Principled asymmetric boosting approaches to rapid training and classification in face detection

Pham, Minh Tri

2009

Pham, M. T. (2009). Principled asymmetric boosting approaches to rapid training and classification in face detection. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/15664>

<https://doi.org/10.32657/10356/15664>

Principled Asymmetric Boosting Approaches to Rapid Training and Classification in Face Detection

Pham Minh Tri

Division of Computing Systems
School of Computer Engineering

*A thesis submitted to the Nanyang Technological University
in fulfillment of the requirements for the degree of
Doctor of Philosophy*

Jan 2009

Acknowledgments

This thesis summarizes the research work done during my Ph.D. candidature at the School of Computer Engineering, Nanyang Technological University, Singapore. Over this period of time I have received considerable help from many people, but none more so than from my supervisor Dr. Cham Tat Jen. His thoughtful guidance, invaluable suggestions and constant encouragement has always been the source of inspiration in my research. What's more, I have also learned a lot about how to be a qualified researcher from his continual enthusiasm, passion and serious attitude towards research, for which I am extremely grateful.

I would also like to thank my seniors - Chen Xi, Song Peng, Pong Hon Keat and Feng Zhou, my juniors - Rudianto Sugiyarto, Huang Yi, Hoang Duc Viet Dung, and Li Gang, my friends around the world - Nguyen Huy Thanh Vu, Tran Le Hong Du, Nguyen Minh Hoai, Wu Bo, Wu Jianxin, Helmut Grabner, and Sebastian Nowozin for their insightful discussions and suggestions in relevant research. Thanks should also go to my friends in the Center for Multimedia and Network Technology (CeMNet), including Zhang Peng, Sudhan Majhi, Mai Ngoc Son, Ngo Viet Anh, Dang Duc Nguyen, Behrouz Saghafi Khadem, Luong Trung Tuan, Pham Duc Minh, Viswanath Gopalakrishnan, Nghiem Viet Nam, Lu Feng, and Hu Yiqin for their warm help and encouragement.

With regard to the work of online asymmetric boosting, I would like to thank Chen Xi, Song Peng, Pham Minh Tuan, Vo Thi Thanh Giang, and Nguyen Huy Thanh Vu for their thorough reviews on our paper published and presented in the IEEE Computer

Society Conference on Computer Vision in Minneapolis, Minnesota, USA in 2007.

With regard to the work of fast training and selection of Haar-like features using statistics, I would like to thank Wu Bo, Wu Jianxin, Huang Chang for their insightful discussions, and Pham Minh Tuan, Vo Thi Thanh Giang, Nguyen Huy Thanh Vu, Pham Duc Minh, Luong Trung Tuan, Nguyen Minh Nhut, and Dang Xuan Hong for their thorough reviews on our paper published and presented in the International Conference on Computer Vision in Rio de Janeiro, Brazil in 2007.

With regard to the work of multi-exit asymmetric boosting, I would like to thank Xiao Rong and Wu Jianxin for having graciously shared their databases, Pham Minh Tuan, Vo Thi Thanh Giang and Nguyen Huy Thanh Vu for their thorough reviews, and especially my junior Hoang Duc Viet Dung for his solid collaboration on our paper published and presented in the IEEE Computer Society Conference on Computer Vision in Anchorage, Alaska, USA in 2008.

Thanks also go to Luong Trung Tuan, Nghiem Viet Nam, Nguyen Viet Anh, Hoang Duc Viet Dung, and Vo Thi Thanh Giang for their many contributions in helping me to revise the thesis' manuscript.

I would like to take this opportunity to thank Nanyang Technological University for providing me the research scholarship. I would also like to thank the Head of Division - Dr. Chia Liang Tien, Clement, and CeMNet's lab technicians - Chua Poo Hua and Oh Hwee May for their help to the convenient access to the computing facilities.

Finally, I wish to express my deepest gratitude to my parents Pham Thuong Cat and Phan Minh Tan, my older brother Pham Minh Tuan, and my beloved girlfriend Vo Thi Thanh Giang for their unfailing love and constant support and care.

Contents

Acknowledgments	i
Abstract	xiii
Publications	xv
Nomenclature	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Topics and Approaches of this Thesis	7
1.2.1 Online Learning of An Asymmetric Boosted Classifier	7
1.2.2 Fast training of Haar-like features using Statistics	8
1.2.3 Choosing an Asymmetric Goal	10
1.2.4 Bounding a True Asymmetric Error from its corresponding Em- pirical Asymmetric Error	10
1.3 Thesis Outline	11
2 Overview of Appearance-based Face Detection	13
2.1 Cascade Design	15
2.2 Learning a Boosted Classifier	19
2.3 Learning a Weak Classifier	22

2.3.1	Haar-like Features	24
2.3.2	Haar-like Feature Classifier	28
3	Online Asymmetric Boosting	29
3.1	Introduction	29
3.2	Online Asymmetric Boosting	31
3.2.1	Why Equal Skewness?	33
3.2.2	Balancing the Skewness Presented to Weak Classifiers	36
3.2.3	Online Learning	42
3.2.3.1	Propagating the Weights	43
3.2.3.2	Estimating c_m	46
3.3	Experimental Results	47
3.3.1	Online Boosting	47
3.3.2	Online Co-training for Moving Objects Detection	49
3.3.3	Online Learning for Face Detection	50
3.4	Conclusions	53
4	Fast Training and Selection of Haar-like Features	55
4.1	Introduction	55
4.2	Training a Weak Classifier using Haar-like Features	60
4.2.1	Training a Boosted Classifier in the Cascade	60
4.2.2	Linear Relationship between Integral Image and Feature Values	61
4.2.3	Training Feature Classifiers using Statistics	62
4.2.4	Time Complexity	66
4.3	Implementation and Experimental Results	67
4.3.1	Experiment Setup	67
4.3.2	Training Time of a Weak Classifier	68
4.3.3	Performance of Boosted Classifiers trained with Different Methods	70

4.3.4	Final Performance	71
4.4	Relation to Other Methods	73
4.5	Conclusions	76
5	Multi-exit Asymmetric Boosting	77
5.1	Introduction	77
5.2	Overview	80
5.2.1	General Framework	80
5.2.2	Related Work	82
5.3	Multi-exit Asymmetric Boosting	85
5.3.1	Overview of the Method	85
5.3.2	Bayes Asymmetric Error	86
5.3.3	Analysis of Asymmetric Goals	88
5.3.4	Updating Weights	93
5.4	Experimental Results	94
5.4.1	Setup	94
5.4.2	Training with Different Asymmetric Goals	94
5.4.3	Final Performance	97
5.5	Conclusions and Future Work	101
6	Margin-based Bounds on an Asymmetric Error	103
6.1	Introduction	103
6.1.1	Related Work	104
6.2	Main Results	107
6.2.1	Bounds for a General Class of Binary Classifiers	108
6.2.2	Bounds for Convex Combinations of Classifiers	113
6.3	Discussion and Conclusions	115

7	Conclusions and Future Work	117
7.1	Summary of Research	117
7.2	Future Work	119
7.2.1	Learning a Weak Classifier	119
7.2.2	Learning a Boosted Classifier	120
7.2.3	Beyond Learning a Classifier	121
 Appendices:		
A	Derivations for Training a Haar-like Feature Classifier	125
B	Probability Tools	129
B.1	Basic Tools	129
B.2	Concentration Inequalities	130
	 Bibliography	 135

List of Figures

1.1	Face detection in action	1
1.2	Cascade of classifiers for face detection	3
1.3	Nineteen types of Haar-like features used in our experiment	9
2.1	Three levels of the VJ framework.	16
2.2	The mother 2D Haar wavelets used by Papageorgiou <i>et al.</i> [57]; (a): “vertical”, (b): “horizontal”, (c): “diagonal”.	24
2.3	The four mother 2D Haar-like wavelets used by Viola and Jones [79], the third wavelet is not present in Papageorgiou <i>et al.</i> ’s work [57]. . . .	25
2.4	A Haar-like wavelet with a specific size at a specific location.	26
2.5	The three types of mother 2D Haar wavelets used by Li <i>et al.</i> [44] The rectangles are of size $x \times y$ and are at distances of (dx, dy) apart. Each feature takes a value calculated by the weighted sum of the pixels in the rectangles.	27
2.6	Sparse feature and its granules used by Huang <i>et al.</i> [31].	28

3.1	A case where examples are highly skewed $\gamma_1 \gg 1$: Positives are squares, negatives are triangles. Suppose we train with four weak classifiers, which are linear separators. The first classifier is labeled as ‘1’. On the left is the result trained by Viola and Jones’ method: while the subsequent weak classifiers are well-modeled the first classifier is highly affected by γ_1 , resulting in rejecting a significant proportion of positives. On the right is the result trained by our method. Most of the positives are preserved.	35
3.2	Learning curves for Mushroom dataset	48
3.3	Some classification results from PETS2001 dataset1. White box: region predicted as pedestrian. Black box: region predicted as non-pedestrian. .	49
3.4	Performance on the PETS2001 datasets. Each example corresponds to a detected moving region. Left: PETS2001 dataset 1. Right: PETS2001 dataset 2.	49
3.5	Some detection results of our Online Asymmetric Boosting on the MIT+CMU test set.	50
3.6	ROC curves of the algorithms on the MIT+CMU set.	51
3.7	An example of training a boosted classifier with 10 weak classifiers, and with different algorithms.	52
4.1	Four types of Haar-like features used in [79]	56
4.2	Diagram of this method to fast-train a weak classifier. The notations are explained in subsequent sections.	58
4.3	Nineteen types of features used in our experiment	59
4.4	Comparison on the time to train a weak classifier	68
4.5	The ROC curves of three node classifiers with $M = 15$	70
4.6	The ROC curves of three node classifiers with $M = 200$	71

4.7	Histograms of the class-conditional feature value distributions presented to the best feature classifier when training weak classifier 1 using our method with $T = 295,920$	72
4.8	Histograms of the (weighted) class-conditional feature value distributions presented to the best feature classifier when training weak classifier 200 using our method with $T = 295,920$	73
4.9	Some results on the MIT+CMU test set	74
4.10	The ROC curves on the MIT+CMU test set of the three experimented cascades.	75
5.1	Illustration of how the operating point of a boosted classifier moves when more weak classifiers are trained. Blue solid curves are equivalent to the ROC curves of the boosted classifier. (a): A case when the symmetric goal G_1 is used, <i>i.e.</i> , $\lambda = 1$. (b): A case when an asymmetric goal G_λ is used with $\lambda = \alpha_0/\beta_0$	89
5.2	Comparison of training a boosted classifier with different λ and stopping bounds	95
5.3	A closer view on the comparison of training a boosted classifier with different bounds and λ 's, where $10 \leq \lambda \leq 100$	96
5.4	Comparing different methods on the MIT+CMU test set	99
5.5	ROC curves reported on the MIT+CMU test set by recent methods . . .	100
5.6	Some results of our method on the MIT+CMU test set	101

List of Algorithms

1	AdaBoost [67]	19
2	Offline Asymmetric Boosting	36
3	<i>Propagate_Weights</i> ($\hat{y}, y, v, \mathbf{v}_m$)	43
4	Online Asymmetric Boosting	47
5	Fast Training of a Weak Classifier	63
6	Multi-exit Asymmetric Boosting	86
7	<i>Train_Segment</i> ($m, \mathcal{H}, \mathcal{Z}$)	87

Abstract

Asymmetric boosting, while acknowledged to be important to imbalanced classification problems like face detection, is often based on the trial-and-error methodology to obtain the best boosted classifier, rather than on principled methods. This thesis solves a number of issues related to asymmetric boosting and the use of asymmetric boosting in face detection. It shows how a proper understanding and use of asymmetric boosting leads to improvement in the learning time, the learning capacity, the detection speed and the detection accuracy of a face detector.

First, an integrated framework for both online learning and asymmetric learning of a boosted classifier is presented. In addition, the proposed method adaptively balances the skewness of the weight distribution of the two classes presented to the weak classifiers, allowing them to be trained more equally. An additional constraint on propagating the weights of the data points is introduced, allowing the online learning to converge faster. When compared with the Online Boosting algorithm recently applied to object detection problems, a 0-10% increase in accuracy and 5-30% gain in learning speed were observed.

Second, training a face detector using boosting and Haar-like features often requires weeks of computation on a single CPU machine. The bottleneck is in the training of a weak classifier, currently in the order of minutes. Traditional techniques for training a weak classifier usually run in time $O(NT \log N)$, with N examples (approximately 10,000), and T Haar-like features (approximately 40,000). A method to train a weak

classifier in time $O(N + T)$ is presented, by using only the statistics of the weighted input data. Experimental results reveal a significantly reduced training time of a face detector from weeks to just a few hours. In particular, this method trades off a minimal increase in training time for a very large increase in the set of Haar-like features explored, enjoying a significant gain in accuracy.

Third, a generalized framework for representing a boosted classifier with multiple exit nodes is introduced. A method for training such a classifier is also proposed, which combines the recent idea of propagating scores across boosted classifiers and the use of asymmetric goals. A means for determining the ideal asymmetric goal is provided, which is theoretically justified under a conservative bound on the operating point target in the receiver-operator characteristic (ROC) curve, and is empirically near-optimal under the exact bound. Moreover, the method automatically minimizes the number of weak classifiers, avoiding the need to retrain a boosted classifier multiple times as in conventional methods. Experimental results show a significant reduction in the training time and the number of weak classifiers, as well as an improvement in accuracy.

Fourth, a set of bounds on the generalization ability of a boosted classifier trained with an asymmetric goal is proposed, as current generalization bounds are not designed for asymmetric errors. The proposed bounds show that, unlike traditional boosting methods where there is no difference between a margin of a positive example and that of a negative example, the penalties applied to the margins are different for different classes.

Finally, this thesis concludes with a discussion of future work for appearance-based face detection. These include potential advances in the learning of weak and boosted classifiers as well as in the online learning of face detectors.

Publications

Portions of the work described in this thesis have also appeared in:

Conference papers

- Minh-Tri Pham and Tat-Jen Cham. Online Learning Asymmetric Boosted Classifiers for Object Detection. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*, Minneapolis, MN, 2007.
- Minh-Tri Pham and Tat-Jen Cham. Fast Training and Selection of Haar features using Statistics in Boosting-based Face Detection. In *Proc. 11th IEEE International Conference on Computer Vision (ICCV'07)*, Rio de Janeiro, Brazil, 2007.
- Minh-Tri Pham, Viet-Dung D. Hoang, Tat-Jen Cham. Detection with Multi-exit Asymmetric Boosting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, Alaska, 2008.

The source code of the methods proposed in this thesis is available at:

- **PyCV - A Python package for Computer Vision**
<http://www.ntu.edu.sg/home5/pham0004/pycv/>

Nomenclature

The following terms are commonly used throughout the thesis.

Term	Description
<i>a.s.</i>	almost surely
<i>d.o.f.</i>	degree of freedom
<i>e.g.</i>	for example
<i>i.e.</i>	that is
<i>i.i.d.</i>	independently and identically distributed
<i>s.t.</i>	such that
<i>vs.</i>	versus
<i>w.r.t.</i>	with respect to
$\mathbb{P}[\cdot]$ (or $\hat{\mathbb{P}}[\cdot]$)	probability (or empirical probability) of an event
$\mathbb{P}_w[\cdot]$ (or $\hat{\mathbb{P}}_w[\cdot]$)	probability (or empirical probability) of an event <i>w.r.t.</i> probability measure $\mathbb{P}_w$ (or $\hat{\mathbb{P}}_w$) defined in the context
$\mathbb{E}[\cdot]$ (or $\hat{\mathbb{E}}[\cdot]$)	expectation (or empirical expectation) of a random variable or a function of random variables
$\mathbb{E}_w[\cdot]$ (or $\hat{\mathbb{E}}_w[\cdot]$)	expectation (or empirical expectation) of a random variable or a function of random variables <i>w.r.t.</i> probability measure $\mathbb{P}_w$ (or $\hat{\mathbb{P}}_w$)

Term	Description
DR	detection rate
FAR	false acceptance rate
FRR	false rejection rate
x	input point
$\mathcal{X}$	input space
y	output class
$\mathcal{Y}$	set of output classes; $\mathcal{Y} = \{-1, 1\}$ in this thesis
$h : \mathcal{X} \rightarrow \mathcal{Y}$	discrete-valued weak classifier
$\tilde{h} : \mathcal{X} \rightarrow \mathbb{R}$	real-valued weak classifier
$f : \mathcal{X} \rightarrow \mathbb{R}$	confidence (score) function of a boosted classifier, usually defined as $f(x) = \sum_i c_i h_i(x)$ or $f(x) = \sum_i \tilde{h}_i(x)$
$F = \text{sign}(f)$	classification function of a boosted classifier
$\mathcal{J}_\lambda(F)$ or $\mathcal{J}_\lambda(f)$	asymmetric error of a classifier

Chapter 1

Introduction

1.1 Motivation

Face detection is a computer vision task that identifies and determines the locations and sizes of human faces in arbitrary (digital) 2D images. It detects face locations (and possibly facial features) and ignores anything else, such as buildings, trees and bodies.

Figure 1.1: Face detection in action

Face detection has many applications in the real world. It serves as the first step

in many vision-based human computer interaction systems, to enable or bootstrap face recognition, face tracking, face pose estimation and facial expression recognition. Most existing methods on these topics assume human faces in an image or a video sequence has been identified and localized. To build a fully automated system that extracts information from images containing human faces, it is essential to use face detection as the first step to detect human faces before any subsequent task can proceed.

Face detection is also used in many surveillance systems. Some applications are used to identify and locate a human being in a scene, or to count the number people passing through a place in specified time intervals. Recently, face detection has been incorporated into digital cameras and movie cameras for subject-aware auto-focusing.

While a human being can do face detection effortlessly, this task is not easy at all for a machine. The challenges associated with face detection can be attributed to the following factors:

- **Pose.** The face images vary due to the pose of the objects (*e.g.*, human pose, distances among facial features).
- **View.** The images vary due to the relative camera-to-object view, and some parts like an eye or half a face may become partially or wholly occluded. In an image with a group of people, some faces may partially occlude other faces.
- **Presence or absence of structural components.** Features such as beards, mustaches, and glasses, may or may not be present and there is great variability among these components including shape, color, and size.
- **Imaging conditions.** When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of an object.

Early face-detection algorithms focus on the detection of frontal human faces. Examples include learning-based approaches of Sung and Poggio [62, 71, 72], Rowley *et*

Figure 1.2: Cascade of classifiers for face detection

al. [65], Osuna *et al.* [55], Schneiderman and Kanade [68] in which face detection is considered as a problem of binary classification of face and non-face patches. Newer algorithms attempt to solve the more general and difficult problem of multi-view face detection. That is, the detection of faces that are either rotated along the axis from the face to the observer (in-plane rotation), or rotated along the vertical or left-right axis (out-of-plane rotation), or both. Perhaps, the most successful early work of this kind was due to Yow and Cipolla [92, 93, 94, 95]. It was later followed by many researchers, including but not limited to Viola and Jones [35], Li *et al.* [44], Wu *et al.* [83, 85] and Huang *et al.* [32]. Surveys of face detection methods up to 2001 are available in [27, 91].

There are three factors in evaluating the performance of a face detector: detection rate, false acceptance rate, and detection speed. The detection rate of a face detector measures the probability that the detector truly recognizes a face patch as a face. Sometimes, the inverse of the detection rate, *i.e.*, the false rejection rate, which measures the probability that the detector wrongly classifies a face patch as a non-face, is used. The false acceptance rate measures the probability that the detector classifies a non-face patch as a face, while the detection speed measures how fast *in average* the face detector classifies a patch.

Many recent successful face detection algorithms follow the framework of Viola and Jones [79], which can detect faces in *real-time* with a high detection rate and an extremely low false acceptance rate. In this framework, face detection is implemented as a binary pattern-classification task. A large number of (usually square or rectangular)

sub-windows (or patches) of the image, located at different locations and scales, are examined. For each sub-window, its content is transformed into features. After which a classifier, trained on face and non-face examples, decides whether that sub-window contains is a face or not.

The key insight of achieving real-time detection speed in Viola and Jones' framework is the idea of designing the whole complex classifier as a cascade of classifiers with increasing complexity, illustrated in figure 1.2. An input patch is classified as a face if all the classifiers F_k classify it as a face. With this arrangement, simple non-face patches are rejected much faster by early and simple classifiers, allowing the cascade to classify a large number of patches in a small amount of time. Even though the coarse-to-fine idea has existed for several decades, and was widely used in the automatic target recognition techniques of the 1970s and 80s [6], the work of Viola and Jones was the first to apply it in face detection.

Despite the advantage of detection speed that a cascade can offer, there are a number of issues in learning it. It is not known beforehand how many classifiers are needed, nor which combination of ROC operating points (each defined by a detection rate and a false acceptance rate) produces an optimal cascade. Currently, these parameters are obtained mainly by trial and error, though some progress has been made [9, 70].

Viola and Jones [79, 80] and conventional methods often use AdaBoost or one of its variants [21, 67] to train F_k as a *boosted classifier*¹. However, one has to devise a learning strategy to find an optimal trade-off among three important factors of a boosted classifier: detection rate, false acceptance rate, and number of weak classifiers. To maintain a high detection rate and an extremely low false acceptance rate for the overall cascade, each individual classifier F_k must ensure an extremely small false rejection rate and a moderate false acceptance rate. On the one hand, to achieve both of these criteria

¹Sometimes, F_k is also called as a *voting classifier* because it is a linear combination of weak classifiers.

concurrently, one needs to train as many weak classifiers as possible. On the other hand, it is also essential to minimize the number of weak classifiers of a boosted classifier, as it is roughly proportional to the running time of the classifier.

Another problem with boosting techniques like AdaBoost comes from their learning objective function, which is to minimize an upper bound of the (empirical) classification error of the trained boosted classifier. In these methods, there is no difference between penalizing a false positive and penalizing a false negative. However, when constructing a cascade, a false negative is a lot more severe than a false positive². Unfortunately, even if the boosted classifier is trained with an infinite number of weak classifiers, there is no guarantee that the final boosted classifier satisfies a desired pair of false acceptance rate and false rejection rate. In fact, if the two positive and negative classes are *not separable*, *not every pair of desired error rates* can be achieved³.

In [79], Viola and Jones fixed the problem of obtaining a classifier favoring one type of error rate over the other by introducing a threshold at the end of a boosted classifier. As the threshold decreases, the boosted classifier accepts more false positives and fewer false negatives. While this sounds like the problem is solved, the approach raises a number of questions:

1. Is it possible to reach the desired rates in the first place?
2. If it is possible, how many weak classifiers are to be trained to ensure that a threshold allowing the boosted classifier to achieve the desired rates exists?
3. Given the previous question answered, are these weak classifiers *optimal* for the boosted classifier?

In their follow-up publication [80], Viola and Jones introduced a notion called *asymmetric boosting*, and showed that their new algorithm outperforms their previous ap-

²Because while a non-face patch passes a classifier, it can still be rejected by subsequent classifiers, but once a face patch is rejected it cannot be recovered later.

