] R. Xiao, L. Zhu, and H.-J. Zhang, "Boosting chain learning for object detection," in *ICCV*, 2003, pp. 709 –715.

[100] B. Zhang, G. Ye, Y. Wang, J. Xu, and G. Herman, "Finding shareable informative patterns and optimal coding matrix for multiclass boosting," in *Proceedings of International Conference on Computer Vision*, 2009, pp. 56 –63.

[101] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4689 –4708, july 2011.

[102] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Statistics and Its Interface*, vol. 2, pp. 349–3660, 2009.

[103] Q. Zhu, Q. Zhu, S. Avidan, S. Avidan, M. chen Yeh, M. chen Yeh, K. ting Cheng, and K. ting Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *In CVPR*, 2006, pp. 1491–1498.

[104] H. Zou, J. Zhu, and T. Hastie, "New multicategory boosting algorithms based on multicategory fisher-consistent losses," *Annals of Statistics*, vol. 2, 2008.