³Chapter 5 discusses this in more detail.

proach. The main idea of asymmetric boosting is the introduction of a new objective function. Instead of finding a classifier F_k that minimizes the empirical classification error

$$\hat{\mathbb{P}}[F_k(x) \neq y] \quad (1.1)$$

as in traditional methods⁴, asymmetric boosting finds a classifier minimizing a term called *asymmetric error*:

$$\hat{\mathbb{P}}[F_k(x) = 1|y = -1] + \lambda \hat{\mathbb{P}}[F_k(x) = -1|y = 1], \quad (1.2)$$

where λ is a parameter, often referred to as *the degree of asymmetry*, that controls the tradeoff between the two error rates. Here, we refer to the goal of minimizing an asymmetric error as *an asymmetric goal*. If we increase λ , the resulting false rejection rate tends to decrease and the resulting false acceptance rate tends to increase. Conversely, if we decrease λ , the resulting false rejection rate tends to increase and the resulting false acceptance rate tends to decrease. Because the resulting classifier favors one type of error over the other, it is also referred to as an *asymmetric classifier*. Viola and Jones also showed that, by using asymmetric boosting, the resulting number of weak classifiers are much fewer than using the original AdaBoost method. A few papers express asymmetric boosting in different forms, including [18, 29, 49], but the main idea is the same.

Empirical results in many papers (*e.g.*, [29, 49, 80]) show that asymmetric boosting is a better approach than the former in obtaining an boosted classifier having biased error rates. However, like the former, it raises a number of questions that are not addressed by current methods:

1. What is the choice of λ to train a boosted classifier? The current solution that existing work uses is an adhoc fashion. Multiple boosted classifiers are trained with different values of λ and number of weak classifiers. The best classifier is

⁴ $\hat{\mathbb{P}}$ denotes the discrete distribution representing the training set, defined in (2.5).

selected manually, and is usually based on the whether it matches a desired pair of error rates and whether the number of weak classifiers needed is minimal.

2. Is there a relationship between the final error rates and the degree of asymmetry λ given priorly?
3. Is there a way to select λ before training?
4. How well does the resulting asymmetric boosted classifier generalize?

It is therefore, the theme of this thesis to further enhance our knowledge about asymmetric boosting, and to give reasonable answers to *all* of the questions above. We show how asymmetric boosting, when applied to the problem of face detection, improves the accuracy, the training time, and the detection speed of a face detector.

1.2 Topics and Approaches of this Thesis

The following problems are addressed in this thesis.

1.2.1 Online Learning of An Asymmetric Boosted Classifier

Recently, there has been an increased interest in applying boosting techniques on computer vision problems that require the *online* learning of a boosted classifier [34, 24, 64]. To do this, these methods use the same underlying Online Boosting algorithm proposed by Oza [56], which in turns minimizes the classification error while updating the weak classifiers online.

However, in the tasks of object detection and recognition (e.g., face detection or database retrieval), one is often faced with a binary classification problem where the probability of observing a positive example (*i.e.*, the target object) is much lower than the probability of observing a negative example (*e.g.*, background, non-target object).

In such domains, an asymmetric classifier that can avoid the danger of missing positive patterns is often more desirable than a classifier learned by minimizing the classification error. Methods addressing the problem of learning asymmetric classifiers have been proposed in [18, 29, 49, 80]. To our best knowledge, these methods cannot be done online.

In chapter 3, we introduce a novel boosting algorithm which addresses the problem of learning *online* an *asymmetric* boosted classifier. The idea of learning *online* is similar to the Online Boosting algorithm [56] having been recently used [24, 34, 64], but with a faster convergence speed. On the aspect of learning *asymmetrically*, our method is similar to that of Viola and Jones [80]. Besides, our method also seeks to balance the *skewness* of the weight distribution of the two classes presented to each weak classifiers (to be discussed in the chapter), so that they are trained more equally.

1.2.2 Fast training of Haar-like features using Statistics

One of the greatest obstacles to a wider use of cascades is that it takes a long time to train a cascade, even if all parameters are properly assigned a priori. For example, it took weeks of computation to produce the final cascade in [79], using multiple machines. The bottleneck is at the training of a weak classifier⁵, of which the time complexity is $O(NT \log N)$, where N is the number of examples used and T is the size of the feature set. In practice, one has to run many trials and to choose the best configuration, which results in an even longer training time.

In chapter 4, we present a method to train weak classifiers with a speed much faster than traditional methods, bringing down the total training time of a face detector from weeks to *just a few hours*. Our approach takes a different point of view from traditional methods. Noticing that both N and T are not only dominant, but also important factors in constructing the weak classifiers, rather than trying to reduce either N or T as

⁵usually in minutes practically

Figure 1.3: Nineteen types of Haar-like features used in our experiment

in traditional methods, we *eliminate the multiplication* of N and T incurred in the time complexity in training a weak classifier by using *statistics*. This reduces the time complexity to $O(N+T)$. In our approach, the time to train a weak classifier is approximately linear to either N or T , whichever is more dominant, *but not both*. By doing so, we can train weak classifiers with more examples – giving better generalization, as well as more features – giving a wider hypothesis set for the weak classifiers.

Experimental results show that with our new method, we can train up to 19 Haar-like feature types (figure 1.3) on a 24×24 image sub-window, generating in total 295,920 features, six times the number of features used by Viola and Jones in [81]. Yet, it takes 6 seconds to train all feature classifiers and select the best one on a conventional Intel Pentium IV PC.

1.2.3 Choosing an Asymmetric Goal

In chapter 5, we introduce a generalized representation for a boosted classifier with multiple exit nodes, and propose a method to training which combines the idea of propagating scores across boosted classifiers [87, 89] and the use of asymmetric goals [80]. Concurrently, we analyze the relationship between the choice of an asymmetric goal and the resulting pair of false acceptance rate and false rejection rate of the boosted classifier obtained from training using this goal as the objective. This relationship allows us to select an asymmetric goal prior to training, such that after the training, the number of weak classifiers is minimized while at the same time the target error rates are reached. This avoids the need to retrain a boosted classifier multiple times for the empirically best performance as in conventional methods. Furthermore, we explain how the weak classifiers of many existing methods are trained sub-optimally. The results show significant reduction in training time and in the number of weak classifiers having been trained, as well as better accuracy, compared to conventional cascades and multi-exit boosted classifiers.

1.2.4 Bounding a True Asymmetric Error from its corresponding Empirical Asymmetric Error

In traditional classification learning methods, there is no difference between a false negative and a false positive, *i.e.*, they are equally penalized. The idea of using this asymmetric goal is to weigh false negatives and false positives differently so as to favor one type of error rate over the other. Traditional bounds for boosting [36, 37, 67] explain the relationship between the *empirical* classification error the true generalization error of a classifier. They are not designed to explain the performance of an asymmetric classifier because they do not take into account the asymmetry of the goal. Conversely, in an attempt to explain the performance of an asymmetric classifier, we propose in chapter 6

the first bounds on the true asymmetric error based on the empirical asymmetric error of a classifier, which have not been addressed before in the literature.

1.3 Thesis Outline

The subsequent chapters of the thesis are organized as follows. Chapter 2 gives an overview of recent real-time appearance-based face detection methods, and explains common concepts about cascades and boosting used in these methods.

In chapter 3, an integration method of asymmetric boosting and online boosting is covered. Furthermore, the convergence rate of the method is improved via the enforcement of an extra converging condition. The weak classifiers are trained more equally by balancing the skewness of the prior probabilities of the joint weight distribution, leading to slightly better accuracy.

In chapter 4, a fast method to train the weak classifiers of a face detector is proposed. The key idea of the method is based using the *statistics* of the Haar-like features, so that the training time of a feature is independent of the training size. This reduces the time complexity in training a weak classifier by an order of magnitude.

The analysis of the relationship between an asymmetric goal and the output error rates of a boosted classifier learned by using this goal is presented in chapter 5. Also in this chapter, a general framework capturing traditional and recent cascades, as well as a new kind of boosted classifier called *multi-exit boosted classifier* is introduced. It is shown in this chapter how multi-exit boosted classifier outperforms traditional cascades, as well as how a number of existing methods train their weak classifiers sub-optimally. The knowledge obtained from the analysis allows us to select an asymmetric goal prior to training, such that after the training, the number of weak classifiers is minimized while at the same time reaching the target error rates. This avoids the need to retrain a boosted classifier multiple times for the empirically best performance as in conventional

methods.

We analyze and propose a set of generalization bounds for true asymmetric error in chapter 6. Due to the novelty of asymmetric goals, these kinds of bounds have not been addressed before in the literature.

The contributions described in chapter 3, 4, and 5 have been presented in [60], [59], and [61] respectively. The contribution in chapter 6 has been submitted for publication and is under review.

Chapter 2

Overview of Appearance-based Face Detection

In this chapter, we give an overview of face detection methods, especially those using boosting and cascades to give real-time detection speed. Face detection is a necessary first-step in many face recognition systems, with the purpose of localizing and extracting the face region from the background. It also has several applications in areas such as content-based image retrieval, video coding, video conferencing, crowd surveillance, and intelligent human-computer interfaces. However, it was not until recently that the face detection problem received considerable attention among researchers. A wide variety of techniques have been proposed, ranging from simple edge-based algorithms to composite high-level approaches which utilize advanced pattern recognition methods.

While numerous methods have been proposed to detect faces in a single image of intensity or color images, we are unaware of any survey on this particular real-time face detection topic. Surveys of face detection methods before 2001 were written by Hjelmas and Low [27] and Yang *et al.* [91]. None of these methods could run in real-time until the seminal work of Viola and Jones in 2001 [79].

In general, early techniques to detect faces from a single intensity or color image can

be classified into four categories, with some methods clearly overlap category boundaries. More details can be found in the respected surveys [22, 27, 91]:

1. **Knowledge-based methods.** These rule-based methods encode human knowledge of what constitutes a face. Usually, the rules capture the relationships among parts of a face [90].
2. **Constellation-based approaches.** These algorithms aim to find structural features of the face (*e.g.*, eyes, mouths, and noses), and then use these to locate the faces [13, 42, 92, 93, 94, 95].
3. **Template-matching methods.** Several standard patterns of a face are stored to describe the object as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection [12, 40].
4. **Appearance-based methods.** In contrast to template-matching, the models (or templates) are learned from a set of training images which should capture the representative variability of face appearance. A sub-window is scanned over the probed image with different locations and scales. Each obtained patch is classified using these learned models to determine if it corresponds to an object [11, 43, 55, 62, 63, 65, 68, 71, 72].

Among the face detection methods, appearance-based methods using learning algorithms have attracted much attention recently and have demonstrated excellent results.

Instead of designing a single complex classifier and applying it to every possible patch, recently coarse-to-fine search has been used to achieve computational efficiency. The coarse-to-fine idea has existed for several decades, and, in particular, was used widely in the automatic target recognition techniques of the 1970s and 80s [6]. It was arguably first applied to object detection in the work of Amit and Geman [2]. In 2001,

the idea was popularized by Viola and Jones’s real-time object detection framework [79]. In this framework, which we name as the VJ framework in what follows, a cascade of classifiers with increasing complexity is built, as illustrated in level 1 in figure 2.1. An input patch is classified as a face only if all the classifiers classify it as a face. Most non-face patches are quickly rejected by the early classifiers.

The VJ framework demonstrated impressive performance. It was the first method that achieved real-time detection speed and high accuracy comparable to that of previous state of the art methods. Many subsequent papers have shown that the VJ framework is the key to achieving both real-time detecting speed and high accuracy. In fact, in recent years, there has been almost no work that offers real-time detection speed and high accuracy without being related to this framework.

The VJ framework consists of three levels, as shown in figure 2.1. Level 1 is a cascade of node classifiers. In level 2, each node classifier is a boosted classifier, which linearly combines a number of weak classifiers, obtained from training using AdaBoost or one of its variants. In level 3, each weak classifier corresponds to a simple rectangular Haar-like feature that can be extracted extremely fast, in microseconds. Despite its success, the VJ framework imposes a number of challenging learning issues in all the three levels, attracting a great amount of research contributions in the recent years.

For easy understanding, we now review recent work related to all the three levels of the VJ framework. We assume all face sub-windows define the *positive* class and all non-face sub-windows define the *negative* class.

2.1 Cascade Design

Basically, a cascade is a sequence of boosted classifiers arranged from simple classifiers to complex ones. The complexity of a boosted classifier is measured by the number of weak classifiers it contains. The harder the classification sub-problem is, the more weak

Figure 2.1: Three levels of the VJ framework.

classifiers are needed, and at the same time the longer it will take to evaluate or to predict the class of an input point¹.

Mathematically, let x be the input point representing the input image sub-window, let $F_k(x)$ represent the k -th boosted classifier in a cascade (where $k \in \{1, \dots, Q\}$), and let $C(x)$ represent the cascade itself. The cascade $C(x)$ classifies x by:

$$C(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } F_k(x) = 1 \quad \forall k \in \{1, \dots, Q\} \\ -1 & \text{otherwise} \end{cases} \quad (2.1)$$

In practice, the evaluation of all $F_k(x)$ for $k \in \{1, \dots, Q\}$ is done in sequence. The

¹More details are explained in the next sections.

evaluation stops as soon as the first $F_k(x)$ returns -1, or all $F_k(x)$ return 1. As a result, the *average* time to reject an example is much smaller than the time to accept an example as face. However, since face examples are much rarer than non-face examples, the overall evaluation time is small.

Despite the cascade’s efficiency in detection, it imposes a number of issues in learning. It is not known beforehand how many boosted classifiers are needed. The overall detection rate of a cascade is the product of detection rates associated with all individual boosted classifiers in the cascade; similarly, the overall false acceptance rate is the product of all classifiers’ false acceptance rates. It is also not known which combination of ROC operating points (each defined by a detection rate and a false acceptance rate) produces an optimal cascade.

Currently, the common practice to go around this problem is via trial and error. At each stage of the training of a cascade, multiple boosted classifiers with different parameters are trained to solve the same classification sub-problem. Then, the boosted classifier that the human trainer sees the most fitted is manually selected and inserted to the cascade.

Some progress towards automatic training of a cascade has been made by Sun *et al.* [70] and Brubaker *et al.* [9]. In these approaches, the key assumption² is: it is possible to achieve subsequent boosted classifiers with *the same fixed resources* that perform *the same as* the current boosted classifier. With this assumption, they could predict the outcome of the cascade without having to train subsequent boosted classifiers, and estimated the key parameters for training the current boosted classifiers: the desired detection rate, the desired false acceptance rate, and the number of weak classifiers. However, in practice, this assumption does not strongly hold because subsequent classifiers are harder to train due to increases in difficulty of the subsequent sub-problems [9]. Hence, only limited improvement was observed.

²It is called the “repeatability assumption” in these papers.

Instead of training many boosted classifiers for a cascade, which requires a huge amount of time, Bourdev and Brandt [8] proposed a method that trains a single and very long boosted classifier using a modified version of AdaBoost that is biased towards favoring the positive class; and subsequently “calibrates” the classifier so that it becomes a cascade, by incorporating a threshold in *every* weak classifier. However, because the thresholds are obtained *after* all the weak classifiers are trained, the weak classifiers become sub-optimal *w.r.t.* the learning goal of the cascade. Xiao *et al.* [88] corrected this by updating the training set *before* training each weak classifier. Sochman and Matas[69] proposed a similar idea. But instead of introducing one threshold per weak classifier, they introduced two: one to classify the input as positive immediately, and the other to classify the input as negative immediately. They used Wald’s sequential probability ratio test [82] and a likelihood ratio approximation to estimate the thresholds.

Still, two issues are raised and not addressed in the work following this direction. First, bootstrapping³ is required at every weak classifier, incurring a significant extra computational cost. Second and more importantly, the requirement for early rejection diminishes as progress is made down the cascade, where the problem becomes dominated by accuracy (as classification becomes harder) as opposed to speed (since most of the obvious negative samples have already been rejected earlier). In such cases, having a decision made at each weak classifier effectively discards important information that may have been exploited if the decisions were postponed until further downstream, leading to less accurate classification.

Xiao *et al.* [89] and Wu *et al.* [87] independently noticed that the scores obtained from the previous boosted classifier may be exploited downstream, and proposed to propagate the scores from one boosted classifier to the next. When this is done, a cascade may be considered as a single boosted classifier with multiple exits. Results show

³Bootstrapping is a process of preparing a set of training examples that have been *positively predicted* by all previously trained boosted classifiers in the cascade before training a boosted classifier. This requirement is a must to ensure that the data set well-represents the current classification problem.

Algorithm 1 AdaBoost [67]

Require: Training set $\{(x_n, y_n)_{n=1}^N\}$.
 Start with weights $w_n = 1/N, n = 1, 2, \dots, N$.
for $m = 1$ to M **do**
 Normalize w_n so that $\sum_{n=1}^N w_n = 1$.
 Train weak classifier $h_m(x) \in \{-1, 1\}$ using data $\{(x_n, y_n, w_n)_{n=1}^N\}$, where w_n is the weight of example (x_n, y_n) .
 Set $c_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$, where $e_m = \sum_{n=1}^N w_n 1_{[y_n \neq h_m(x_n)]}$.
 Set $w_n = w_n \exp(-y_n c_m h_m(x_n)), n = 1, 2, \dots, N$.
end for
return the classifier $F(x) = \text{sign}(\sum_{m=1}^M c_m h_m(x))$.

a significant reduction in the number of weak classifiers. Further investigation about this direction is discussed in chapter 5.

2.2 Learning a Boosted Classifier

To train a boosted classifier, existing methods use a popular machine-learning algorithm called AdaBoost [20, 67] or one of its variants. AdaBoost produces a strong classifier by linearly combining a number of weak classifiers. The weak classifiers are allowed to be so *weak* that their classification errors can be close to 0.5, while the resulting strong classifier has a very low classification error. The pseudocode of AdaBoost is illustrated in algorithm 1.

The strong boosted classifier is a linear combination of M weak classifiers:

$$F(x) \stackrel{\text{def}}{=} \text{sign}(f_M(x)) \quad (2.2)$$

$$f_M(x) \stackrel{\text{def}}{=} \sum_{i=1}^M c_i h_i(x) \quad (2.3)$$

where c_i is the voting coefficient associated with weak classifier $h_i : \mathcal{X} \rightarrow \mathcal{Y}$.

Friedman *et al.* [21] showed that, at any stage of AdaBoost, a new weak classifier $h_m(x)$ is learned by minimizing an exponential upperbound of the classification error of

the boosted classifier:

$$(c_m, h_m(x)) = \arg \min_{c, h(x)} \hat{\mathbb{E}} \left[\exp \left(ch(x) + \sum_{i=1}^{m-1} c_i h_i(x) \right) \right], \quad (2.4)$$

where $\hat{\mathbb{E}}$ denotes the expectation *w.r.t.* the empirical distribution $\hat{\mathbb{P}}$ defined by:

$$\hat{\mathbb{P}} = \frac{1}{N} \sum_{i=1}^N \delta_{x_i, y_i}, \quad (2.5)$$

and $\delta_{x_i, y_i}(\cdot)$ is a probability distribution concentrated at point (x_i, y_i) .

Because the weak classifiers $h_m(x)$ produce output in $\mathcal{Y}$, AdaBoost is often referred to as a *discrete-valued* boosting method. In [21], Friedman *et al.* proposed a number of *real-valued* boosting methods in which the weak classifiers are replaced by real-valued ones. Their boosted classifiers are of the form

$$f_M(x) \stackrel{\text{def}}{=} \sum_{i=1}^M \tilde{h}(x), \quad (2.6)$$

where $\tilde{h} : \mathcal{X} \rightarrow \mathbb{R}$.

Schapire *et al.* [66, 67] provided some theory to support their discrete-valued algorithms, in the form of upper bounds on generalization error. This theory had evolved in the computational learning community, initially based on the concepts of PAC learning. Koltchinskii *et al.* [36, 39, 38, 37] followed up this line of research by tightening up these bounds significantly. However, these bounds are only applicable to discrete-valued boosting methods. At the moment, no bounds for real-valued boosting methods have been proposed.

One problem is that the learning objective function discussed in these papers does not differentiate between a false positive and a false negative. On the contrary, in face detection using a cascade, a false negative plays a much more important role than a false positive in face detection. This is because if a non-face sub-window passes a classifier, it

can still be rejected by subsequent classifiers, while once a face sub-window is rejected it cannot be recovered later.

At first, Viola and Jones proposed to fix the issue by introducing a threshold at the end of a boosted classifier [79]. As discussed in section 1.1, this raised a number of unanswered questions. More importantly, in chapter 5.3, we show that the use of a non-zero threshold is a clear indication that the weak classifiers are trained *sub-optimally w.r.t.* the learning goal of a boosted classifier, implying that the boosted classifiers themselves are not trained optimally.

Later, Viola and Jones [80] and follow-up researchers [18, 29, 49] introduced new learning objective functions that penalize a false negative much more than a false positive (hence the notion *asymmetric boosting*). Despite better empirical results over previous approaches, the main concern with these methods is how to choose properly an asymmetric goal before training, which was not clearly understood. We show in chapter 5 how an asymmetric goal can be chosen prior to training such that the number of weak classifiers is approximately minimized, when the desired pair of error rates are reached for the boosted classifier. Using our proposed asymmetric goal, we avoid the common practice, in existing methods, of training multiple boosted classifiers with different asymmetric goals to select the best one, which consumes a huge amount of training time.

Notice that the output range of the asymmetric error in (1.2) is interval $[0, \lambda + 1]$. As λ approaches either 0 or ∞ , the interval becomes too small or too large, introducing numerical instability. Hence, throughout this thesis, we consider the following asymmetric error, which is a normalized version of (1.2), defined by:

$$\mathcal{J}_\lambda(F) \stackrel{\text{def}}{=} \frac{1}{\lambda + 1} \mathbb{P}[F(x) = 1 | y = -1] + \frac{\lambda}{\lambda + 1} \mathbb{P}[F(x) = -1 | y = 1] \quad (2.7)$$

for a true asymmetric error (*i.e.*, asymmetric error with respect to the true but unknown

distribution $\mathbb{P}$) and:

$$\hat{\mathcal{J}}_\lambda(F) \stackrel{\text{def}}{=} \frac{1}{\lambda+1} \hat{\mathbb{P}}[F(x) = 1|y = -1] + \frac{\lambda}{\lambda+1} \hat{\mathbb{P}}[F(x) = -1|y = 1] \quad (2.8)$$

for an empirical asymmetric error (*i.e.*, asymmetric error with respect to the training set, represented by distribution $\hat{\mathbb{P}}$). A learning method using the empirical asymmetric error in (2.8) produces exactly the same boosted classifier as the one produced by the method using the asymmetric error in (1.2), but the ranges of $\mathcal{J}_\lambda(F)$ and $\hat{\mathcal{J}}_\lambda(F)$ are always fixed at interval $[0, 1]$ for all $\lambda \geq 0$.

The error functions to be minimized in boosting techniques are often with respect to the training distribution $\hat{\mathbb{P}}$ in (2.5). While Schapire *et al.* and Koltchinskii *et al.* pioneered in bounding the true generalization error of a boosted classifier based on the margin distribution of the examples. Their work cannot be applied to asymmetric boosting because they do not take into account the asymmetry of the goal. We propose the first set of bounds on true asymmetric error which are based on the *class-conditional* margin distributions of the examples in chapter 6.

2.3 Learning a Weak Classifier

A weak classifier $h(x)$ is learned from the original training set $\{(x_i, y_i)_{i=1}^N\}$ associated with a discrete weight distribution $(w_1, \dots, w_N)$. More precisely, $h(x)$ is obtained from:

$$h \stackrel{\text{def}}{=} \arg \min_{h'} \hat{\mathbb{E}}_w [h'(x) \neq y], \quad (2.9)$$

where $\hat{\mathbb{E}}_w$ is the expectation with respect to a weighed empirical distribution $\hat{\mathbb{P}}_w$ given by:

$$\hat{\mathbb{P}}_w = \left(\sum_{i=1}^N w_i \delta_{x_i, y_i} \right) / \left(\sum_{i=1}^N w_i \right). \quad (2.10)$$

In face detection, a weak classifier is often selected from a discrete set of feature classifiers. That is, $h \in \mathcal{H}$ with $|\mathcal{H}| < \infty$. Each feature classifier is associated with a feature, which is a piece of information extracted from the input patch. Here, x is equivalent to a matrix $\mathbf{I}$ representing the input patch. The common sizes of $\mathbf{I}$ are 19×19 , 20×20 , or 24×24 .

A feature classifier often contains a vector of parameters θ . We can replace $h(x)$ with $h(x; \theta)$ to emphasize that feature classifier depends on the parameters. By tuning these parameters, better accuracy can be achieved. The best vector θ is typically chosen as the one minimizing the same goal function as in (2.9):

$$\hat{\theta} = \arg \min_{\theta} \hat{\mathbb{E}}_w [h(x; \theta) \neq y]. \quad (2.11)$$

Sometimes, to simplify the analysis, the set of values of θ and the set of features are grouped together into a single set. In this case, $\mathcal{H}$ can be visualized as a discrete set of continuous spaces.

Two factors in designing a feature include:

1. Fast extraction. The *average* number of features used for evaluating a sub-window is proportional to the evaluation time of the sub-window. Therefore, the faster a feature can be evaluated the faster the detection runs.
2. Discrimination. The more discriminative between face and non-face, the better the system's accuracy.

A number of features have been proposed in the literature, including pixel-based features [1, 3], Haar-like features [31, 44, 46, 54, 59, 79], histograms of oriented gradients [14], shape-based features [84], and geometrically stable features [47, 53]. Among these features, Haar-like features have been receiving the most attention due to their fast feature evaluation time with a reasonable discrimination ability. Throughout this thesis,

Figure 2.2: The mother 2D Haar wavelets used by Papageorgiou *et al.* [57]; (a): “vertical”, (b): “horizontal”, (c): “diagonal”.

Haar-like features are used for training the weak classifiers of face detectors. We now give a review on Haar-like features.

2.3.1 Haar-like Features

Haar-like wavelets were first introduced in face detection by Papageorgiou *et al.* [57]. In their work, an arbitrary patch of size 19×19 is represented by an over-complete set of 2D Haar-like wavelet basis functions. The set is generated by shifting and dilating three mother 2D wavelets, as shown in figure 2.2. However, a subset of the over-complete set is learned to derive a compact representation of the object class, which is then used as an input to a support vector machine classifier to determine if the patch corresponds to the object class.

While Papageorgiou *et al.* used the wavelets for representation, Viola and Jones [79] proposed to use them for classification. In their work, the result of integrating a wavelet with a patch is called a *Haar-like feature*. A weak classifier is created and associated with the feature by thresholding it with a properly trained threshold. AdaBoost [20] is used as an efficient scheme for selecting these features, and is discussed later in section 2.2. Beside introducing the fourth mother wavelet, as shown in figure 2.3, Viola and Jones also proposed the use of integral image for *extremely* fast feature evaluation.

Given an image I , the integral image $\tilde{I}$ at location (p, q) (row q , column p) contains

Figure 2.3: The four mother 2D Haar-like wavelets used by Viola and Jones [79], the third wavelet is not present in Papageorgiou *et al.*'s work [57].

the sum of the pixel intensities above and to the left of (p, q) , inclusively:

$$\ddot{\mathbf{I}}_{q,p} = \sum_{p' \leq p} \sum_{q' \leq q} \mathbf{I}_{q',p'}. \quad (2.12)$$

By using the integral image, the sum of pixel intensities of a rectangle (x, y, w, h) (top left corner (x, y) with w pixels per row and h pixels per column), denoted as $\pi(x, y, w, h)$, can be computed using at most *four* references to the integral image, independent of its location or size:

$$\begin{aligned} \pi(x, y, w, h) &\stackrel{\text{def}}{=} \sum_{x \leq p \leq x+w} \sum_{y \leq q \leq y+h} \mathbf{I}_{q,p} \\ &= \ddot{\mathbf{I}}_{y-1, x-1} + \ddot{\mathbf{I}}_{y+h-1, x+w-1} - \ddot{\mathbf{I}}_{y+h-1, x-1} - \ddot{\mathbf{I}}_{y-1, x+w-1}. \end{aligned} \quad (2.13)$$

For example, consider the fourth Haar-like wavelet residing on top of rectangle $(8, 12, 8, 4)$ in figure 2.4. Suppose the grey area is untouched, the Haar-like feature v is the sum of pixel intensities in white rectangles minus the sum of pixel intensities in

Figure 2.4: A Haar-like wavelet with a specific size at a specific location.

black rectangles. The value of v can be computed by:

$$\begin{aligned}
 v &\stackrel{\text{def}}{=} -\pi(8, 12, 4, 2) + \pi(12, 12, 4, 2) + \pi(8, 14, 4, 2) - \pi(12, 14, 4, 2) \\
 &= -\left(\ddot{\mathbf{I}}_{11,7} + \ddot{\mathbf{I}}_{13,11} - \ddot{\mathbf{I}}_{13,7} - \ddot{\mathbf{I}}_{11,11}\right) \\
 &\quad + \left(\ddot{\mathbf{I}}_{11,11} + \ddot{\mathbf{I}}_{13,15} - \ddot{\mathbf{I}}_{13,11} - \ddot{\mathbf{I}}_{11,15}\right) \\
 &\quad + \left(\ddot{\mathbf{I}}_{13,7} + \ddot{\mathbf{I}}_{15,11} - \ddot{\mathbf{I}}_{15,7} - \ddot{\mathbf{I}}_{13,11}\right) \\
 &\quad - \left(\ddot{\mathbf{I}}_{13,11} + \ddot{\mathbf{I}}_{15,15} - \ddot{\mathbf{I}}_{15,11} - \ddot{\mathbf{I}}_{13,15}\right) \\
 &= -\left(\ddot{\mathbf{I}}_{11,7} + \ddot{\mathbf{I}}_{11,15} + \ddot{\mathbf{I}}_{15,7} + \ddot{\mathbf{I}}_{15,15}\right) \\
 &\quad + 2\left(\ddot{\mathbf{I}}_{11,11} + \ddot{\mathbf{I}}_{13,11} + \ddot{\mathbf{I}}_{13,15} + \ddot{\mathbf{I}}_{15,11}\right) \\
 &\quad - 4\ddot{\mathbf{I}}_{13,11}
 \end{aligned} \tag{2.14}$$

Since only nine elements of $\ddot{\mathbf{I}}$ are needed, v is evaluated in just a few hundred CPU clock cycles. Note that we assume $\ddot{\mathbf{I}}_{q,p} = 0$ if either $p < 0$ or $q < 0$.

Due to their fast evaluation speed, Haar-like features has attracted a lot of attention

Figure 2.5: The three types of mother 2D Haar wavelets used by Li *et al.* [44] The rectangles are of size $x \times y$ and are at distances of (dx, dy) apart. Each feature takes a value calculated by the weighted sum of the pixels in the rectangles.

from scientists and researchers. A natural extension to Haar-like features is to introduce more mother wavelets to the feature set. This was adopted in the work of Li *et al.* [44]. Three basic types of features were introduced as shown in figure 2.5.

Huang *et al.* [31] further extended Haar-like features in a slightly different way. Instead of using rectangles, they proposed sparse granular features, as depicted in figure 2.6, of the form:

$$F(\pi) = \sum_i \alpha_i \pi(x, y, s), \quad \alpha_i \in \{-1, +1\} \quad (2.15)$$

where $\pi(x, y, s)$ represents a sum of pixel intensities in a square (with top left corner (x, y) and length s) on the patch, and α_i is its coefficient.

Another direction of generalizing Haar-like feature is to in-plane rotate them with different angles. The work of Lienhart and Maydt [46] pioneered this by showing how 45-degree rotated Haar-like features can be dealt with by introducing 45-degree rotated integral image. It was later followed by the work of Du *et al.* [16] in which Haar-like features are rotated by $\pm \arctan(0.5)$ degrees.

Figure 2.6: Sparse feature and its granules used by Huang *et al.* [31].

2.3.2 Haar-like Feature Classifier

One can think of a Haar-like feature as a *linear projection* of matrix $\mathbf{I}$ down to $\mathbb{R}$. Learning a Haar-like feature classifier is, therefore, equivalent to finding the best parameters minimizing the weighted empirical classification error similar to (2.11), but with input x being replaced by Haar-like feature v . That is:

$$\hat{\theta} = \arg \min_{\theta} \hat{\mathbb{E}}_w [h(v; \theta) \neq y]. \quad (2.16)$$

Once the parameters for every Haar-like feature classifiers have been obtained the next step is to select feature classifier minimizing (2.9). Usually, because the set of Haar-like features are large, the process of training a single weak classifier often takes about a few minutes. In chapter 4, we introduce a technique to dramatically reduce the training time of a weak classifier from minutes down to *just a few seconds*, while increasing the size of the feature set to at least *six* times. The key idea to our approach is the use of *statistics* to train feature classifiers.

Chapter 3

Online Asymmetric Boosting

3.1 Introduction

AdaBoost [20, 67] has been successfully used in many machine learning and pattern recognition tasks in computer vision. Its underlying idea is to combine an ensemble of M weak learners to produce the final classifier with very high accuracy. In the tasks of object detection and recognition, especially for face detection, impressive results [30, 44, 79, 80] have been reported when cascading or hierarchically organizing such boosted classifiers from coarse to fine, achieving a final classifier with a high detection rate and fast detection speed.

Recently, there has been considerable interest in applying boosting techniques on computer vision problems that require online learning. Examples are: online selection of discriminative features of Grabner and Bischof [24], online conservative learning of Roth *et al.* [64], and online co-training of Javed *et al.* [34]. These methods use the same underlying Online Boosting algorithm proposed by Oza [56] to learn online. The algorithm minimizes the classification error while updating the weak classifiers online.

However, in the tasks of object detection and recognition (e.g., face detection or database retrieval), one often encounters a binary classification problem where the prob-

ability of observing a positive example (*i.e.*, the target object) is much lower than the probability of observing a negative example (*e.g.*, background, non-target object). In such domains, an asymmetric classifier that can avoid the danger of missing positive patterns is often more desirable than the one minimizing the classification error. High detection rates can be achieved by cascading the classifiers (*e.g.*, [79, 80]) or hierarchically organizing them (*e.g.*, [30, 44]) from coarse to fine.

There have been papers addressing the problem of learning asymmetric classifiers [18, 29, 49, 80]. Despite their minor differences, the common approach of these methods is to setup an asymmetric expected loss where false negatives are penalized more than false positives. Viola and Jones [80] introduced an asymmetric loss: false negatives costs k times more than false positives. They applied an asymmetric re-weighting technique before training each weak classifier so that the result is equivalent to minimizing their asymmetric loss. Ma and Ding [49] proposed a cost-sensitive learning technique [18] to face detection with a re-weighting scheme similar to that of Viola and Jones [80]. However, the weight-updating rule [18] involves a parameter c which is application-dependent and needs extensive trials for best performance. Masnadi-Shirazi and Vasconcelos summarized these and other similar asymmetric boosting methods in [51]. To our best knowledge, these methods can not be done online.

In this chapter, we introduce a novel boosting algorithm which addresses the problem of learning *online* an *asymmetric* boosted classifier. The idea of learning *online* is similar to the Online Boosting algorithm [56] having been recently used [24, 34, 64], but with a faster convergence rate. On the aspect of learning *asymmetrically*, our method is similar to that of Viola and Jones [80]. Besides, our method also asymptotically balances the *skewness* of the weight distribution of the two classes presented to each weak classifier (to be discussed in the chapter), so that they are trained more equally.

The remaining parts of the chapter are organized as follows. In Section 3.2, we introduce our algorithm for the online learning of a boosted asymmetric classifier, providing

analysis and drawing relations to the offline case. Section 3.3 gives the experimental results of comparing our method with other methods. We present conclusions in Section 3.4.

3.2 Online Asymmetric Boosting

Suppose data come in online, and at iteration N , we observe a new training data point $x_N \in \mathcal{X}^1$ and its corresponding class $y_N \in \mathcal{Y} = \{-1, 1\}$. We assume the distribution of the two classes is highly skewed, *i.e.*, $\mathbb{P}[y = 1] \ll \mathbb{P}[y = -1]$. We wish to learn online a boosted classifier $F(x)$ to predict the class of an unknown point x . The boosted classifier is an ensemble of M discrete-valued weak classifiers $h_m(x) \in \mathcal{Y}$ related by:

$$F(x) \stackrel{\text{def}}{=} \text{sign}(f_M(x)) \quad (3.1)$$

$$f_M(x) \stackrel{\text{def}}{=} \sum_{m=1}^M c_m h_m(x), \quad (3.2)$$

where c_m is the voting weight associated with the m -th weak classifier $h_m(x)$ [20].

Our goal is to learn *online* $(c_m, h_m(x))$ for all $m = 1, 2, \dots, M$ so as to minimize an expected asymmetric loss $\mathcal{J}_\lambda(F)$, similar to the proposal of Viola and Jones in [80]:

$$\mathcal{J}'_\lambda(F) \stackrel{\text{def}}{=} \mathbb{E}[\varepsilon_\lambda(F(x), y)], \quad (3.3)$$

where $\varepsilon_\lambda(F(x), y)$ represents the asymmetric loss of a point (x, y) mis-classified by

¹The dimensionality of input is not important in this chapter. Hence we adopt normal letter x for simplicity.

$F(x)$, defined by

$$\varepsilon_\lambda(F(x), y) \stackrel{\text{def}}{=} \begin{cases} 1/\sqrt{\lambda} & \text{if } y = 1 \text{ and } F(x) = -1 \\ \sqrt{\lambda} & \text{if } y = -1 \text{ and } F(x) = 1 \\ 0 & \text{otherwise,} \end{cases} \quad (3.4)$$

where λ is a parameter which tells how many times we penalize a false negative more than a false positive.

Notice that unlike in learning offline, in learning online one cannot assume that the prior probability $\mathbb{P}[y]$ is known *a priori* because we do not observe the whole training set at the beginning. The loss $\mathcal{J}'_\lambda(F)$ is therefore slightly different from the asymmetric error $\mathcal{J}_\lambda(F)$ introduced in (2.7). However, if $\mathbb{P}[y]$ is given, there exists a λ' such that $\mathcal{J}'_{\lambda'}(F)$ equals a fixed multiplication of $\mathcal{J}_\lambda(F)$, as shown in theorem 3.2.1. In this sense, the two loss functions are actually equivalent.

Theorem 3.2.1. *Let $\lambda' = \lambda \frac{\mathbb{P}[y=1]}{\mathbb{P}[y=-1]}$. The two loss functions $\mathcal{J}'_{\lambda'}(F)$ and $\mathcal{J}_\lambda(F)$ are equivalent up to a multiplication factor.*

Proof. We have:

$$\begin{aligned} & \mathcal{J}_{\lambda'}(F) \\ &= \frac{1}{\lambda' + 1} (\mathbb{P}[F_k(x) = 1 | y = -1] + \lambda' \mathbb{P}[F_k(x) = -1 | y = 1]) \\ &= \frac{1}{\lambda' + 1} \left(\mathbb{P}[F_k(x) = 1 | y = -1] + \lambda \frac{\mathbb{P}[y = 1]}{\mathbb{P}[y = -1]} \mathbb{P}[F_k(x) = -1 | y = 1] \right) \\ &= \frac{\sqrt{\lambda}}{(\lambda' + 1) \mathbb{P}[y = -1]} \\ & \quad \left(\frac{1}{\sqrt{\lambda}} \mathbb{P}[F_k(x) = 1 \wedge y = -1] + \sqrt{\lambda} \mathbb{P}[F_k(x) = -1 \wedge y = 1] \right) \\ &= \frac{\sqrt{\lambda}}{(\lambda' + 1) \mathbb{P}[y = -1]} \mathcal{J}_\lambda(F). \end{aligned} \quad (3.5)$$

□

Let the *empirical probability distribution* $\hat{\mathbb{P}}$ after observing N examples be defined as:

$$\hat{\mathbb{P}} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N \delta_{(x_i, y_i)}, \quad (3.6)$$

where $\delta_{(x_i, y_i)}(\cdot)$ is a probability distribution concentrated at point (x_i, y_i) , and denote $\hat{\mathbb{E}}$ as its corresponding *empirical expectation* (i.e., expectation w.r.t. $\hat{\mathbb{P}}$).

Like other boosting methods, instead of minimizing the expected asymmetric loss $\mathcal{J}'_\lambda(F)$ directly, we minimize an upper bound of the empirical asymmetric loss $J_\lambda(F)$ ²:

$$J_\lambda(f_m) \stackrel{\text{def}}{=} \hat{\mathbb{E}} \left[\sqrt{\lambda}^y \exp(-y f_m(x)) \right] \geq \hat{\mathbb{E}} [\varepsilon_\lambda(F(x), y)]. \quad (3.7)$$

However, unlike other methods, while minimizing this bound, we also seek to balance the *skewness* of the weight distribution of the two classes presented to each of the weak classifier, allowing them to be trained more equally and thereby resulting in a better performance. This can be achieved by carefully controlling the weights prior to training each weak classifier.

3.2.1 Why Equal Skewness?

To see why this is the case, let us first define the term “skewness” as follows.

Definition The skewness of a binary distribution $\mathbb{P}[y]$, (where $y \in \mathcal{Y} = \{-1, 1\}$) is given by

$$\gamma \stackrel{\text{def}}{=} \log \frac{\mathbb{P}[y = -1]}{\mathbb{P}[y = 1]}. \quad (3.8)$$

Our definition of skewness is slightly different from common definitions of skewness. Nevertheless, its property is similar: γ is positive for right-skewed distributions (i.e., $\mathbb{P}[y = -1] > \mathbb{P}[y = 1]$), and is negative for left-skewed distributions (i.e., $\mathbb{P}[y = -1] < \mathbb{P}[y = 1]$).

²See [80] for more details.

Consider Viola and Jones' reweighting technique in [80]. In this technique, before training the m -th weak classifier, the current weight of every example (x_n, y_n) (for $n = 1 \dots N$) is multiplied with $\sqrt[2M]{\lambda_{y_n}^y}$. Let $v_m(x, y)$ and $w_m(x, y)$ be the functions representing the weight of example (x, y) before and after being multiplied respectively, at the m -th stage. Also, denote by γ_m the skewness of the (empirical) class distribution weighted by $v_m(x, y)$:

$$\gamma_m \stackrel{\text{def}}{=} \log \frac{\hat{\mathbb{P}}_{v_m}[y = -1]}{\hat{\mathbb{P}}_{v_m}[y = 1]}, \quad (3.9)$$

where

$$\hat{\mathbb{P}}_{v_m} \stackrel{\text{def}}{=} \left(\sum_{i=1}^N v_m(x_i, y_i) \delta_{(x_i, y_i)} \right) / \left(\sum_{i=1}^N v_m(x_i, y_i) \right), \quad (3.10)$$

and by γ'_m the skewness of the (empirical) class distribution weighted by $w_m(x, y)$:

$$\gamma'_m \stackrel{\text{def}}{=} \log \frac{\hat{\mathbb{P}}_{w_m}[y = -1]}{\hat{\mathbb{P}}_{w_m}[y = 1]}, \quad (3.11)$$

where

$$\hat{\mathbb{P}}_{w_m} \stackrel{\text{def}}{=} \left(\sum_{i=1}^N w_m(x_i, y_i) \delta_{(x_i, y_i)} \right) / \left(\sum_{i=1}^N w_m(x_i, y_i) \right). \quad (3.12)$$

The following theorem shows how the skewness is changed due to the reweighting technique.

Theorem 3.2.2. *Suppose the weighted marginal distribution of y presented to the m -th weak classifier has skewness γ_m , if we multiply the weight of each example (x_n, y_n) by $\sqrt{\lambda_{y_n}^y}$, the newly weighted marginal distribution has skewness $\gamma'_m = \gamma_m - \log \lambda_m$.*

Proof. By definition, we have:

$$w_m(x, y) = v_m(x, y) \sqrt{\lambda_m^y}. \quad (3.13)$$

Let w_m^p and w_m^n be the total weights of positive and negative examples respectively *after* being multiplied, and v_m^p and v_m^n be the total weights of positive and negative examples

Figure 3.1: A case where examples are highly skewed $\gamma_1 \gg 1$: Positives are squares, negatives are triangles. Suppose we train with four weak classifiers, which are linear separators. The first classifier is labeled as ‘1’. On the left is the result trained by Viola and Jones’ method: while the subsequent weak classifiers are well-modeled the first classifier is highly affected by γ_1 , resulting in rejecting a significant proportion of positives. On the right is the result trained by our method. Most of the positives are preserved.

respectively *before* being multiplied. By definition:

$$\gamma'_m = \log \frac{w_m^n}{w_m^p} = \log \frac{v_m^n / \sqrt{\lambda_m}}{v_m^p \sqrt{\lambda_m}} \quad (3.14)$$

$$= \log \frac{v_m^n}{v_m^p \lambda_m} = \log \frac{v_m^n}{v_m^p} - \log \lambda_m = \gamma_m - \log \lambda_m. \quad (3.15)$$

□

Suppose the weight distribution of the classes initially have skewness γ_1 . Based on theorem 3.2.2, by multiplying the weights with $\sqrt[2M]{\lambda^{y_n}}$, the new skewness is given by:

$$\gamma'_1 = \gamma_1 - \frac{1}{M} \log \lambda. \quad (3.16)$$

However, after training a weak classifier, because of the balanced reweighting scheme of AdaBoost (corollary 2 in Friedman *et al.* [21]), the total weights of positive examples

Algorithm 2 Offline Asymmetric Boosting

- 1: Start with weights $v_n = 1/N, n = 1, 2, \dots, N$.
 - 2: **for** $m = 1$ to M **do**
 - 3: Choose λ_m .
 - 4: Set $w_n = v_n \sqrt{\lambda_m^{y_n}}, n = 1, 2, \dots, N$, and normalize w_n so that $\sum_{n=1}^N w_n = 1$.
 - 5: Fit weak classifier $h_m(x) \in \{-1, 1\}$ using data (x_n, y_n) with weights w_n .
 - 6: Set $c_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$, where $e_m = \sum_{n=1}^N w_n 1_{[y_n \neq h_m(x_n)]}$.
 - 7: Set $v_n = w_n \exp(-y_n c_m h_m(x_n)), n = 1, 2, \dots, N$.
 - 8: **end for**
 - 9: **return** the classifier $F(x) = \text{sign}(\sum_{m=1}^M c_m h_m(x))$.
-

and negative examples after being updated *are equal*; which means $\gamma_m = 0$ for all $m > 1$. Therefore, when the updated weights are multiplied with $^{2M}\sqrt{\lambda^{y_n}}$, the skewness of the weight distribution of the classes becomes:

$$\gamma'_m = -\frac{1}{M} \log \lambda. \quad (3.17)$$

Hence, while subsequent weak classifiers *may* be trained equally, training the first classifier is largely affected by γ_1 . The result is that the first classifier could be trained wrongly, due to the high skewness of the weight distribution, as illustrated in figure 3.1. By ensuring all weak classifiers to be trained with equal skewness, we expect their decision boundaries after training to have equal effects. Therefore, we seek to balance the *skewness* of the weight distribution of the classes presented to the weak classifiers.

3.2.2 Balancing the Skewness Presented to Weak Classifiers

To simplify the explanation, assume for the moment that we are learning offline. The offline version of our algorithm is sketched in algorithm 2. The algorithm itself is similar to Viola and Jones's Asymmetric AdaBoost [80]. However, real-valued weak classifiers [67] are replaced by discrete-valued weak classifiers [20]. Rather than multiplying the weights presented to the m -th weak classifier with $^{2M}\sqrt{\lambda^{y_n}}$, the algorithm multiplies them with more general parameters $\sqrt{\lambda_m^{y_n}}$ (e.g., λ_m can be set to $^M\sqrt{\lambda}$). Our goal is to design

parameters $\lambda_1, \lambda_2, \dots, \lambda_M$ so that the algorithm:

1. Minimizes $J_\lambda(f_M)$ in (3.7).
2. Ensures equal skewness of the weight distribution of the classes presented to weak classifiers.

Let us denote, by $v_{m,n} \stackrel{\text{def}}{=} v_m(x_n, y_n)$ and by $w_{m,n} \stackrel{\text{def}}{=} w_m(x_n, y_n)$, the weight of the n -th example *before* and *after* being multiplied with $\sqrt{\lambda_m^{y_n}}$ respectively. According to the algorithm, they are propagated by the following rules:

$$v_{m+1}(x_n, y_n) = w_m(x_n, y_n) \exp(-y_n c_m h_m(x_n)) \quad (3.18)$$

$$w_m(x_n, y_n) = v_m(x_n, y_n) \sqrt{\lambda_m^{y_n}} / Z_m \quad (3.19)$$

where Z_m is a normalization factor to make $w_m(x, y)$ a distribution. That is

$$Z_m \stackrel{\text{def}}{=} \sum_{n=1}^N v_m(x_n, y_n) \sqrt{\lambda_m^{y_n}}. \quad (3.20)$$

Similarly to AdaBoost, algorithm 2 has a few properties that are described in the following theorem and its corollaries.

Theorem 3.2.3. *Consider the Offline Asymmetric Boosting algorithm presented in (algorithm 2), the algorithm sequentially fits weak classifier $(c_m, h_m(x))$ by minimizing $\hat{\mathbb{E}} \left[\left(\prod_{i=1}^m \sqrt{\lambda_i^{y_i}} \right) \exp(-y(f_{m-1}(x) + ch(x))) \right]$ w.r.t. $(c, h(x))$.*

Proof. The proof is analogous to Result 1 in Friedman *et al.* [21], except that here we have an extra term $\prod_{i=1}^m \sqrt{\lambda_i^{y_i}}$. In addition, the proof in [21] requires an approximation of $h(x)$ up to the second order. The approximation is avoided in our proof. Let $w_m(x, y)$ be the weighting function of the examples when presenting to the m -th weak classifier.

By induction, we have

$$w_m(x, y) = \left(\prod_{i=1}^m \sqrt{\lambda_i^y} \right) \exp(-y f_{m-1}(x)) / Z_m, \quad (3.21)$$

where Z_m is the normalization factor.

The goal is rewritten as:

$$J_{m, \lambda_m}(c, h) \stackrel{\text{def}}{=} \hat{\mathbb{E}} \left[\left(\prod_{i=1}^m \sqrt{\lambda_i^y} \right) \exp(-y(f_{m-1}(x) + ch(x))) \right] \quad (3.22)$$

$$= \hat{\mathbb{E}} [Z_m w_m(x, y) \exp(-ych(x))] \quad (3.23)$$

$$= Z_m \hat{\mathbb{E}} [w_m(x, y)] \hat{\mathbb{E}}_{w_m} [\exp(-ych(x))], \quad (3.24)$$

where $\hat{\mathbb{E}}_{w_m} [\cdot]$ is the (empirical) expectation *w.r.t.* $\hat{\mathbb{P}}_{w_m}$, the (empirical) weight distribution for training the m -th weak classifier. Now consider the two choices of $h(x)$:

$$\begin{aligned} & \hat{\mathbb{E}}_{w_m} [e^{-ych(x)}] \\ &= \hat{\mathbb{E}}_{w_m} [e^{-ych(x)} (1_{yh(x)=1} + 1_{yh(x) \neq 1})] \end{aligned} \quad (3.25)$$

$$= \hat{\mathbb{E}}_{w_m} [e^{-c} 1_{yh(x)=1} + e^c 1_{yh(x) \neq 1}] \quad (3.26)$$

$$= e^{-c} \hat{\mathbb{E}}_{w_m} [1_{yh(x)=1}] + e^c \hat{\mathbb{E}}_{w_m} [1_{yh(x) \neq 1}] \quad (3.27)$$

$$= e^{-c} \hat{\mathbb{P}}_{w_m} [h(x) = y] + e^c \hat{\mathbb{P}}_{w_m} [h(x) \neq y]. \quad (3.28)$$

Denote by $\hat{\epsilon}_{w_m}(h) = \hat{\mathbb{P}}_{w_m} [h(x) \neq y]$ the *empirical weighted error* of weak classifier $h(x)$, then:

$$\begin{aligned} \hat{\mathbb{E}}_{w_m} [e^{-ych(x)}] &= e^{-c} (1 - \hat{\epsilon}_{w_m}(h)) + e^c \hat{\epsilon}_{w_m}(h) \\ &= e^{-c} + (e^c - e^{-c}) \hat{\epsilon}_{w_m}(h). \end{aligned} \quad (3.29)$$

Therefore, minimizing (3.22) *w.r.t.* $h(x)$ is equivalent to minimizing $\epsilon_{w_m}(h)$ *w.r.t.* $h(x)$.

Next, suppose $\hat{h}(x)$ is the weak classifier trained by minimizing $\hat{\epsilon}_{w_m}(h)$. Setting the derivative of $J_{m,\lambda_m}(c, \hat{h})$ w.r.t. c to 0, we get:

$$\hat{c} \stackrel{\text{def}}{=} \arg \min_c \hat{\mathbb{E}}_{w_m} \left[e^{-yc\hat{h}(x)} \right] = \frac{1}{2} \log \frac{1 - \hat{\epsilon}_{w_m}(\hat{h})}{\hat{\epsilon}_{w_m}(\hat{h})}. \quad (3.30)$$

□

Corollary 3.2.4. *Consider the Asymmetric AdaBoost algorithm of Viola and Jones [80] where Discrete AdaBoost is used instead of Real AdaBoost. The algorithm sequentially fits weak classifier $(c_m, h_m(x))$ by minimizing w.r.t. $(c, h(x))$ the following function:*

$$\hat{\mathbb{E}} \left[{}^{2M}\sqrt{\lambda y^m} \exp(-y(f_{m-1}(x) + ch(x))) \right]. \quad (3.31)$$

Proof. The results follows from theorem 3.2.3 considering $\lambda_1 = \lambda_2 = \dots = \lambda_M = \sqrt[M]{\lambda}$. □

Corollary 3.2.5. *Consider the Offline Asymmetric Boosting algorithm presented in (algorithm 2), after the m -th weak classifier is trained:*

$$\hat{\mathbb{E}} [yh_m(x)v_{m+1}(x, y)] = 0. \quad (3.32)$$

Proof. The result is similar to corollary 2 in Friedman *et al.* [21], and follows from considering setting the partial derivative of $J_{m,\lambda_m}(c, h)$ w.r.t. c to 0 at point $(c_m, h_m(x))$. □

Corollary 3.2.6. *Consider the Offline Asymmetric Boosting algorithm presented in algorithm 2, at the optimal $h_m(x) = \arg \min_h J_{m,\lambda_m}(c, h)$:*

$$\hat{\mathbb{E}} [yv_{m+1}(x, y)] = 0. \quad (3.33)$$

Proof. Setting the point-wise derivative of $J_{m,\lambda_m}(c, h)$ w.r.t. $h(x)$ to 0, and then marginalizing it over x , we get the result. $\square$

Theorem 3.2.3 shows that at the m -th weak classifier, the algorithm fits $(c_m, h_m(x))$ by minimizing

$$J_{m,\lambda_m}(c, h) \stackrel{\text{def}}{=} \hat{\mathbb{E}} \left[\left(\prod_{i=1}^m \sqrt{\lambda_i^y} \right) \exp(-y(f_{m-1}(x) + ch(x))) \right]. \quad (3.34)$$

Therefore, to minimize $J_\lambda(f_M)$ in (3.7), the following condition must be true:

$$\prod_{m=1}^M \lambda_m = \lambda. \quad (3.35)$$

Note that AdaBoost itself is an M -stage greedy process where at each stage, a weak classifier is learned given that the previous weak classifiers are fixed. Hence, to minimize the asymmetric error of the boosted classifier, at least the last weak classifier must be chosen to minimize this error. Our method does not principally minimize $J_\lambda(f_M)$ from the beginning (and neither do other methods [29, 49, 80]). As pointed out by Viola and Jones, by doing so the first weak classifier absorbs all the asymmetric weights, leaving only symmetric weights to subsequent weak classifiers, which might not be a good idea. It is better to distribute the asymmetric weights among the weak classifiers equally, so that each of them only absorb the same amount of asymmetric weights, and give similar results.

The relationship of γ_m and γ'_m is given by (theorem 3.2.2):

$$\gamma'_m = \gamma_m - \log \lambda_m. \quad (3.36)$$

For the first weak classifier, γ_1 is the skewness of the original unweighted class distribu-

tion. For subsequent weak classifiers, if $h_{m-1}(x)$ is at the optimal

$$h_{m-1}(x) = \arg \min_h J_{m-1, \lambda_{m-1}}(c, h), \quad (3.37)$$

then based on corollary 3.2.6, the total weights of positive and negative examples weighted by $v_m(x, y)$ are equal, implying that $\gamma_m = 0$.

Therefore, by assuming that all classifiers $h_m(x)$ minimize their goal $h_m(x) \rightarrow \arg \min_h J_{m, \lambda_m}(c, h)$, the second condition can be established:

$$\gamma_1 - \log \lambda_1 = -\log \lambda_m \quad \forall m \in \{2..M\}. \quad (3.38)$$

Solving the system of (3.35) and (3.38) analytically, we get:

$$\lambda_1 = \exp \left(\frac{1}{M} \log \lambda + \frac{M-1}{M} \gamma_1 \right) \quad (3.39)$$

$$\lambda_m = \exp \left(\frac{1}{M} (\log \lambda - \gamma_1) \right) \quad \forall m \in \{2..M\}. \quad (3.40)$$

However, in practice $h_{m-1}(x)$ are not always at the optimal, because we use crude approximations to conditional expectation, such as decision trees or other constrained models, and the training size may not be large enough. Therefore, γ_m may not be 0 for $m > 1$.

We estimate λ_m based on γ_m and the remaining amount of λ to be distributed among the remaining weak classifiers. Suppose up to $m-1$ weak classifiers have been trained (and hence the values of $\lambda_1, \lambda_2, \dots, \lambda_{m-1}$ are obtained). The value of λ_m is chosen by constraining it with condition (3.35), assuming that all remaining weak classifiers minimize their goal ($h_m(x) \rightarrow \arg \min_h J_{m, \lambda_m}(c, h)$), and by seeking to balance the skewness of the weight distributions of the classes among the remaining weak classifiers. Thus, to

find the value of λ_m , we establish a system of equations as follows:

$$\prod_{m'=1}^M \lambda_{m'} = \lambda \quad (3.41)$$

$$\gamma_m - \log \lambda_m = -\log \lambda_{m'} \quad \forall m' \in \{m+1..M\}, \quad (3.42)$$

where the variables are $\lambda_m, \lambda_{m+1}, \dots, \lambda_M$. Similarly to (3.39), once we solve this system, we get:

$$\lambda_m = \exp \left(\frac{1}{M-m+1} \left(\log \lambda - \sum_{i=1}^{m-1} \log \lambda_i \right) + \frac{M-m}{M-m+1} \gamma_m \right). \quad (3.43)$$

3.2.3 Online Learning

Our challenge is to convert algorithm 2 into an online learning algorithm. The common technique to do so is to replace offline weak classifiers with online versions, so that each new example can be swept through each weak classifier just once, and then discarded. The advantage of online learning algorithms is, one can learn from a much larger set with comfortable computational cost, and yet, without having to store the training data.

Designing an online weak classifier is not a problem. There are many of them available in the literature (*e.g.*, Naïve Bayes, LDA, perceptron). The challenge though, is how to present the proper weights of the new example to each weak classifier so that they converge asymptotically to their offline counterparts.

Recently, some variants of boosting adapted to online learning have been proposed in the literature [19, 56]. They are, however, mainly designed for the symmetric case³. Here, we are interested in ideas proposed by Oza [56] which have been recently applied to computer vision problems [24, 34, 64]. Although their work was not designed for asymmetric classifiers, some of their ideas are propagated to our method.

³*i.e.*, false positives and false negatives are of equal importance.

Algorithm 3 *Propagate_Weights*($\hat{y}, y, v, \mathbf{v}_m$)

```

1: Compare predicted output  $\hat{y}$  and desired output  $y$ .
2: if true positive then
3:   Set  $v_m^{tp} = v_m^{tp} + v$ .
4:   Set  $a_m = (v_m^{tp} + v_m^{fn}) / (v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn})$ .
5:   Set  $v = v_m^{tp} \frac{a_m N / 2}{v_m^{tp}}$ 
6: else if false negative then
7:   Set  $v_m^{fn} = v_m^{fn} + v$ .
8:   Set  $a_m = (v_m^{tp} + v_m^{fn}) / (v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn})$ .
9:   Set  $v = v_m^{fn} \frac{(1-a_m) N / 2}{v_m^{fn}}$ 
10: else if false positive then
11:   Set  $v_m^{fp} = v_m^{fp} + v$ .
12:   Set  $a_m = (v_m^{tp} + v_m^{fn}) / (v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn})$ .
13:   Set  $v = v_m^{fp} \frac{a_m N / 2}{v_m^{fp}}$ 
14: else
15:   Set  $v_m^{tn} = v_m^{tn} + v$ .
16:   Set  $a_m = (v_m^{tp} + v_m^{fn}) / (v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn})$ .
17:   Set  $v = v_m^{tn} \frac{(1-a_m) N / 2}{v_m^{tn}}$ 
18: end if
19: return  $(a_m, v, \mathbf{v}_m)$ .

```

3.2.3.1 Propagating the Weights

In [56], Oza propagated the weights rather than by using the normal updating rule, which is sensitive to wrong predictions at early iterations, but by seeking to achieve two conditions which will be eventually guaranteed by the normal updating rule. The first condition is, the total weight of examples presented to a weak classifier is equal to N . It is reasonable considering that the weight of each new example can be set to 1 initially, and that the weight-updating rule is just a way of re-distributing the weights among the examples.

The second condition comes from the fact that, after c_m is optimally chosen (given $h_m(x)$ is fixed), the total weights presented to the next weak classifier, of correctly classified examples and wrongly classified examples, *are equal* (see corollary 2 in [21]). By asymptotically tuning the weights presented to the next classifiers to have a balance between the sum of correctly classified weights and that of wrongly classified weights, the

weights are more similar to what they are in offline AdaBoost, thereby convergence can occur more rapidly. To achieve both conditions asymptotically, they scaled the weights as follows:

$$\bar{v}_{m+1,n} = \begin{cases} \bar{v}_{m,n} \frac{N/2}{\bar{v}_m^{sc}} & \text{if } y_n = h_m(x_n) \\ \bar{v}_{m,n} \frac{N/2}{\bar{v}_m^{sw}} & \text{if } y_n \neq h_m(x_n) \end{cases} \quad (3.44)$$

where $\bar{v}_m^{sc}$ and $\bar{v}_m^{sw}$ are the total weights of the examples, correctly classified and wrongly classified by $h_m(x)$ respectively, after observing N examples; and $\lambda_{m,n}$ is the weight of example (x_n, y_n) presented to the m -th weak classifier. The idea is, when N is large, the total weight presented to the next weak classifier of those correctly classified and that of those wrongly classified should both converge to $N/2$.

The property of balance between the total weight of correctly classified examples and that of wrongly classified examples also holds for weights $v_{m,n}$ (equations (3.18) and (3.19)) in our method (see corollary 3.2.5 for the proof). Besides, we assume that each weak classifier $h_m(x)$ eventually reaches an (possibly local) optimal point when $N \rightarrow \infty$. Based on corollary 3.2.6, it means the total weight of positive examples and the total weight of negative examples will eventually be *equal*.

Therefore, in our method, the weights $v_{m,n}$ are propagated to asymptotically achieve three conditions rather than two, expecting a faster convergence rate. The first two conditions are the same as those of Oza and Russel, but applied to weights $v_{m,n}$. The last condition is to ensure the total weights $v_{m,n}$ of positive examples and negative examples to be equal.

Our weight-updating rule is as follows:

$$v_{m+1,n} = \begin{cases} v_{m,n} \frac{a_m N/2}{v_m^{tp}} & \text{if } y = 1 \text{ and } h_m(x_n) = 1 \\ v_{m,n} \frac{(1-a_m)N/2}{v_m^{fn}} & \text{if } y = 1 \text{ and } h_m(x_n) = -1 \\ v_{m,n} \frac{a_m N/2}{v_m^{fp}} & \text{if } y = -1 \text{ and } h_m(x_n) = 1 \\ v_{m,n} \frac{(1-a_m)N/2}{v_m^{tn}} & \text{if } y = -1 \text{ and } h_m(x_n) = -1 \end{cases} \quad (3.45)$$

where v_m^{tp} , v_m^{fn} , v_m^{fp} , and v_m^{tn} are the total weights of, true positive examples, false negative examples, false positive examples, and true negative examples respectively, after observing N examples; and a_m is an estimate of the weighted probability that $h_m(x)$ predicts a positive class, given by:

$$a_m \stackrel{\text{def}}{=} \frac{v_m^{tp} + v_m^{fn}}{v_m^{tp} + v_m^{fn} + v_m^{fp} + v_m^{tn}}. \quad (3.46)$$

Theorem 3.2.7 verifies that our weight-updating rule asymptotically achieves the three conditions above. Note that γ_m can be estimated from a_m by:

$$\gamma_m = \log \frac{\hat{\mathbb{P}}_{v_m}[y = -1]}{\hat{\mathbb{P}}_{v_m}[y = 1]} \approx \log \frac{1 - a_m}{a_m}. \quad (3.47)$$

Theorem 3.2.7. *Consider the weight-updating rule proposed in (3.45), when $N \rightarrow \infty$:*

$$v_m^{tp} + v_m^{fn} + v_m^{fp} + v_m^{tn} \rightarrow N \quad (3.48)$$

$$v_m^{tp} + v_m^{fn} - v_m^{fp} - v_m^{tn} \rightarrow 0 \quad (3.49)$$

$$v_m^{tp} - v_m^{fn} - v_m^{fp} + v_m^{tn} \rightarrow 0 \quad (3.50)$$

Proof. We prove by induction. These conditions are true for $m = 1$. Suppose they are true up to $m = i - 1$ for some i . We consider v_i^{tp} , v_i^{fn} , v_i^{fp} , and v_i^{tn} as the total weight estimates of true positives, false negatives, false positives, and true negatives respectively of the $(i - 1)$ -th classifier (multiplied by factor N). By definition, as $N \rightarrow \infty$:

$$v_i^{tp} \rightarrow a_{i-1}N/2 \quad (3.51)$$

$$v_i^{fn} \rightarrow (1 - a_{i-1})N/2 \quad (3.52)$$

$$v_i^{fp} \rightarrow a_{i-1}N/2 \quad (3.53)$$

$$v_i^{tn} \rightarrow (1 - a_{i-1})N/2 \quad (3.54)$$

The results follow. $\square$

3.2.3.2 Estimating c_m

Once $h_m(x)$ is updated using the weight-updating rule in (3.45), we need to estimate c_m . Let w_m^{tp} , w_m^{fn} , w_m^{fp} , and w_m^{tn} be the total weights $w_{m,n}$ of, true positive examples, false negative examples, false positive examples, and true negative examples respectively, after observing N examples. Based on theorem 3.2.3, we get:

$$c_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}, \quad (3.55)$$

where

$$e_m \stackrel{\text{def}}{=} \hat{\epsilon}_{w_m}(h_m) \quad (3.56)$$

is the empirical weighted error of $h_m(x)$, given by

$$\hat{\epsilon}_{w_m}(h_m) \stackrel{\text{def}}{=} \hat{\mathbb{P}}_{w_m} [h_m(x) \neq y]. \quad (3.57)$$

From (3.19), we can estimate e_m by:

$$e_m \approx \frac{w_m^{fn} + w_m^{fp}}{w_m^{tp} + w_m^{fn} + w_m^{fp} + w_m^{tn}} \quad (3.58)$$

$$= \frac{v_m^{fn} \sqrt{\lambda_m} + v_m^{fp} / \sqrt{\lambda_m}}{(v_m^{tp} + v_m^{fn}) \sqrt{\lambda_m} + (v_m^{fp} + v_m^{tn}) / \sqrt{\lambda_m}} \quad (3.59)$$

$$= \frac{\lambda_m v_m^{fn} + v_m^{fp}}{\lambda_m (v_m^{tp} + v_m^{fn}) + (v_m^{fp} + v_m^{tn})}. \quad (3.60)$$

The final online algorithm is presented in algorithm 4. Note that, since we seek to scale the sum of $v_{m,n}$ to N , we expect Z_m to be a constant factor. Therefore, we do not normalize $w_{m,n}$.

Algorithm 4 Online Asymmetric Boosting**Require:** λ

- 1: Set $\mathbf{v}_m = (v_m^{tp}, v_m^{tn}, v_m^{fp}, v_m^{fn})^T = \mathbf{0}$, $m = 1, 2, \dots, M$.
- 2: Set $\gamma_m = 0$, $m = 1, 2, \dots, M$.
- 3: **for** each new example (x_N, y_N) **do**
- 4: Set $v = 1$.
- 5: **for** $m = 1$ to M **do**
- 6: Set λ_m according to equation (3.43). {see section 3.2.2}
- 7: Learn $h_m(x)$ incrementally using new example $(x_N, y_N, v\sqrt{\lambda_m^{y_N}})$.
- 8: Set $(a_m, v, \mathbf{v}_m) = \text{Propagate_Weights}(h_m(x_N), y_N, v, \mathbf{v}_m)$. {see algorithm 3}
- 9: Set $\gamma_m = \log \frac{1-a_m}{a_m}$.
- 10: Set e_m according to equation (3.60).
- 11: Set $c_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$.
- 12: **end for**
- 13: **end for**
- 14: Output the classifier: $F(x) = \text{sign}(\sum_{m=1}^M c_m h_m(x))$.

3.3 Experimental Results

We performed three experiments to test the performance of our method, and to see how other online boosting methods for object detection problems [24, 34, 64] benefited from our method. All our experiments shown below classified objects in real time on a standard PC (Intel Pentium IV 2.00 GHz with 512 MB RAM).

3.3.1 Online Boosting

We compared our method with the Online Boosting algorithm of Oza [56] on several two-class datasets (table 3.1) from the UCI KDD [26] presented in the work. The last two datasets were our synthetic datasets. We used Naïve Bayes classifiers as weak classifiers. Because the datasets were symmetric (*i.e.*, positive examples and negative examples are equal), we set $\lambda = 1$. The testing environments were set up similar to that of Oza's, with 10 runs of 5-fold cross validation.

The results are summarized in table 3.1. We noticed that our method performed better than Online Boosting on under-complete datasets (those with a '+' in table 3.1),

Dataset	AdaBoost	Online Boosting	Our method
Promoters	0.8455	0.7136	0.7429+
Breast Cancer	0.9445	0.9573	0.9552
German Credit	0.735	0.6879	0.7013+
Chess	0.9517	0.9476	0.9501
Mushroom	0.9966	0.9987	0.9988
Census Income	0.9365	0.9398	0.9372
Synthetic 5-dim	0.9382	0.9049	0.9251
Synthetic 50-dim	0.8923	0.7972	0.8404+

Table 3.1: Online Boosting vs. Our method. Each row represents a symmetric dataset. Each column represents which learning algorithm was used. For each cell, a classifier was learned using the respected algorithm and the respected dataset. The value in the cell represents the accuracy of the classifier.

Figure 3.2: Learning curves for Mushroom dataset

and performed similarly on over-complete datasets. This was expected as our method converged faster than Online Boosting in early iterations, but eventually both methods converged to that of offline AdaBoost (*e.g.*, see figure 3.2) as more examples were observed.

Figure 3.3: Some classification results from PETS2001 dataset1. White box: region predicted as pedestrian. Black box: region predicted as non-pedestrian.

Figure 3.4: Performance on the PETS2001 datasets. Each example corresponds to a detected moving region. Left: PETS2001 dataset 1. Right: PETS2001 dataset 2.

3.3.2 Online Co-training for Moving Objects Detection

In [34], Javed *et al.* proposed an online co-training framework for object detection. In their work, a classifier is first trained offline on a generic scenario; and then updated using Online Boosting, allowing the boosted classifier to adapt to the current scenario.

We implemented their method on the problem of pedestrian detection. Similar to their work, we used a background model to subtract moving regions, followed by scaling the regions down to 30x30 pixels, and then applying PCA on the gradient magnitudes to obtain the global features. The classifier (pedestrian detector) was first trained offline using 150 manually labeled images, and then updated online. We compared the performance of their method when using Online Boosting, and when using our method. We

Figure 3.5: Some detection results of our Online Asymmetric Boosting on the MIT+CMU test set.

used the PETS2001 datasets ⁴ for training and testing the results. The performance (in figure 3.3 and figure 3.4) shows that co-training benefited from our method.

3.3.3 Online Learning for Face Detection

We analyzed the performance of five algorithms on the problem of face detection: Viola and Jones' Offline Asymmetric AdaBoost (DAB.VJ) [80], our Offline Asymmetric Boosting (DAB.A), our Online Asymmetric Boosting (DAB.OA), Online AdaBoost for feature selection using Online Boosting (DAB.OFS) [24], and Online AdaBoost for feature selection using our Online Asymmetric Boosting (DAB.OAFS). Here, the feature selection technique in the last two algorithms is the technique presented in [24, 64] that use Oza's Online Boosting.

The face detectors followed the same flowchart as in [79]. A total of 1500 frontal face images and 5000 non-face images were collected. Face and non-face images were scaled to a resolution of 24x24 pixels. Haar-like features were extracted from the sub-windows, forming a feature pool of 5000 features. For each algorithm, a cascade of 25 boosted classifiers, from coarse to fine, was set up. The first classifier had one weak classifier, while the final classifiers had 200 weak classifiers.

⁴<http://ftp.pets.rdg.ac.uk/PETS2001/>

Figure 3.6: ROC curves of the algorithms on the MIT+CMU set.

We tested the algorithms on the MIT+CMU frontal face test set ⁵ (see figure 3.6 and figure 3.7). Since the purpose was to compare these methods, we used fewer boosted classifiers and a smaller feature pool than existing offline-trained face detectors.

The values of λ tested in our experiments ranged from 1 to 10^8 . For early boosted classifiers, λ tended to be very large (*i.e.*, 10^5 and above) to compensate for the high skewness of the prior probabilities. As the negative (non-face) region got smaller, as in training subsequent boosted classifiers in the cascade, the skewness got lower and λ could be reduced.

For algorithms DAB.VJ, DAB.A, and DAB.OA, a weak classifier selected the best feature from the big feature pool. Hence, the training process for each of these algorithms took a very long time. For algorithms that use online feature selection (FS), *i.e.*, DAB.OFS and DAB.OAFS, the size of the pool was much smaller (we set the size to 250, the same as proposed in [24]), and hence the training processes were much shorter (about 20 times faster), but at the cost of reduced detection rates. Each boosted classifier was online-learned after running the learning algorithm for 10,000 to 100,000 iterations,

⁵http://vasc.ri.cmu.edu/idb/html/face/frontal_images/

Figure 3.7: An example of training a boosted classifier with 10 weak classifiers, and with different algorithms.

until the classifier appeared to be fully converged. We observed that online classifiers were often not so stable at first, and only converged after a fair number of iterations had passed. In training a boosted classifier with 10 weak classifiers (*i.e.*, figure 3.7), it often took from 50 to 1000 iterations before convergence started to appear. In training a boosted classifier with 200 weak classifiers, it took less than 10,000 iterations before convergence started to appear. When the best feature of the big pool was somehow swapped into the small pool, there was a good chance it would be rejected some time later, because of 1) being poorly trained, and 2) being compared with those not as good but well trained in the small pool. As a result, the best features selected by the additional FS process were, though relatively good, not so good as those selected from the big feature pool.

Careful examination of the ROC curves shows a few points:

- Our skewness balancing scheme resulted in a small performance gain over that of Viola and Jones.
- Our Online Asymmetric Boosting converged asymptotically to the first two offline algorithms as more examples were trained.

- The online feature selection technique also benefited from our Online Asymmetric Boosting algorithm.

3.4 Conclusions

In this chapter, we presented an integrated framework for learning asymmetric boosted classifiers and learning online to address the problem of learning online asymmetric boosted classifiers, which is applicable to object detection problems. In addition, our method seeks to balance the skewness of the weight distribution of the classes presented to the weak classifiers, allowing them to be trained more equally, and hence resulting in a small performance gain. In terms of online learning, we proposed to impose three conditions on the weights of the examples presented to weak classifiers, resulting in a faster convergence rate when compared to the Online Boosting algorithm [56] used in online boosting methods for object detection problems. Our experiments showed that, by replacing Online Boosting with our algorithm, about a 0-10% increase in accuracy and about 5-30% gain in learning speed were observed, depending on the problem the algorithms were applied to.

Chapter 4

Fast Training and Selection of Haar-like Features

4.1 Introduction

Face detection is one of the classic problems in computer vision, with applications in many areas like surveillance, robotics, multimedia processing, and human-computer interfaces (HCI). Developing a face detection system is still an open problem, but there have been important successes over the past several years. A standard image-based approach to face detection is scanning in raster order every squared sub-window of d pixels over multiple image scales, and classifying if each sub-window contains a face or not. When this strategy is used, face detection becomes a *rare event binary classification* problem, in a sense that among millions of sub-windows, only a few contain faces.

Among the most influential image-based systems is the cascade-based face detector of Viola and Jones [79]. The cascade, as illustrated in figure 2.1, is a one-branch tree of Q nodes. Each node is a classifier designed to reject a large portion of the non-face sub-windows and pass all of the face ones. Consequently, most non-face sub-windows are rejected quickly before reaching the final node, resulting in a very fast face detection

Figure 4.1: Four types of Haar-like features used in [79]

performance.

The node classifiers are constructed using an algorithm similar to AdaBoost [20]. The algorithm combines an ensemble of weak classifiers to produce a final boosted classifier with very high accuracy. Many of the recent papers using the cascade paradigm focus on improving the underlying boosting algorithm, such as AsymBoost [60, 80], FloatBoost [44], GentleBoost [45], and RealBoost [32, 87].

The weak classifiers are typically trained from a discrete set of Haar-like features, which are rectangular Haar-like wavelets, as illustrated in figure 4.1, at different positions, widths, and heights. Each feature is associated with a feature classifier, which classifies a sub-window by first integrating the sub-window with the feature's wavelet (using integral images), and then thresholding the value with a properly chosen threshold. The weak classifier is simply selected as the feature classifier with the smallest error.

One of the greatest obstacles to a wider use of cascades is that it takes a long time to train a cascade, even if all parameters are properly chosen a priori. It took weeks of computation to produce the final cascade in [79], using multiple machines. The bottleneck is in the training of a weak classifier, with time complexity $O(NT \log N)$, where N is the number of examples used and T is the size of the feature set. Training a single weak classifier has often run in minutes. In practice, one has to run many trials and choose the best configuration, resulting in even longer training time.

Since reducing N weakens the generalization, efforts have been made to reduce T by filtering the feature set and selecting only those *discriminant* for the current weak

classifier, *e.g.*, ranking using mutual-information [25] and Forward Feature Selection (FFS) [83]. However, ranking [25] may end up selecting only one single feature, which might *not* be the best feature, and the speed of FFS [83] is partly achieved by ignoring the weight distribution of boosting. Theoretically, FFS does not have the same hypothesis space as AdaBoost, so the features selected by FFS are not compatible for boosting. Other existing more accurate filters (*e.g.*, CMIM [77]) run in time nearly as slow as training a weak classifier. For further details about feature selection, we refer the readers to [9].

Wu *et al.* [86] proposed to decrease the training time by using *caching*. They exploited that in different rounds of the AdaBoost algorithm, the weight distribution changes but the feature values of the training set remain constant. Here, a feature value is the value of a feature extracted from a given example. If one sorts all the N feature values, extracted from N examples, for each feature classifier, and stores all the sorting orders in a pre-processing stage, one can later compute the best threshold for each feature classifier in time $O(N)$, regardless of the weights. Using this strategy, it is possible to reduce the training time from $O(NT \log N)$ to $O(NT)$. The reported empirical result was rather impressive, increasing the training speed by a factor of around 15 to 60. However, this technique requires a huge amount of memory to store the sorting orders: $O(NT)$. Suppose $N = 10,000$ and $T = 40,000$ and two bytes are used to store an index, the memory required is about 800MB. Nevertheless, to our best knowledge, this method has been the fastest implementation of training a face detector before our method is proposed, and is the source of comparison for our method.

In this chapter, we present a method to train weak classifiers with even faster speed than that of Wu *et al.* [87]. Our approach takes a different point of view from traditional methods. Noticing that both N and T are not only dominant but also important factors in constructing the weak classifiers, rather than trying to reduce either N or T , we *break up the multiplication* of N and T incurred in the time complexity of training a weak classi-

Figure 4.2: Diagram of this method to fast-train a weak classifier. The notations are explained in subsequent sections.

fier *by using statistics*. In our approach, the time complexity to train a weak classifier is approximately linear to either N or T , whichever is more dominant, *but not both*. By doing so, we can train weak classifiers with more examples – giving better generalization, as well as more features – giving a wider hypothesis set for the weak classifiers.

While we respect that feature filtering methods speed up the training process, we argue that even though the Haar-like feature set is over-complete, it is still a small discrete subset in the feature space. Only four types of features were actually used in [79]. By exploring other types one could potentially increase the detector’s performance [32, 45]. However, increasing the number of types directly increases the size of the feature set dramatically. This in turn makes the training time longer, which is by far one of the main reasons that stops many methods from exploring other feature types.

In our approach, we used 19 Haar-like feature types (figure 4.3) on a 24×24 image sub-window, generating in total 295,920 features, six times the number of features used by Viola and Jones in [81]. Yet, it took 6 seconds to train all feature classifiers and select the best one on a conventional Intel Pentium IV PC.

The key idea of our training approach is, we treat image sub-windows as high dimensional random vectors, and keep all the necessary statistics of the data prior to training a weak classifier. We rely on classification methods which only use statistics to construct the feature classifiers, and on boosting to accelerate the performance. After estimating

Figure 4.3: Nineteen types of features used in our experiment

the statistics, we can train all T feature classifiers in time *independent* of N , which allows us to train and select much more features with a small increase in the training time. Besides, the total memory storage for all the statistics is small, depending only on T and d - the number of pixels of a sub-window.

The remaining parts of the chapter are organized as follows. The framework of this method is described in section 4.2. In section 4.3, we discuss about the decisions made in the method, and use empirical results to justify them. Comparisons to other methods are presented in section 4.4. Conclusions are presented in section 4.5.

4.2 Training a Weak Classifier using Haar-like Features

4.2.1 Training a Boosted Classifier in the Cascade

For clarity, we briefly describe how boosting is used to train a node/boosted classifier in the cascade. For further details, we refer the readers to [79, 80]. Suppose the training set has N examples, $\{(\mathbf{I}_n, y_n)\}$ of the random couple $(\mathbf{I}, y)$, where $\mathbf{I}$ represents the image sub-window which is a random matrix of size $\sqrt{d}$ -by- $\sqrt{d}$, whose elements $\mathbf{I}_{q,p}$ (pixel at row q , column p) are random variables¹, and y_n is the class it belongs to. Let $y \in \mathcal{Y} = \{-1, 1\}$ with the notion that $y = 1$ being the face (positive) class and $y = -1$ being the non-face (negative) class.

We denote by $\vec{\mathbf{A}}$ a vectorized form of an arbitrary matrix $\mathbf{A}$ by aligning all the elements of $\mathbf{A}$ in a column with a predefined order². Let $\mathbf{x} = \vec{\mathbf{I}}$. Obviously, $\mathbf{x} \in \mathbb{R}^d$ and throughout this chapter, we consider it as a d -dimensional random vector with an unknown distribution.

The basic idea of AdaBoost is to train an ensemble of M weak classifiers $h_m(\mathbf{x})$ in the form:

$$f_M(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{m=1}^M c_m h_m(\mathbf{x}), \quad (4.1)$$

where c_m are voting coefficients. This is done greedily by training $(c_m, h_m(\mathbf{x}))$ for m from 1 to M . However, at the m -th stage, the training examples $(\mathbf{x}_n, y_n)$ are weighted differently using a weighting function $w_m(f_{m-1}(\mathbf{x}_n), y_n)$, based on the performance of the previous stage. The idea is to increase the weights of those wrongly classified examples and decrease the weights of those correctly classified examples, so that subsequent weak classifiers focus more on the wrongly classified (harder) examples. Different weighting functions lead to different optimizing criteria (*e.g.*, see [60, 80]), which affect

¹Note that unlike other chapters, it is essential to emphasize that the input is a multivariate in this chapter. Hence, we first denote it as matrix $\mathbf{I}$. Later, we vectorize $\mathbf{I}$ and present it as a random vector $\mathbf{x}$.

²Whichever order is used does not matter in this chapter.

the final false acceptance rate (FAR) and false rejection rate (FRR) of the ensemble. By controlling M and $w_m(\cdot)$, one can train the ensemble to have a very low FRR and a moderate FAR, suitable to reject a large portion of negative examples.

The choice of M , the choice of $w_m(\cdot)$, and how they affect the final FAR and FRR are not addressed in this chapter. We only assume that at the m -th stage, the weighting function $w_m(\cdot)$ is known. Since m is fixed throughout the process of training a weak classifier, we omit subscript m where possible for short hand notations.

4.2.2 Linear Relationship between Integral Image and Feature Values

Denote by $\mathbf{H}_t$ the matrix representing the t -th Haar-like wavelet *w.r.t.* the coordinate system of the probed sub-window. $\mathbf{H}_t$ is a matrix of size $\sqrt{d}$ -by- $\sqrt{d}$, whose elements' values are in a small discrete set. Also, denote by v_t the random variable representing the feature value observed from integrating the t -th Haar-like feature $\mathbf{H}_t$ with a sub-window $\mathbf{I}$. Let $\mathbf{h}_t = \vec{\mathbf{H}}_t$, we get:

$$v_t = \mathbf{H}_t \circ \mathbf{I} = \sum_{p=1}^d \sum_{q=1}^d \mathbf{H}_{t,q,p} \mathbf{I}_{q,p} = \mathbf{h}_t^T \mathbf{x}, \quad (4.2)$$

where $\mathbf{H}_{t,q,p}$ represents the pixel at column p and row q of matrix $\mathbf{H}_t$. This means v_t can be considered as the result of *linearly projecting* the random vector $\mathbf{x}$ down to $\mathbb{R}$ using $\mathbf{h}_t$ as the direction of projection (along with a scaling factor).

Now consider the integral image $\ddot{\mathbf{I}}$ [79] of sub-window $\mathbf{I}$, defined by:

$$\ddot{\mathbf{I}}_{q,p} \stackrel{\text{def}}{=} \sum_{p'=1}^p \sum_{q'=1}^q \mathbf{I}_{q',p'}. \quad (4.3)$$

Let $\ddot{\mathbf{x}} = \vec{\ddot{\mathbf{I}}}$. Equation (4.3) implies that both $\ddot{\mathbf{x}}$ and $\mathbf{x}$ are linearly related by:

$$\ddot{\mathbf{x}} = \mathbf{B}\mathbf{x}, \quad (4.4)$$

where $\mathbf{B}$ is a fixed and non-singular linear transformation matrix expressing the cumulative summary. A feature value v_t is also linearly related to the vectorized integral image $\ddot{\mathbf{x}}$:

$$v_t = \mathbf{h}_t^T \mathbf{x} = \mathbf{h}_t^T (\mathbf{B}^{-1} \ddot{\mathbf{x}}) = \mathbf{g}_t^T \ddot{\mathbf{x}}, \quad (4.5)$$

where $\mathbf{g}_t = \mathbf{B}^{-1T} \mathbf{h}_t$ is a fixed d -dimensional vector. As pointed out in [79], the nice aspect of combining Haar-like features with integral images is that $\mathbf{g}_t$ are *sparse* vectors with very few non-zero elements, typically less than 10. As a result, computing a feature value can be done extremely fast, with a few hundred CPU clocks.

Note that because vectors $\mathbf{g}_t$ are sparse, they should be better computed from their definitions rather than from multiplying the inverse of $\mathbf{B}$ with $\mathbf{h}_t$. Although $\mathbf{B}$ is a very large matrix (*i.e.*, of size d -by- d), it is only useful for theoretical derivations. In practice, we never have to compute $\mathbf{B}$, or its inverse, explicitly. An example is given in section 2.3.1, showing how to a feature value is computed as a linear combination of a few pixels of the integral image $\ddot{\mathbf{I}}$. In this example, the non-zero elements of vector $\mathbf{g}_t$ are the coefficients of the linear combination. Computation of matrix $\mathbf{B}$ is completely avoided.

4.2.3 Training Feature Classifiers using Statistics

By exploring the linear relationship above further, the statistics of v_t can be computed efficiently from the statistics of $\ddot{\mathbf{x}}$. Suppose μ_t and σ_t^2 are the mean and the variance of

Algorithm 5 Fast Training of a Weak Classifier

Require: $(\ddot{\mathbf{x}}_n, y_n, w_n)$ for $n = 1..N$. Here, $\ddot{\mathbf{x}}_n = \mathbf{B}\mathbf{x}_n$ denotes the integral image of the n -th input example $(\mathbf{x}_n, y_n)$, y_n denotes its class, and $w_n = w_m(\mathbf{x}_n, y_n)$ denotes its current weight.

- 1: **for** each class c **do**
- 2: Compute $\hat{z}_c = \sum_{n:y_n=c} w_n$.
- 3: Compute $\hat{\mathbf{m}}_c = \hat{z}_c^{-1} \sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n$.
- 4: Compute $\hat{\Sigma}_c = \left(\hat{z}_c^{-1} \sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \ddot{\mathbf{x}}_n^T \right) - \hat{\mathbf{m}}_c \hat{\mathbf{m}}_c^T$. {see sections 4.2.3 and 4.3.2}
- 5: **end for**
- 6: **for** each feature t **do**
- 7: **for** each class c **do**
- 8: Compute $\hat{\mu}_{t,c} = \hat{\mathbf{m}}_c^T \mathbf{g}_t$.
- 9: Compute $\hat{\sigma}_{t,c}^2 = \mathbf{g}_t^T \hat{\Sigma}_c \mathbf{g}_t$. {see section 4.2.2}
- 10: **end for**
- 11: Use $\{(\hat{z}_c, \hat{\mu}_{t,c}, \hat{\sigma}_{t,c}^2) | c \in \{-1, +1\}\}$ to train feature classifier $\eta_t(v_t)$ {see section 4.2.3}
- 12: **end for**
- 13: Select the feature classifier $\eta_t(v_t)$ with the smallest classification error, denote its index as t^* .
- 14: **return** $f_m(\mathbf{x}) = \eta_{t^*}(\mathbf{g}_{t^*}^T \ddot{\mathbf{x}}) = \eta_{t^*}(\mathbf{g}_{t^*}^T \mathbf{B}\mathbf{x})$ as the weak classifier.

v_t . Denote by $\mathbb{E}[\cdot]$ the expectation of a random variable or a random vector. We have:

$$\mu_t = \mathbb{E}[v_t] = \mathbb{E}[\ddot{\mathbf{x}}^T \mathbf{g}_t] = \mathbb{E}[\ddot{\mathbf{x}}^T] \mathbf{g}_t = \mathbf{m}_{\ddot{\mathbf{x}}}^T \mathbf{g}_t, \quad (4.6)$$

where $\mathbf{m}_{\ddot{\mathbf{x}}} = \mathbb{E}[\ddot{\mathbf{x}}]$ is the mean vector of $\ddot{\mathbf{x}}$, and:

$$\sigma_t^2 = \mathbb{E}[v_t^2] - \mathbb{E}[v_t]^2 \quad (4.7)$$

$$= \mathbb{E}[\mathbf{g}_t^T \ddot{\mathbf{x}} \ddot{\mathbf{x}}^T \mathbf{g}_t] - \mathbf{g}_t^T \mathbf{m}_{\ddot{\mathbf{x}}} \mathbf{m}_{\ddot{\mathbf{x}}}^T \mathbf{g}_t \quad (4.8)$$

$$= \mathbf{g}_t^T (\mathbb{E}[\ddot{\mathbf{x}} \ddot{\mathbf{x}}^T] - \mathbb{E}[\ddot{\mathbf{x}}] \mathbb{E}[\ddot{\mathbf{x}}]^T) \mathbf{g}_t \quad (4.9)$$

$$= \mathbf{g}_t^T \Sigma_{\ddot{\mathbf{x}}} \mathbf{g}_t, \quad (4.10)$$

where $\Sigma_{\ddot{\mathbf{x}}} = \mathbb{E}[\ddot{\mathbf{x}} \ddot{\mathbf{x}}^T] - \mathbb{E}[\ddot{\mathbf{x}}] \mathbb{E}[\ddot{\mathbf{x}}]^T$ is the covariance matrix of $\ddot{\mathbf{x}}$. Since $\mathbf{g}_t$ is a *sparse* vector, if we know both the mean vector and the covariance matrix of $\ddot{\mathbf{x}}$, computing μ_t

and σ_t^2 can be done extremely fast as well.

An important observation is, while $\Sigma_{\mathbf{x}}$ itself may not be accurately estimated due to large dimensionality, the projected statistics μ_t and σ_t^2 are much more accurately estimated as scalars. Additionally, higher order statistics may be obtained, but these are not explored in this chapter.

We decided to train feature classifiers using only statistics of feature values and class. Several reasons were considered for this decision. On the one hand, the performance of a feature classifier in terms of accuracy is reduced, which in turns affects the weak classifier. However, when the problem becomes harder, as in the subsequent layers of the cascade, most weak classifiers tend to perform poorly, with errors typically close to 40-50 percent [81]. At those layers, the performance of an ensemble relies less on the performance of individual weak classifiers, but rather on the boosting scheme of the boosting algorithm used. Experiments in section 4.3.3 justify our argument.

On the other hand, if the mean vector and the covariance matrix of the current stage's random vector are known, all the statistics needed (up to the second order) to train a classifier can be computed extremely fast, and *independent* of the sample size N . Besides, estimating the mean vector and the covariance matrix only needs to be done once per weak classifier, and then substantially used for all T features, saving a large amount of computation.

We investigate this idea using a simple type of feature classifier widely known in the literature: simple thresholding with class-conditional Gaussian distribution assumptions (*e.g.*, see chapter 2 in [17]). Here, the feature classifier $h_t(v_t)$ assumes

$$v_t|c \sim \mathcal{N}(\mu_{t,c}, \sigma_{t,c}^2) \quad (4.11)$$

for every class $c \in \mathcal{Y}$, where v_t is the random variable representing the observed feature value when the image sub-window is weighted by $w_m(\cdot)$, $\mu_{t,c}$ is the mean of the observed

feature value conditional on class c , and $\sigma_{t,c}^2$ is the variance of the observed feature value conditional on class c . That is, v_t is a linear projection of $\mathbf{u}$ down to a line where $\mathbf{u}$ is defined as the weighted version of $\mathbf{x}$ using $w_m(\cdot)$. To be precise, $\mathbf{u}$ is distributed by an empirical distribution $\hat{\mathbb{P}}_m$ given by:

$$\hat{\mathbb{P}}_m \stackrel{\text{def}}{=} \left(\sum_{n=1}^N w_m(\mathbf{x}_n, y_n) \delta_{\mathbf{x}_n, y_n} \right) / \left(\sum_{n=1}^N w_m(\mathbf{x}_n, y_n) \right), \quad (4.12)$$

where $\delta_{\mathbf{x},y}(\cdot)$ is a probability distribution concentrated at point $(\mathbf{x}, y)$.

There are some intuitive reasons for the Gaussian distribution assumptions. A feature value can be considered as a linear combination of all d (roughly 20^2 to 24^2 in practice) pixels of a sub-window. Based on the Central Limit Theorem, if the pixels are independent, the distribution of the feature value approaches a normal distribution. Independence may not hold for neighboring pixels in the sub-window, but it is more likely when the pixels are far away from each other. Therefore, despite the fact that $\mathbf{I}$ is not Gaussian, v_t is still likely approximately normally distributed. Evidence supporting this argument is reported in Wu *et al.* [86], Bourdev and Brandt [8], and in figures 4.7 and 4.8 in section 4.3.

To train the classifier, one finds a threshold $\theta_t \in \mathbb{R}$ and a parity $p_t \in \{-1, +1\}$ that minimizes the error rate $\epsilon_t(\theta_t, p_t)$ given by:

$$\epsilon_t(\theta_t, p_t) \stackrel{\text{def}}{=} \hat{\mathbb{P}}_m[v_t > \theta_t \wedge y = -p_t] + \hat{\mathbb{P}}_m[v_t < \theta_t \wedge y = p_t]. \quad (4.13)$$

The minimization problem is non-convex, but has a closed form solution, as described in appendix A as well as in [17], eliminating the need to linearly search for the optimal threshold. Once the optimal values (θ_t^*, p_t^*) have been obtained, the feature value is classified by:

$$\hat{y} = \eta_t(v_t) = p_t^* \text{sign}(v_t - \theta_t^*). \quad (4.14)$$

The feature classifier with the smallest error rate is chosen as the weak classifier.

We only need to estimate the statistics up to the second order to compute the probabilities in (4.13). Let $\ddot{\mathbf{u}} = \mathbf{B}\mathbf{u}$ be the integral image of $\mathbf{u}$, an analysis similar to equations from (4.6) to (4.10) shows that:

$$\mu_{t,c} = \mathbf{m}_{\ddot{\mathbf{u}}|c}^T \mathbf{g}_t \quad (4.15)$$

$$\sigma_{t,c}^2 = \mathbf{g}_t^T \Sigma_{\ddot{\mathbf{u}}|c} \mathbf{g}_t, \quad (4.16)$$

where $\mathbf{m}_{\ddot{\mathbf{u}}|c}$ and $\Sigma_{\ddot{\mathbf{u}}|c}$ are the mean vector and the covariance matrix of the $\ddot{\mathbf{u}}|c$ class-conditional random vector. Therefore, if $\mathbf{m}_{\ddot{\mathbf{u}}|c}$ and $\Sigma_{\ddot{\mathbf{u}}|c}$ are known, all the $\mu_{t,c}$ and $\sigma_{t,c}^2$ can be computed very fast as well.

The class-conditional mean vectors $\mathbf{m}_{\ddot{\mathbf{u}}|c}$ and covariance matrices $\Sigma_{\ddot{\mathbf{u}}|c}$ can be *estimated directly* from the weighted training set, by computing the weighted class-conditional sample mean vectors and covariance matrices:

$$\hat{\mathbf{m}}_{\ddot{\mathbf{u}}|c} = z_{\ddot{\mathbf{u}}|c}^{-1} \sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \quad (4.17)$$

$$\begin{aligned} \hat{\Sigma}_{\ddot{\mathbf{u}}|c} = & \left(z_{\ddot{\mathbf{u}}|c}^{-1} \sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \ddot{\mathbf{x}}_n^T \right) \\ & - \hat{\mathbf{m}}_{\ddot{\mathbf{u}}|c} \hat{\mathbf{m}}_{\ddot{\mathbf{u}}|c}^T, \end{aligned} \quad (4.18)$$

where $z_{\ddot{\mathbf{u}}|c} = \sum_{n:y_n=c} w_n$ is the total weight of class c , and $w_n = w_m(\mathbf{x}_n, y_n)$ is the weight applied to the n -th training example $(\mathbf{x}_n, y_n)$.

This method's pseudo-code is illustrated in algorithm 5.

4.2.4 Time Complexity

This method can be broken into two components. The first component (from line 1 to line 5) estimates the global class-conditional statistics, which is dominated by estimating

the covariance matrices. Given N examples of d dimensions, the time complexity of this component is $O(Nd^2)$. However, d is considered a constant, so the time complexity is $O(N)$. In practice, d is small and a fast algorithm exists to compute a covariance matrix³.

The other component (from lines 6 to 12) computes the local class-conditional means and variances by projecting the original class-conditional mean vectors and covariance matrices in different directions. Because $\mathbf{g}_t$ are *sparse*, the projections can be computed extremely fast by taking into account only a few non-zero elements per projection. All other elements are untouched. Thus, the time to compute the projected statistics for a feature classifier can be considered $O(1)$. Each feature classifier is trained using only six real numbers. Therefore, its training time complexity is also $O(1)$. In total, the time complexity of the second component is $O(T)$, and the total time to train a weak classifier is $O(N + T)$.

4.3 Implementation and Experimental Results

4.3.1 Experiment Setup

To justify the arguments discussed in the previous section, we ran a few experiments. We collected 1521 face images from the BioID Face Database⁴, 508 face images from the AR Face Database [50], and a few hundred more face images with known (manually labeled) face locations from the web. Altogether we obtained about 2500 face images. The faces were scaled and aligned to a base resolution of 24×24 pixels. They were further mirrored to form a training set of around 5000 faces. Note that our face training set contained mostly adult faces. Similar to traditional methods, we generated sufficient non-face sub-windows from a collection of a few thousand large images containing no faces.

³See our discussion in section 4.3.2.

⁴<http://www.bioid.com/downloads/facedb/index.php>

Figure 4.4: Comparison on the time to train a weak classifier

The cascades were trained using the bootstrapping strategy used in [79, 80, 81]. We implemented two types of cascade: one using the faster AdaBoost implementation proposed by Wu *et al.* [86], the other using our proposed training method. Each node classifier was trained using AsymBoost [80] with a training set of 5000 positive sub-windows and 5000 negative sub-windows which were false positives of the previous node. Nineteen different feature types (figure 4.3) were used to generate a total of 295,920 Haar-like features.

For testing, we used the MIT+CMU test set which consists of 130 grayscale images with 507 frontal faces [65]. Post-processing was the same as in [81]. The experiments were done on a 2.8GHz Intel Pentium IV PC with 1.5 GB memory.

4.3.2 Training Time of a Weak Classifier

Figure 4.4 shows the time to train a weak classifier on both methods. To be as fair as possible, we optimized the critical code sections in both implementations. In the faster AdaBoost implementation, the section of computing the T thresholds was written in highly optimized *C* code. We were able to obtain the training time of 12.4 seconds when $T = 40,000$.

The bottleneck of our method is at computing the weighted class-conditional sample

covariance matrices, which is expected at time $O(Nd^2)$. However, by reordering the terms carefully, a covariance matrix can be computed by multiplying a matrix with its transpose, as shown below.

For shorthand notations, we omit subscript $\mathbf{u}|c$ in the statistics in (4.17) and (4.18) and consider only the examples of class c , *i.e.*, we consider only indices n where $y_n = c$. Let:

$$\check{\mathbf{x}}_n = \sqrt{\frac{w_n}{z}} (\ddot{\mathbf{x}}_n - \mathbf{m}). \quad (4.19)$$

Now, consider multiplying matrix $\check{\mathbf{X}} = (\check{\mathbf{x}}_1, \check{\mathbf{x}}_2, \dots, \check{\mathbf{x}}_{N_c})^T$, where N_c is the number of examples belonging to class c , with its transpose:

$$\begin{aligned} \check{\mathbf{X}}\check{\mathbf{X}}^T &= \sum_{n:y_n=c} \check{\mathbf{x}}_n \check{\mathbf{x}}_n^T = \sum_{n:y_n=c} \frac{w_n}{z} (\ddot{\mathbf{x}}_n - \mathbf{m}) (\ddot{\mathbf{x}}_n - \mathbf{m})^T \\ &= \frac{1}{z} \sum_{n:y_n=c} w_n (\ddot{\mathbf{x}}_n \ddot{\mathbf{x}}_n^T - 2\ddot{\mathbf{x}}_n \mathbf{m}^T + \mathbf{m} \mathbf{m}^T) \\ &= \frac{1}{z} \left(\sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \ddot{\mathbf{x}}_n^T \right) - \frac{2}{z} \left(\sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \right) \mathbf{m}^T + \mathbf{m} \mathbf{m}^T \\ &= \frac{1}{z} \left(\sum_{n:y_n=c} w_n \ddot{\mathbf{x}}_n \ddot{\mathbf{x}}_n^T \right) - \mathbf{m} \mathbf{m}^T = \hat{\Sigma}, \end{aligned} \quad (4.20)$$

we get the sample covariance matrix. Matrix $\check{\mathbf{X}}$ has d rows and N_c columns. It took 1.82 seconds to compute $\check{\mathbf{X}}\check{\mathbf{X}}^T$ with $N_c = 5000$ and $d = 24^2$, by further relying on a highly optimized linear algebra package named GotoBLAS [23].

The second component of this method ran in only 2.1 seconds to compute all the statistics required by $T = 295,920$ features.

Figure 4.5: The ROC curves of three node classifiers with $M = 15$

4.3.3 Performance of Boosted Classifiers trained with Different Methods

We constructed three boosted classifiers of 15 weak classifiers: faster AdaBoost with $T = 40,000$ features, our method using the same feature set with faster AdaBoost, and our method with $T = 295,920$ features. By changing the controlling parameters, we obtained the ROC curves, illustrated in figure 4.5. The total amounts of time to train the boosted classifiers are: 180 seconds for faster AdaBoost, 75 seconds for our method with $T = 40,000$, and 90 seconds for our method with $T = 295,920$.

An analysis of the three ROC curves shows that the ROC curve of our method with $T = 40,000$ is slightly lower than the other two, while these two are similar. The results suggest that for small M - the number of weak classifiers in the ensemble, when using the same feature set, this method did not perform as well as AdaBoost. However, with more feature types the accuracy of this method increased with a minimal increase in the training time.

We constructed another three boosted classifiers but this time with 200 weak classifiers. The total amounts of time to train the boosted classifiers are: 2400 seconds for faster AdaBoost, 1000 seconds for our method with $T = 40,000$, and 1200 seconds for

Figure 4.6: The ROC curves of three node classifiers with $M = 200$

our method with $T = 295,920$. Similarly, the ROC curves are illustrated in figure 4.6.

From the ROC curves, the performance of faster AdaBoost and that of our method with $T = 40,000$ are relatively similar. However, our method with $T = 295,920$ outperformed the other two convincingly. The results indicate that it is beneficial to add more feature types, especially even when the problem becomes harder. The same observation was drawn in [45].

In addition, figures 4.7 and 4.8 illustrate the histograms (with 32 bins) of the class-conditional feature value distributions presented to the best feature classifiers, when training weak classifiers 1 and 200 respectively, using our method with $T = 295,920$. They show that these distributions approximated Gaussian distributions. Thus, our Gaussian assumption was reasonable.

4.3.4 Final Performance

We constructed three cascades: the first cascade used the faster AdaBoost implementation with $T = 40,000$ features, the second cascade used our method utilizing the same feature set of the first one, the final cascade used our method with $T = 295,920$. The three cascades had 30, 30, and 25 layers respectively. We stopped training the cascades when observing that the next ensemble (of usually 200 weak classifiers) could not re-

Figure 4.7: Histograms of the class-conditional feature value distributions presented to the best feature classifier when training weak classifier 1 using our method with $T = 295,920$.

duce the total false acceptance rate by less than 5% while maintaining the detection rate at above 99.9%. The ROC curves are illustrated in figure 4.10.

The performances of the first and the second cascade in the ROC curves show that both cascades performed relatively similar. However, the total training time of this method was significantly reduced (see table 4.1). In our experiment, we saw a gaining factor of 2.5 in training speed.

Moreover, when training the third cascade, we saw a reduction of layers due to better performance in every node. The ROC curves show that the third cascade out-performed the faster AdaBoost implementation in accuracy, and was trained at 2.4 times faster due to fewer layers but slightly longer training time per weak classifier.

In addition, the faster AdaBoost implementation required 800MB of memory to store

Figure 4.8: Histograms of the (weighted) class-conditional feature value distributions presented to the best feature classifier when training weak classifier 200 using our method with $T = 295,920$.

all the intermediate values, while our method only needed 30MB of memory for the same purpose.

4.4 Relation to Other Methods

Recently, there have been proposals to replace Discrete AdaBoost with Real AdaBoost, and use real-valued weak classifiers instead of thresholding [32, 87]. Their approach differs to ours in approximating the class-conditional distributions: they use histograms while we use Gaussian assumptions. Although we experimented with discrete-valued weak classifiers, it is possible to produce real-valued weak classifiers compatible to their

Figure 4.9: Some results on the MIT+CMU test set

Method	Total training time	Memory requirement
Faster AdaBoost [86] with $T = 40,000$	13h20m	800MB
Our method with $T = 40,000$	2h13m	30MB
Our method with $T = 295,920$	3h02m	30MB

Table 4.1: The time and the memory requirement for training the three experimented cascades.

approach by returning

$$\eta_t(v_t) = 0.5 \log \frac{\mathbb{P}[v_t|y = 1]}{\mathbb{P}[v_t|y = -1]} \quad (4.21)$$

instead of ± 1 . However, while Discrete AdaBoost is strongly resistant to overfitting [5], it is not known if the same argument holds for Real AdaBoost. The choice of the number of bins B raises another concern. The larger B is, the less accurate the per-bin values are estimated, and the more overfitted the weak classifier becomes.

The FFS method proposed by Wu *et al.* [83] was reported about 2.5 to 3.5 times faster than the faster AdaBoost implementation empirically. However, FFS does not maintain any weight distribution at every round, and is not directly related to boosting. On the other hand, this method solely relies on the weighting scheme of boosting to boost up the performance. Due to the fundamental difference, we did not include experiments on FFS.

Our method shares some analogy with Online Boosting [56, 60]: the proposed weak

Figure 4.10: The ROC curves on the MIT+CMU test set of the three experimented cascades.

Method	No of face examples	No of features	No of weak classifiers	No of nodes
Viola-Jones [81]	9,500	40,000	4,297	32
FloatBoost [44]	6,000	-	2,546	-
Nested Cascade + LUT [87]	20,000	-	756	16
Sparse Granular Features [32]	30,000	-	-	-
Faster AdaBoost [86]	5,000	40,000	3,870	30
Our method	5,000	295,920	3,502	25

Table 4.2: The reported training results of recent methods.

classifier can be used as an online learner. Unfortunately, using this method along with Online Boosting gains little benefit. When updating a weak classifier with one new example, one has to update all T Haar-like features. The time complexity of Online Boosting with or without this method is still the same: $O(T)$. Nevertheless, if we learn incrementally a block of examples instead of a single individual example, the training time can be significantly reduced.

In section 4.2.2, we proved that Haar-like features are simply linear projections of the probe image sub-window. Though this chapter addresses Haar-like features, this method is clearly applicable to any local features that are linear projections of a random vector,

Method	Time to train a weak classifier	Total training time	CPU speed
Viola-Jones [81]	> 10m	weeks	400MHz
FloatBoost [44]	-	weeks	700MHz
Nested Cascade + LUT [87]	-	weeks	1.4GHz
Sparse Granular Features [32]	< 1m	2 days	3.0GHz
Faster AdaBoost [86]	12.4s	13h20m	2.8GHz
Our method	5.74s	5h30m	2.8GHz

Table 4.3: The reported training time of recent methods.

like the ones used in [32, 44, 45].

We summarize the reported training time of recent methods and ours in table 4.2 and table 4.3.

4.5 Conclusions

In this chapter, we presented a fast method to train and select Haar-like features for the training of a weak classifier in face detection cascades using boosting. It trains a weak classifier in time $O(N + T)$ (in seconds), rather than $O(NT \log N)$ (in minutes) in traditional methods. In comparison with the currently fastest implementation of boosting [86], taking about 12 seconds per weak classifier using 40,000 Haar-like features but requiring almost 1GB of memory, this method trained a weak classifier using nearly 300,000 Haar-like features in 6 seconds, requiring less than 30MB of memory and maintaining a compelling performance. Besides, it suffered a very minimal increase in training time with very large increases in members of Haar-like features, enjoying a significant gain in accuracy. By substantially reducing the training time, this method empowers researchers to much more quickly experiment and explore solutions to other important research issues in this area.

Chapter 5

Multi-exit Asymmetric Boosting

5.1 Introduction

Cascading boosted classifiers has been a successful approach in appearance-based detection since the seminal work of Viola and Jones [79] in face detection. The key insight of building a cascade is to decompose a detection problem into a sequence of binary classification sub-problems of increasing difficulties. Positively predicted examples from the boosted classifier for a sub-problem are used to train the boosted classifier for the next sub-problem, while negatively predicted examples are discarded. The final cascade obtained from this bootstrapping process often has a high detection rate and an extremely low false acceptance rate, while the early rejection mechanism allows the cascade to scan through a large set of examples in a small amount of time.

Despite its efficiency in detection, the cascade approach imposes a number of issues in learning. At the stage level, one has to devise a learning strategy to find an optimal trade-off among three important factors of a boosted classifier: the detection rate, the false acceptance rate, and the number of weak classifiers. To maintain a high detection rate and an extremely low false acceptance rate for the overall cascade, each individual boosted classifier must ensure a close-to-one detection rate and a moderate false ac-

ceptance rate. It is essential to minimize the number of weak classifiers of a boosted classifier, as it is roughly proportional to the running time of the classifier. Conventional methods often use AdaBoost or one of its variants [21, 67] to train a boosted classifier with a fixed maximum number of weak classifiers. To find the best trade-off among the three factors, one has to re-train the classifier multiple times and choose the best candidate manually.

The overall detection rate of a cascade is the product of the detection rates associated with all the individual boosted classifiers in the cascade; similarly, the overall false acceptance rate is the product of all the classifiers' false acceptance rates. However, it is not known beforehand how many classifiers are needed, nor which combination of ROC operating points (each defined by a detection rate and a false acceptance rate) produces an optimal cascade. Currently, these parameters are obtained mainly by trial and error, though some progress has been made [9, 70].

In this chapter, we introduce a notion called multi-exit boosted classifier to describe a boosted classifier *with multiple exits*. Each exit is associated with a weak classifier. A rejection decision at an exit is made if the intermediate boosted score, up to the associated weak classifier, is below a threshold. Some cascade variants can be cast as a multi-exit boosted classifier. We analyze recent cascade training methods in terms of training goals, and show that many of them result in training and/or using sub-optimal weak classifiers. We propose a method to train a multi-exit boosted classifier by minimizing the number of weak classifiers needed to achieve the desired detection and false acceptance rates simultaneously. It also removes the need to run multiple ad hoc trials to discover the best boosted classifiers that satisfy the operating point requirements.

Our training method has a different objective than that of conventional training methods. In conventional methods, the number of weak classifiers is supposed to be fixed (*i.e.*, fixed resources), and the goal in training a boosted classifier is to minimize one type of error rates while constraining the other type. For instance, in [9], the goal is to maximize

the detection rate while keeping the false acceptance rate below a given level. Similarly, in [8], the goal is to minimize the false acceptance rate while keeping the detection rate above a certain level. In both cases and in many other methods (*e.g.*, [44, 89, 79, 83]), the number of weak classifiers are given *a priori*. Our training goal, on the other hand, constrains the desired error rates (*i.e.*, detection rate and false acceptance rate) and minimizes the number of weak classifiers. Such a training goal is more useful than the former two in practice, because the error rates are much more desirable than the amount of resources needed (*i.e.*, the number of weak classifiers). With the former goals, one has to train a boosted classifier multiple times until he/she finds one that matches his/her desired error rates. With the new training goal, this practice is avoidable, as shown in what follows in this chapter.

The remaining parts of the chapter are organized as follows. In section 5.2, we define a multi-exit boosted classifier and cast both the normal cascade and the regular boosted classifier as special cases. An analysis of recent cascade training methods is also offered in this section. In section 5.3, we describe our method to train a multi-exit boosted classifier, and discuss about important aspects in designing the method. Experimental results are presented in section 5.4.

The key contributions of the chapter are:

- a generalized model that unifies existing models such as conventional boosted classifiers, cascades, and more recent multi-exit boosted classifiers;
- a new multi-exit asymmetric boosting method incorporating asymmetric training goals to achieve ROC operating point targets while minimizing the number of weak classifiers;
- a principled formulation of an asymmetric goal that is theoretically optimal for a conservative bound on an ROC operating point requirement, and empirically near-optimal for the exact bound; and

- results demonstrating that the combined framework outperforms existing methods.

5.2 Overview

5.2.1 General Framework

In this section, we introduce a new model of which regular cascades and multi-exit boosted classifiers are defined as special cases. In section 5.2.2, we use the model to point out disadvantages of existing methods.

We restrict ourselves to a problem of imbalanced binary classification $C : \mathcal{X} \rightarrow \mathcal{Y} = \{-1, 1\}$ where the prior probability of the negative class far outweighs the prior probability of the positive class, *i.e.*, $\mathbb{P}[y = 1] \ll \mathbb{P}[y = -1]$ with y being the class. We consider the following model:

$$C(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } F_m(x) = 1 \quad \forall m \in \mathcal{M} \\ -1 & \text{otherwise} \end{cases} \quad (5.1)$$

$$F_m(x) \stackrel{\text{def}}{=} \text{sign}(f_m(x) - \theta_m) \quad (5.2)$$

$$f_m(x) \stackrel{\text{def}}{=} \sum_{i=\mu(m)}^m \tilde{h}_i(x), \quad (5.3)$$

Note that the dimensionality of input is not important in this chapter. Hence we adopt the normal letter x for simplicity. In this model, there are M weak classifiers denoted in sequence by $\tilde{h}_i(x)$ with $i = 1, \dots, M$, where $\tilde{h}_i : \mathcal{X} \rightarrow \mathbb{R}$; in the case of discrete-type weak classifiers, $\tilde{h}_i(x) = c_i h_i(x)$ with $h_i : \mathcal{X} \rightarrow \mathcal{Y}$ and coefficient $c_i \in \mathbb{R}$. Out of these M weak classifiers, we specify a subset of weak classifiers that acts as *exit nodes*, represented by a set of indices $\mathcal{M} \subset \{1, \dots, M\}$. We assume that the last classifier is always an exit node, hence $M \in \mathcal{M}$. Associated with each exit node is a corresponding *entrance node*, represented through the function $\mu(m)$ which is an index to a weak classifier earlier in the sequence. The boosted classifier comprising the weak classifiers

between a pair of entrance and exit nodes is associated with $F_m(x)$. Note that while each exit node is a unique weak classifier, entrance nodes may be shared (*i.e.*, they map to the same weak classifier).

This model is general in that it encompasses a range of existing and new models, *e.g.*, (a) the normal boosted classifier in this model is simply a single-exit boosted classifier utilizing all the weak classifiers, defined by $\mathcal{M} = \{M\}$ and $\mu(m) = 1$ for all $m \in \mathcal{M}$; and (b) the conventional cascade is represented by defining $\mu(m) = m_0 + 1$ where m_0 is the largest index in $\mathcal{M}$ satisfying $m_0 < m$, or $m_0 = 0$ if m is already the smallest index in $\mathcal{M}$. Conventional cascades as expected have entrance-exit intervals that are non-overlapping.

The variant that we explore in this chapter is the single boosted classifier with a single entrance but *multiple exits*. This model is characterized by $\mu(m) = 1$ and $|\mathcal{M}| > 1$ – this relates to multiple exit nodes sharing the same entrance node at the first weak classifier. A special case of the multi-exit boosted classifier is the soft/dynamic cascade [8, 88] in which $\mathcal{M} = \{1, \dots, M\}$. Other more complex variants exist that await future analysis.

In learning the m -th weak classifier for function $C(x)$ in (5.1) using AdaBoost or one of its variants [21, 67], the most important factor is the weight vector $\mathbf{w}_m$ associated with the training set provided to the weak classifier. It is often possible to express $\mathbf{w}_m$ in the form

$$w_{m,n} = Z_m^{-1} \exp(-y_n(s_m(x_n) - b_m)), \quad (5.4)$$

where Z_m is the normalization factor to make $\mathbf{w}_m$ a discrete distribution, b_m is a threshold to adjust the trade-off between the false acceptance rate and the false rejection rate when training the m -th weak classifier, and $s_m(x_n)$ is a *score function* of input point x_n , defined as:

$$s_m(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } m = 1, \text{ or } \mu(m) = m \\ f_{m-1}(x) & \text{otherwise} \end{cases} \quad (5.5)$$

In the original versions of AdaBoost [67], there are no thresholds b_m . It is known from literature [21, 67] that in such cases, the minimizer $\tilde{h}_m(x)$ of the classification error of the weighted training set $\{(x_n, y_n, w_{m,n})_{n=1}^N\}$ is approximately the minimizer of an (empirical) exponential loss:

$$\hat{\mathbb{E}} \left[\exp(-y(s_m(x) + \tilde{h}(x))) \right], \quad (5.6)$$

where $\hat{\mathbb{E}}$ is the expectation *w.r.t.* the empirical distribution $\hat{\mathbb{P}}$ given in (3.6):

$$\hat{\mathbb{P}} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N \delta_{(x_i, y_i)}. \quad (5.7)$$

By introducing $b_m \neq 0$, the exponential loss becomes:

$$\hat{\mathbb{E}} \left[\exp(-y(s_m(x) + \tilde{h}(x) - b_m)) \right], \quad (5.8)$$

and is an upper bound of *an asymmetric error* [80]:

$$\begin{aligned} &= \hat{\mathbb{P}}[y = -1] \exp(-b_m) \hat{\mathbb{E}} \left[\exp \left((s_m(x) + \tilde{h}(x)) \right) | y = -1 \right] \\ &\quad + \hat{\mathbb{P}}[y = 1] \exp(b_m) \hat{\mathbb{E}} \left[\exp \left(-(s_m(x) + \tilde{h}(x)) \right) | y = 1 \right] \\ &\geq \hat{\mathbb{P}}[y = -1] \exp(-b_m) FAR(s_m + \tilde{h}) \\ &\quad + \hat{\mathbb{P}}[y = 1] \exp(b_m) FRR(s_m + \tilde{h}), \end{aligned} \quad (5.9)$$

where $FAR(f) = \hat{\mathbb{E}}[1_{[f(x) \geq 0]} | y = -1]$ and $FRR(f) = \hat{\mathbb{E}}[1_{[f(x) < 0]} | y = 1]$ are the false acceptance rate and the false rejection rate respectively of a function $f(x)$.

5.2.2 Related Work

In their original work [79], Viola and Jones sampled N_1 positive examples and N_2 negative examples, and initially set weights $0.5/N_1$ for positive examples and $0.5/N_2$ for

negative examples, respectively. Effectively, this removed the prior probabilities not only from training the first weak classifier, but also from training subsequent weak classifiers. Since they did not use any threshold b_m , the goal at training the m -th weak classifier became finding a function $\tilde{h}$ that minimizes an upper bound of:

$$FAR(s_m + \tilde{h}) + FRR(s_m + \tilde{h}). \quad (5.10)$$

This symmetric goal, however, did not guarantee a high detection rate for a boosted classifier. Viola and Jones proposed to threshold the boosted classifier's score $f_m(x)$ ($m \in \mathcal{M}$) by a value $\theta_m \neq 0$. By adjusting θ_m , they were able to achieve a high detection rate (and also a high false acceptance rate). Many recent methods followed this strategy [8, 9, 44, 48, 70, 83, 88, 89]. While this approach is computationally trivial, it is also sub-optimal in that the θ_m -defined ROC operating point was not arrived at by training the weak classifiers with the asymmetric goal in (5.8)¹.

In a follow-up paper [80], Viola and Jones addressed this problem by introducing an asymmetric goal for training the m -th weak classifier of a boosted classifier, which is to minimize the following asymmetric error *w.r.t.* $\tilde{h}$:

$$FAR(s_m + \tilde{h}) + \lambda FRR(s_m + \tilde{h}), \quad (5.11)$$

where λ is related to b_m by:

$$\lambda = \exp(b_m) / \exp(-b_m) \quad (5.12)$$

$$\Leftrightarrow b_m = 0.5 \log \lambda. \quad (5.13)$$

There were no suggestions on how λ should be selected. While in chapter 3, we proposed

¹Consequently the ROC curve obtained by varying θ_m upper-bounds the error expressed by the “proper” ROC curve obtained by varying b_m .

a skewness balancing method for selecting λ , it is required that the number of weak classifiers be known in advance.

Xiao *et al.* [89] and Wu *et al.* [83] independently noted that scores obtained from the previous boosted classifier may be exploited downstream, and proposed to propagate the previous scores from one boosted classifier to the next. Though their weak classifiers were trained using the symmetric goal in (5.10), they showed significant improvements in their empirical results. Nevertheless, their cascades may be considered as initial boosted classifiers with multiple exits.

Bourdev and Brandt [8] trained a very long boosted classifier using the symmetric goal and subsequently utilized a calibration algorithm to break this strong classifier into a cascade, through a rejection threshold at every weak classifier. One interesting point in their method for training a long boosted classifier is that at each iteration, they introduced hard negative examples to the training set, making the training process biased towards favoring the positive class. However, because the thresholds were obtained *after* all the weak classifiers were trained, the weak classifiers became sub-optimal *w.r.t.* the goal of the cascade. Xiao *et al.* [88] corrected this by updating the training set *before* training each weak classifier. Still, the symmetric goal was employed by both these methods to train weak classifiers, resulting in sub-optimal feature selection. While the approach of making decisions at each weak classifier is potentially interesting, a number of issues are raised. First, bootstrapping is required at every weak classifier, incurring significant extra computational cost. Second and more importantly, the requirement for early rejection diminishes as progress is made down the cascade, where the problem becomes dominated by accuracy (as classification becomes harder) as opposed to speed (since most of the obvious negative samples will have already been rejected earlier). In such cases, having a decision made at each weak classifier effectively discards important information that may have been exploited if decisions were postponed until further downstream, leading to less accurate classification.

Another problem with the methods above is that the number of weak classifiers per boosted classifier must be specified prior to training. This number is important because it is a trade-off between a better ROC operating point and a shorter running time for a boosted classifier. Tuning the number in an ad hoc fashion is difficult and time consuming. Our approach avoids this by choosing a proper asymmetric goal that greedily minimizes the number of weak classifiers to reach the target detection rate and false acceptance rate simultaneously.

5.3 Multi-exit Asymmetric Boosting

From the general framework provided in section 5.2.1, it is clear that a multi-exit boosted classifier is more complex than a regular cascade, because in the former all weak classifiers are used to form a single boosted classifier, while in the latter the weak classifiers are grouped into a number of segments. In what follows, we propose a method, called multi-exit asymmetric boosting, to train a multi-exit boosted classifier using an asymmetric learning goal (algorithm 6).

Again, throughout this chapter, instead of using Viola and Jones’ asymmetric goal, we use its normalized version defined in (2.7):

$$\mathcal{J}_\lambda(F_m) \stackrel{\text{def}}{=} \frac{1}{\lambda + 1} \mathbb{P}[F_m(x) = 1 | y = -1] + \frac{\lambda}{\lambda + 1} \mathbb{P}[F_m(x) = -1 | y = 1]. \quad (5.14)$$

5.3.1 Overview of the Method

Unlike conventional cascades, a multi-exit boosted classifier may be considered as a collection of boosted classifiers that *share* overlapping sets of weak classifiers. Training multi-exit boosted classifiers thus entails a number of added complexities. Similar to Xiao *et al.* [89], we incorporate bootstrapping and discrete-valued asymmetric boosting into a single method. A slightly modified variant of asymmetric boosting is summarized

Algorithm 6 Multi-exit Asymmetric Boosting**Require:**

- a generator G that produces *i.i.d.* training examples
- maximum false acceptance rate α_0
- maximum false rejection rate β_0
- a stopping condition D {see section 5.3.2}

- 1: $M_0 = m = 0$
- 2: $\mathcal{M} = \emptyset$
- 3: $\mathcal{H} = \emptyset$ {list of parameters of C }
- 4: $\lambda = \alpha_0/\beta_0$ {see section 5.3.3}
- 5: **repeat**
- 6: Generate $\mathcal{Z} = \{(x_n, y_n)\}_{n=1}^{N_1+N_2}$ of N_1 positive examples and N_2 negative examples such that $f_m(x_n) \geq 0 \quad \forall m \in \mathcal{M}, n \in \{1, \dots, N_1 + N_2\}$
- 7: $(m, \mathcal{H}) = \text{Train_Segment}(m, \mathcal{H}, \mathcal{Z})$ {see algorithm 7}
- 8: **if** not D **then**
- 9: $M_0 = m$
- 10: $\mathcal{M} = \mathcal{M} \cup \{m\}$
- 11: **end if**
- 12: **until** D
- 13: **return** $\mathcal{M}$ and $\mathcal{H}$ as the list of parameters of C

in algorithm 7.

We use the same strategy as conventional methods [79, 80] in designing the target minimum detection rate and maximum false acceptance rate at every exit node. That is, we are interested in considering a rejection decision only when the false acceptance rate is below α_0 and the detection rate over $1 - \beta_0$, where α_0 and β_0 are predefined as part of the cascade design. The problem of finding the optimal combination of desired rates for all exit nodes are not addressed in this chapter, and it is assumed that (α_0, β_0) is provided for every exit node.

5.3.2 Bayes Asymmetric Error

In traditional classification methods, if the two positive and negative classes are not separable, *i.e.*, if the class of an input point is not deterministic, that is:

$$\mathbb{P}[y|x] \notin \{0, 1\} \tag{5.15}$$

Algorithm 7 $Train_Segment(m, \mathcal{H}, \mathcal{Z})$

-
- 1: Initialize weights: {see section 5.3.4}

$$w_n = e^{-(s_m(x_n) - 0.5 \log \lambda) / N_1} \quad \forall n : y_n = 1$$

$$w_n = e^{(s_m(x_n) - 0.5 \log \lambda) / N_2} \quad \forall n : y_n = -1$$
 - 2: **repeat**
 - 3: $m = m + 1$
 - 4: Normalize weights w_n
 - 5: Train $h_m(x)$ using $\{(x_n, y_n, w_n)\}_{n=1}^{N_1+N_2}$
 - 6: Find the weighted training error of h_m :

$$e_m = \sum_{n=1}^{N_1+N_2} w_n 1_{[y_n \neq h_m(x_n)]}$$
 - 7: Compute the corresponding coefficient:

$$c_m = 0.5 \log \frac{1-e_m}{e_m}$$
 - 8: $\mathcal{H} = \mathcal{H} \cup \{c_m, h_m\}$
 - 9: Update weights: {see section 5.3.4}

$$w_n = w_n e^{-y_n c_m h_m(x_n)} \quad \forall n$$
 - 10: Estimate the training error rates:

$$\alpha_m = \frac{1}{N_2} \sum_{n: y_n = -1} 1_{[f_m(x_n) \geq 0]}$$

$$\beta_m = \frac{1}{N_1} \sum_{n: y_n = 1} 1_{[f_m(x_n) < 0]}$$
 - 11: **until** D or $(\alpha_m \leq \alpha_0 \wedge \beta_m \leq \beta_0)$
 - 12: **return** $(m, \mathcal{H})$
-

for some x , then there exists a limit called *the Bayes error* such that no classification error can be smaller than it. Similarly in asymmetric boosting or any method that uses an asymmetric error as an objective function, there is also a lower-bound of the asymmetric error.

The proof for the existence of such a lower-bound is analogous to the traditional analysis of the Bayes error. Given an input point x , a classifier either predicts the output as -1 or 1. In either case, the asymmetric error of classifying x is always not less than

$$\min \left\{ \frac{1}{\lambda + 1} \mathbb{P}(y = -1|x), \frac{\lambda}{\lambda + 1} \mathbb{P}(y = 1|x) \right\}. \quad (5.16)$$

Marginalizing this quantity over the distribution of X , we get an amount that no classifier can produce an asymmetric error smaller than it. We define it as the *the Bayes*

asymmetric error:

$$\epsilon_\lambda \stackrel{\text{def}}{=} \int_x \min \left\{ \frac{1}{\lambda + 1} \mathbb{P}(y = -1|x), \frac{\lambda}{\lambda + 1} \mathbb{P}(y = 1|x) \right\} \mathbb{P}(x) dx. \quad (5.17)$$

As the sub-problem gets harder, it is possible that the desired rates may not be achievable. Therefore, a stopping condition D is needed. In our experiments, we stopped training when the number of weak classifiers between two exit nodes exceeds 200. Conventional methods [79, 80] also used at most 200 weak classifiers to train a boosted classifier.

5.3.3 Analysis of Asymmetric Goals

The main contribution of our approach is to select a proper asymmetric goal that makes asymmetric boosting greedily minimize the number of weak classifiers to achieve a detection rate over $1 - \beta_0$ and a false acceptance rate below α_0 concurrently. Before analyzing how the choice of an asymmetric goal affects the final boosted classifier, let us assume for the moment that AdaBoost minimizes an asymmetric goal instead of its exponential upper bound². For notational simplicity, we denote the false rejection rate and the false acceptance rate of a classifier f_m as $\beta(f_m)$ and $\alpha(f_m)$ respectively. In the conventional methods, one often trains a boosted classifier using AdaBoost with the symmetric goal in (5.10). If we plot all the training ROC operating points $(\beta(f_m), \alpha(f_m))^T$ with an increasing number of weak classifiers $m = 1, 2, \dots$ we obtain a set of points in figure 5.1a (shown in red).

In AdaBoost and many of its variants [21, 67], when an m -th weak classifier is trained, all other weak classifiers are considered fixed. The proper ROC curve of the new boosted classifier is simply the locus of all operating points of the boosted classifier

²In practice, because the minimization is applied on the exponential upper bound, the value of the true goal tends to exhibit a locally oscillating but globally decreasing behavior *w.r.t.* the number of weak classifiers.

Figure 5.1: Illustration of how the operating point of a boosted classifier moves when more weak classifiers are trained. Blue solid curves are equivalent to the ROC curves of the boosted classifier. (a): A case when the symmetric goal G_1 is used, *i.e.*, $\lambda = 1$. (b): A case when an asymmetric goal G_λ is used with $\lambda = \alpha_0/\beta_0$.

obtained through training the m -th weak classifier with different *asymmetric goals* (as opposed to simply varying the threshold on the symmetric goal function). Because of the immense computational effort involved, the proper ROC curve is seldom obtained in entirety for training purposes; instead only a single operating point is derived for each weak classifier. While most of the proper ROC curve remains unknown, a number of ROC curve properties may be inferred as described below.

Let the unknown training ROC curve be denoted by Q_m and expressed in the form

$$L_m(\beta, \alpha) = 0 \quad (5.18)$$

with each instance of (β, α) that satisfies (5.18) representing an ROC operating point trained with a different asymmetric goal. In this chapter, we make the assumption that Q_m is first-order continuous. Equation (5.14) representing an asymmetric error may be rewritten as

$$G_\lambda(\beta, \alpha) \stackrel{\text{def}}{=} \frac{\lambda}{\lambda + 1}\beta + \frac{1}{\lambda + 1}\alpha, \quad (5.19)$$

with a controlling parameter λ . When using G_λ to train the m -th weak classifier, it becomes:

$$J_{m,\lambda}(\tilde{h}) = G_\lambda(\beta(s_m + \tilde{h}), \alpha(s_m + \tilde{h})) \quad (5.20)$$

$$= \frac{\lambda}{\lambda + 1} \beta(s_m + \tilde{h}) + \frac{1}{\lambda + 1} \alpha(s_m + \tilde{h}) \quad (5.21)$$

We assume the resulting weak classifier $\tilde{h}_m$ minimizes $J_{m,\lambda}(\tilde{h})$ globally:

$$\tilde{h}_m \stackrel{\text{def}}{=} \arg \min_{\tilde{h}} J_{m,\lambda}(\tilde{h}). \quad (5.22)$$

As in conventional methods, the boosted classifier is updated via

$$f_m = s_m + \tilde{h}_m. \quad (5.23)$$

Training the weak classifier can be cast as a constrained minimization problem *w.r.t.* (β, α) , with objective function $G_\lambda(\beta, \alpha)$, and a constraint $L_m(\beta, \alpha) = 0$. Thus, at any solution point (if it exists) lying on Q_m , the gradient of goal G_λ (*i.e.*, ∇G_λ) has to be perpendicular to the ROC curve Q_m . Note that ∇G_λ is independent of β and α once λ is fixed.

By using the symmetric goal G_1 (*i.e.*, $\lambda = 1$) as in conventional methods, one effectively chooses an operating point on Q_m at which the gradient is perpendicular to

$$\nabla G_1 = (0.5, 0.5)^T. \quad (5.24)$$

If instead we choose an asymmetric goal G_λ for some $\lambda \geq 0$, we recover a different operating point on Q_m with gradient perpendicular to

$$G_\lambda = \left(\frac{\lambda}{\lambda + 1}, \frac{1}{\lambda + 1} \right)^T. \quad (5.25)$$

By varying λ from 0 to ∞ and retraining the weak classifier each time, we can reconstruct the ROC curve Q_m . Though we have no knowledge of Q_m prior to training a weak classifier, we can somewhat control where the next operating point might be located, by controlling λ .

If we further assume that Q_m is not only first-order continuous but smoothly varies as λ is changed, then a key observation is that Q_m *must be convex*. This is because as λ is varied from 0 to ∞ , the gradient ∇G_λ monotonically changes from vertical to horizontal uniformly throughout the α - β space. Convexity is guaranteed as the operating point (β, α) for a particular λ must not only be on a point of Q_m perpendicular to ∇G_λ , but also be the global minimum of G_λ .

Another important point is, if we use the same λ to train every weak classifier of a boosted classifier, convergence is guaranteed due to iterative decreases in values of a *single* goal function. If we use different goals (*i.e.*, different λ) to train different weak classifiers, it is possible that when a goal function decreases, other goal functions increase. Hence, no convergence is guaranteed. Despite this, previous work shows empirically reasonable results when different goals are used for different weak classifiers [60, 80], but they require that the number of weak classifiers be specified before training. Therefore, to ensure convergence and also to avoid the need to specify *apriori* the number of weak classifiers to be trained, we use the same λ to train every weak classifier of a boosted classifier.

Generally, the training process should be stopped once we reach an operating point in rectangle $O A E B$ in figure 5.1, which represents an exact bound ensuring the detection rate over $1 - \beta_0$ and the false acceptance rate below α_0 . However, let us focus our attention on a more conservative but easier-to-analyze bound, expressed by the triangle $O A B$. This triangle has an interesting property as follows.

Theorem 5.3.1. *If Q_m penetrates triangle $O A B$ for some m , then the use of goal G_{λ^*} with $\lambda^* = \alpha_0 / \beta_0$ ensures that point $(\beta(f_m), \alpha(f_m))^T$ is located inside the triangle.*

Proof. Because Q_m is first-order continuous and intersects with the line segment AB at two locations, there exists a point (β^*, α^*) on Q_m inside triangle OAB at which the gradient of Q_m is parallel to AB . Alternatively:

$$\nabla L_m(\beta^*, \alpha^*) \parallel (\alpha_0, \beta_0)^T. \quad (5.26)$$

If we use $\lambda^* = \alpha_0/\beta_0$ to train the m -th weak classifier, we obtain f_m such that:

$$\nabla L_m(\beta(f_m), \alpha(f_m)) \parallel \left(\frac{\alpha_0/\beta_0}{\lambda^* + 1}, \frac{1}{\lambda^* + 1} \right)^T = \left(\frac{\alpha_0}{\alpha_0 + \beta_0}, \frac{\beta_0}{\alpha_0 + \beta_0} \right)^T. \quad (5.27)$$

Because Q_m is convex, $(\beta(f_m), \alpha(f_m))$ is a unique point. Since

$$(\alpha_0/\beta_0, 1)^T \parallel \left(\frac{\alpha_0}{\alpha_0 + \beta_0}, \frac{\beta_0}{\alpha_0 + \beta_0} \right)^T, \quad (5.28)$$

these two points must coincide:

$$(\beta(f_m), \alpha(f_m)) = (\beta^*, \alpha^*). \quad (5.29)$$

□

Theorem 5.3.1 states that even though the next ROC curve is unknown, if it intersects with triangle OAB , then by using goal $\lambda^* = \alpha_0/\beta_0$, a stopping point is found in no more than one single iteration. This is an absolute guarantee. Other choices of λ may result in one single iteration, but the guarantee is not absolute.

Furthermore, if the ROC curve does not intersect with triangle OAB , we still have a guarantee that the newly found operating point minimizes function G_{λ^*} . Since $\nabla G_{\lambda^*} \parallel (\alpha_0, \beta_0)^T$, it means the newly found operating point is the one nearest to the line containing segment AB in Euclidean distance. This observation is summarized in the following theorem, which we state without proof:

Theorem 5.3.2. *If Q_m does not penetrate triangle OAB for some m , then the use of goal G_{λ^*} with $\lambda^* = \alpha_0/\beta_0$ ensures that the newly found operating point is the one nearest to the line containing segment AB in Euclidean distance.*

Therefore, we use $\lambda = \lambda^*$ to train every weak classifier of a boosted classifier. In theory, the line containing segment AB is $\lambda^*\beta + \alpha = \alpha_0$, so we can use

$$J_{m,\lambda^*}(\tilde{h}_m) \leq \frac{\alpha_0}{\lambda^* + 1} = \frac{\alpha_0\beta_0}{\alpha_0 + \beta_0} \quad (5.30)$$

as the stopping condition. In practice, we use the original stopping condition: $\alpha(f_m) \leq \alpha_0$ and $\beta(f_m) \leq \beta_0$.

Experiments in section 5.4.2 verify our analysis.

5.3.4 Updating Weights

An important advantage in our method is that every weak classifier is trained with the same asymmetric goal as the boosted classifier. Therefore, the number of weak classifiers needed to reach the desired rates is greedily the shortest. In addition, no threshold adjustment needs to be applied to a boosted classifier as in many previous methods. Conversely, the use of a threshold would mean that the asymmetric goal of the boosted classifier is different from that used in training the weak classifiers; hence the weak classifiers would have been sub-optimally trained *w.r.t.* the goal.

We use the same weight-updating rule proposed by Viola and Jones [80]. By simply multiplying the weights of every example (x_n, y_n) with $\sqrt{\lambda^{y_n}}$, before training the first weak classifier, the training goal of every weak classifier essentially becomes asymmetric: $FAR + \lambda FRR$. Notice that for every weak classifier, λ and b_m are related by $b_m = 0.5 \log \lambda$ as in (5.13).

5.4 Experimental Results

5.4.1 Setup

To justify the arguments we made in the previous section, we ran a few experiments. We collected 1521 face images from the BioID Face Database³, 508 face images from the AR Face Database [50], and about 4000 frontal face images used by Xiao *et al.* in [88]. Altogether we obtained about 6000 face images. We used a generator that selects known face locations randomly, and resizes them down to a base resolution of 24×24 pixels, with some perturbation bias to achieve a slightly more robust performance [70]. We used the same collection of a few thousand large images containing no faces used by Wu *et al.* in [86] to generate sufficient non-face sub-windows.

In order to speed up our comparisons, we implemented the technique for fast-training weak classifiers described in [59]. Nineteen types of Haar-like features were used, generating nearly 300,000 Haar features. We observed a training time of 4 seconds per weak classifier. This technique was uniformly applied to all methods compared in table 5.1.

For testing, we used the MIT+CMU test set which consists of 130 grayscale images with 507 frontal faces [65]. Post-processing was the same as in [79]. The experiments were done on a 3.2GHz Intel Pentium IV PC with 1 GB memory running Windows XP.

5.4.2 Training with Different Asymmetric Goals

To understand the relationship between choosing different λ 's for training and the resulting number of weak classifiers, we ran the following experiment. We trained many regular single-entrance single-exit boosted classifiers by varying only λ and the stopping bound (*i.e.*, either exact or conservative). All of these classifiers were trained using the same face training set with 5,000 examples per class, which had been filtered from a cascade of a few layers. This was to avoid a trivial case that a boosted classifier might

³<http://www.bioid.com/downloads/facedb/index.php>

Figure 5.2: Comparison of training a boosted classifier with different λ and stopping bounds

end up using too few weak classifiers. In such case, the significance in choosing λ might not be observable.

We set the desired rates to be: $\alpha_0 = 0.8$ for the false acceptance rate, and $\beta_0 = 0.01$ for the false rejection rate. Hence the choice of λ selected by our method was at $\lambda^* = 80$. The values of λ were selected from $\{p10^q : p \in \{1, 2, 5\} \wedge q \in \{-1, 0, 1, 2, 3, 4\}\}$. In addition, the values in $\{10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 100\}$ were also used. For each λ and each bound, we trained a boosted classifier and counted the numbers of weak classifiers obtained when the algorithm stopped. The results are plotted in two curves (each representing a bound) in figure 5.2.

From figure 5.2, the resultant curves are similar for $\lambda \leq 5$ or $\lambda \geq 100$. They differ at the range $5 \leq \lambda \leq 100$, where they both have low numbers of weak classifiers. Figure 5.3 illustrates an expanded view of this small range.

Figure 5.3: A closer view on the comparison of training a boosted classifier with different bounds and λ 's, where $10 \leq \lambda \leq 100$

Note that the classification problem we used for training these classifiers was relatively hard. This was shown in figure 5.2 at point $\lambda = 10^0 = 1$, at which the numbers of weak classifiers in the two curves were about 300. This point is equivalent to the symmetric goal used in conventional training. In compared with the numbers of weak classifiers obtained using λ near the “optimal” point (*i.e.*, $\lambda \approx 80$), they were about 10 times larger.

It is also understandable why the number of weak classifiers increased as λ approached 0 or ∞ . As $\lambda \rightarrow 0$, the desired false acceptance rate was achieved much faster than the desired false rejection rate. It took many iterations until the desired false rejection rate was achieved. Conversely, as $\lambda \rightarrow \infty$, the desired false rejection rate was achieved much faster than the desired false acceptance rate. It also took many iterations until the desired false acceptance rate was achieved. Therefore, the best choices for λ

would be those near 80.

In figure 5.3, we can see a difference in the number of weak classifiers between the two curves, even though these numbers are small. This is where the exact bound had an advantage over the conservative bound. With the conservative bound, an operating point has to be found inside triangle OAB in figure 5.1 before the training algorithm can stop. With the exact bound, the requirement is relaxed from triangle OAB to rectangle $OAEB$. Thus, in our experiment, the algorithm could stop earlier with the exact bound.

Also, figure 5.3 shows that the best empirical value for λ using the conservative bound was 80, which matched with λ^* chosen by our method.

5.4.3 Final Performance

We compared our method with original cascade [79] and asymmetric cascade [80] of Viola and Jones, boosting chain of Xiao *et al.* [89], nesting-structured cascade of Wu *et al.* [83], soft cascade of Bourdev and Brandt [8], and dynamic cascade of Xiao *et al.* [88]. To obtain free parameters for these methods, we used either their proposed best parameters, or obtained manually by trial and error. Each method requires a different non-empty set of free parameters, even though some of them claim to be fully automatic [8, 88]. Also to make the comparison as reasonable as possible, we simplified their versions by removing ideas that are reasonably irrelevant to the comparison. For example, in nesting-structured cascade, we replaced Real AdaBoost with Discrete AdaBoost. In dynamic cascade, we replaced their Bayesian stump with Discrete AdaBoost. We used the same type of Haar-like weak classifiers trained using the method proposed in chapter 4 for all the methods.

Table 5.1 shows some statistics obtained when training these methods. We report the total training times and the detection speeds based on the best cascades obtained for all the methods. The detection speeds were measured by running each cascade, without

Method	No of weak classifiers	No of exits	Total training time	Detection speed
Viola-Jones [79]	4,297	32	6h20m	4.4fps
Asym cascade [80]	3,502	29	4h30m	8.2fps
Boosting chain [89]	959	22	2h10m	7.8fps
Nested cascade [83]	894	20	2h	8.1fps
Soft cascade [8]	4,871	4,871	6h40m	6.7fps
Dynamic cascade [88]	1,172	1,172	2h50m	6.2fps
Multi-exit boosting	575	24	1h20m	9.8fps

Table 5.1: Some key factors after training recent methods.

removing any boosted/weak classifier from it⁴, on the whole MIT+CMU test set. The MIT+CMU test set is relatively challenging, with some very large images that required significant amounts of computation.

Among these methods, the first four methods had many free parameters thus requiring much more trials and errors to get the best results than the last three. The table shows that the methods that propagate scores (*i.e.*, boosting chain, nesting-structured cascade, dynamic cascade, and multi-exit asymmetric boosting) resulted in significantly smaller numbers of weak classifiers. Among these four methods, multi-exit asymmetric boosting gave the smallest number of weak classifiers, less than two third that of the second best, *i.e.*, nesting-structured cascade.

Figure 5.4 shows different ROC curves of these methods. There are a few interesting points that can be understood from these ROC curves.

Viola and Jones’ original cascade performed well, but was superseded by recent methods. Soft cascade turned out to be on a par with dynamic cascade. There might be a reason for this: even though dynamic cascade obtained better weak classifiers, it removed the “calibration” stage of soft cascade, which would have significantly improved the final performance of soft cascade.

Viola and Jones’ asymmetric cascade performed significantly better Viola and Jones’

⁴In practice, it is very easy to increase the detection speed by removing some stages of the cascade, at the cost of reducing accuracy.

Figure 5.4: Comparing different methods on the MIT+CMU test set

original cascade, but not as good as other methods. It performed better than dynamic cascade, boosting chain, and nesting-structured cascade at first, but became inferior when a lot more weak classifiers were trained. In the first few layers, training with an asymmetric goal indeed had an advantage over that with the symmetric goal. As the sub-problem became much harder as in the final layers of a cascade, the use of propagating the scores from previous boosted classifier became very useful.

Dynamic cascade performed worse than nesting-structured cascade and boosting chain. While dynamic cascade was trained automatically, boosting chain and nesting structured cascade was trained with key parameters manually selected. This result justified our argument in section 5.2.1: as the sub-problem becomes harder, it is better to train longer but more accurate boosted classifiers than making early but less accurate rejection predictions.

Nesting-structured cascade performed comparable to boosting chain. However, multi-

Figure 5.5: ROC curves reported on the MIT+CMU test set by recent methods

exit asymmetric boosting performed better than all previous methods. This was expected because not only multi-exit asymmetric boosting inherited the idea of asymmetric goals from asymmetric cascade, it also inherited the idea of propagating scores from one segment to another. Besides, the values of λ selected by our method often resulted in low numbers of weak classifiers and high accuracy, and were often similar to those manually chosen for asymmetric cascade [80]. Multi-exit asymmetric boosting avoided the need to specify the number of weak classifiers to train per segment, as well as to train a segment multiple times.

Figure 5.5 shows the final ROC curves reported by the recent methods in their original papers. Because each detector in the literature was trained with a different training set, and it was easy to tune a detector to achieve a good performance against a single test set, comparing these detectors is difficult. Dynamic cascade and nesting-structured cascade used a different type of weak classifiers than ours and previous methods. This may

Figure 5.6: Some results of our method on the MIT+CMU test set

explain why very good results in their methods were reported. However, we were unable to produce results as good as those reported by Bourdev and Brandt [8] for the soft cascade, despite that our implementation was very similar to theirs. Xiao *et al.* [88] also re-implemented soft cascade and reported that they could not reproduce results similar to those of Bourdev and Brandt.

5.5 Conclusions and Future Work

We proposed a method to train a multi-exit boosted classifier by combining the idea of propagating scores in [83, 89] and training with an asymmetric goal in [80]. We showed how to properly select an goal that achieves the desired error rates with a minimum number of weak classifiers, avoiding the need to run multiple ad hoc trials to discover

the best boosted classifiers that satisfy the operating point requirements. Experimental results showed not only significant reduction in training time and number of weak classifiers, but also better accuracy compared to conventional cascades and multi-exit boosted classifiers.

We did not address the problem of selecting the desired rates for all the exit nodes, leaving it as an open problem for future work. In the chapter, we used fixed desired rates for every exit node, similar to that of Viola and Jones [79, 80]. Also, our analysis of asymmetric goals might be the first, but future work should provide a deeper investigation into the issue.

Chapter 6

Margin-based Bounds on an Asymmetric Error

6.1 Introduction

Let (x, y) be a random couple defined on a probability space $(\mathcal{X} \times \mathcal{Y}, \Sigma, \mathbb{P})$, where x is an instance in an input space $\mathcal{X}$, $y \in \mathcal{Y} = \{-1, 1\}$ is an output class, Σ is a σ -algebra of subsets of $\mathcal{X} \times \mathcal{Y}$, and $\mathbb{P}$ is a true but *unknown* probability measure. Let $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to $\mathbb{R}$. For $f \in \mathcal{F}$, we use $\text{sign}(f(x))$ to predict the unknown class y . The standard goal of learning a classifier is to find a predictor $\hat{f} \in \mathcal{F}$ based on the training data to obtain a small *generalization error*:

$$\mathbb{P} \left[\text{sign}(\hat{f}(x)) \neq y \right] = \mathbb{P} \left[y\hat{f}(x) \leq 0 \right]. \quad (6.1)$$

In what follows, however, we consider an error which is different from the standard classification error, called *an asymmetric error*:

$$\mathcal{J}_\lambda(F) \stackrel{\text{def}}{=} \lambda_1 \mathbb{P} [F(x) = -1 | y = 1] + \lambda_2 \mathbb{P} [F(x) = 1 | y = -1], \quad (6.2)$$

where $F(x) = \text{sign}(f(x))$ is the binary classifier to be learned, and λ_1 and λ_2 are costs associated with the false rejection rate and false acceptance rate respectively. Note that the asymmetric error used in the previous chapters (*e.g.*, in (2.7)) correspond to a special case of the one we address here, by setting $\lambda_1 = \frac{\lambda}{\lambda+1}$ and $\lambda_2 = \frac{1}{\lambda+1}$.

In traditional classification learning methods, there is no difference between a false negative and a false positive, *i.e.*, they are equally penalized. The idea of using this asymmetric goal is to weigh false negatives and false positives differently so as to favor one type of error over the other. Traditional bounds are not suited to bound an asymmetric error because they do not take into account the fact that the two error rates are treated differently with different costs. Conversely, in an attempt to explain the performance of an asymmetric classifier, we propose in this chapter the first bounds of this type, which to our best knowledge, have not been addressed in the literature.

6.1.1 Related Work

Let $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ be a set of N *i.i.d.* labeled training examples sampled from the true but unknown joint probability distribution $\mathbb{P}$. The standard approach to bounding the standard generalization error was developed in seminal papers of Vapnik and Chervonenkis in the 70s and 80s (*e.g.*, see [15, 76, 78]). It is based on bounding the difference between the generalization error $\mathbb{P}[yf(x) \leq 0]$ and the training error $\hat{\mathbb{P}}_N[yf(x) \leq 0]$ uniformly over the whole function space $\mathcal{F}$ of classifiers f , where $\hat{\mathbb{P}}_N$ is the *empirical measure* based on the sample set S :

$$\hat{\mathbb{P}}_N \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^N \delta_{(x_i, y_i)} \quad \forall (x_i, y_i) \in S, \quad (6.3)$$

with $\delta_{(x,y)}$ being the probability distribution on $\mathcal{X} \times \mathcal{Y}$ concentrated at point (x, y) .

The bounds are frequently expressed in terms of the so-called VC-dimension of the function space. However, in many important examples, (*e.g.*, in boosting or in neural

network learning), the VC-dimension of the class can be very large that makes it impossible to apply the Vapnik–Chervonenkis type of bounds.

Since the invention of voting algorithms such as boosting, the convex hull $\text{conv}(\mathcal{H})$ of a given base function space $\mathcal{H}$ of functions $h : \mathcal{X} \rightarrow [-1, 1]$, defined as

$$\mathcal{F} = \text{conv}(\mathcal{H}) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^M c_i h_i : c_i \geq 0, \sum_{i=1}^M |c_i| = 1, h_i \in \mathcal{H} \right\}, \quad (6.4)$$

has become an important object of study in the machine learning literature. The reason is, traditional techniques using VC-dimension cannot be applied directly because even if the base class $\mathcal{H}$ has a finite VC-dimension, the combined class $\mathcal{F}$ has an infinite VC-dimension.

Schapire *et al.* [66, 67] pioneered a line of research to explain the effectiveness of voting algorithms. In their papers, they developed a new class of bounds on generalization of a convex combination of classifiers, expressed in terms of the empirical distribution of margins $yf(x)$. They showed that given $\alpha \in (0, 1)$, with probability at least $1 - \alpha$ for all $f \in \text{conv}(\mathcal{H})$ where $\mathcal{H}$ is a function space $h : \mathcal{X} \rightarrow \{-1, 1\}$ with VC-dimension V :

$$\mathbb{P}[yf(x) \leq 0] \leq \inf_{\delta \in (0, 1]} \left[\hat{\mathbb{P}}_N[yf(x) \leq \delta] + \frac{C}{\sqrt{N}} \left(\frac{V}{\delta^2} \log^2 \frac{N}{V} + \log \frac{1}{\alpha} \right)^{1/2} \right], \quad (6.5)$$

where C is an absolute constant. They also showed that in many experiments, voting methods tended to classify the majority of examples not only correctly, but also with large margins. This means one can expect δ to be reasonably large and hence, the bound becomes meaningful.

In [36], using the methods of theory of empirical, Gaussian, and Rademacher processes, Koltchinskii and Panchenko generalized and refined this type of bounds. They proposed to measure a function space's complexity in terms of Rademacher and Gaussian complexities, defined as follows.

Rademacher complexity The Rademacher complexity of a function space $\mathcal{H}$ is given by

$$R_N(\mathcal{H}) \stackrel{\text{def}}{=} \frac{1}{N} \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \sum_{j=1}^N \sigma_j h(x_j) \right], \quad (6.6)$$

where $\sigma = (\sigma_1, \dots, \sigma_N)$ with σ_j being *i.i.d.* Rademacher random variables (*i.e.*, random variables taking values in $\{-1, 1\}$ with equal probabilities), and $\mathbb{E}_\sigma$ denotes the expectation *w.r.t.* σ and conditional on all other variables (*i.e.*, they remain fixed).

Gaussian complexity The Gaussian complexity of a function space $\mathcal{H}$ is given by

$$G_N(\mathcal{H}) \stackrel{\text{def}}{=} \frac{1}{N} \mathbb{E}_{\mathbf{g}} \left[\sup_{h \in \mathcal{H}} \sum_{j=1}^N g_j h(x_j) \right], \quad (6.7)$$

where $\mathbf{g} = (g_1, \dots, g_N)$ with g_j being *i.i.d.* standard normal random variables, and $\mathbb{E}_{\mathbf{g}}$ denotes the expectation *w.r.t.* $\mathbf{g}$ and conditional on all other variables.

Similarly to (6.5) they proved that for all $\alpha \in (0, 1)$, with probability at least $1 - \alpha$ for all $f \in \text{conv}(\mathcal{H})$:

$$\begin{aligned} \mathbb{P}[yf(x) \leq 0] &\leq \\ &\inf_{\delta \in (0,1]} \left[\hat{\mathbb{P}}_N[yf(x) \leq \delta] + \frac{8}{\delta} R_N(\mathcal{H}) + \frac{1}{\sqrt{N}} \left(\log \log_2 \frac{2}{\delta} \right)^{1/2} \right] \\ &\quad + \frac{1}{\sqrt{N}} \left(\frac{1}{2} \log \frac{2}{\alpha} \right)^{1/2}. \end{aligned} \quad (6.8)$$

In case that $\mathcal{H}$ is a VC-class of functions $h : \mathcal{X} \rightarrow \{-1, 1\}$ with VC-dimension V , one has the following bound [75]:

$$R_N(\mathcal{H}) \leq C \sqrt{\frac{V}{N}} \quad (6.9)$$

which shows that (6.8) slightly improves (6.5) by getting rid of the logarithmic factor $\log^2 \frac{N}{V}$. By itself this improvement is not significant, but the latter generalizes function

space $\mathcal{F}$ to arbitrary, not only to the convex hulls. The generality allowed the authors to obtain sharper bounds in other situations, for example neural networks.

In the same paper [36] and in subsequent papers [37, 38, 39], Koltchinskii *et al.* further tightened the bounds for voting algorithms by exploring subsets of the convex hull to which the classifier belongs, the sparsity of the weights, and the clustering properties of the classifier.

In this chapter, we construct bounds similar to the first set of bounds of Koltchinskii and Panchenko, but applied to an *asymmetric error* rather than to the standard generalization error. We explore the linearity in the definition of an asymmetric error to derive the proofs. We leave extensions to subsequent bounds of Koltchinskii *et al.* as future work, as these bounds require many inequalities which only hold for *i.i.d.* training examples. The *i.i.d.* requirement does not hold when an asymmetric error is used. Because in this context, the training examples are treated as two independent positive and negative sets with different penalties. Positive examples are sampled from $\mathbb{P}[\cdot|y = 1]$, and negative examples are sampled from $\mathbb{P}[\cdot|y = -1]$. Developing analogous inequalities for an asymmetric error are not trivial tasks. Perhaps, the most difficult ones are the powerful Talagrand's concentration inequalities [73, 74] that Koltchinskii *et al.* used in proving their subsequent bounds.

The remaining sections are outlined as follows. We present the main results and their applications to convex combinations of classifiers in section 6.2. Discussion and conclusions are given in section 6.3.

6.2 Main Results

In bounding the standard generalization error, false positives and false negatives are treated equally, thus only the number of examples N needs considering. Here, because we treat false positives and false negatives differently, the number of positive examples

N_1 and the number of negative examples N_2 are considered instead.

Let us divide the training set S into S_1 and S_2 where S_1 is the set of positive examples x_i^+ , and S_2 is the set of negative examples x_i^- . Denote $\hat{\mathbb{P}}_{N_1}$ and $\hat{\mathbb{P}}_{N_2}$ similarly to $\hat{\mathbb{P}}_N$:

$$\hat{\mathbb{P}}_{N_1} \stackrel{\text{def}}{=} \frac{1}{N_1} \sum_{i=1}^{N_1} \delta_{x_i^+} \quad \forall x_i^+ \in S_1, \quad (6.10)$$

$$\hat{\mathbb{P}}_{N_2} \stackrel{\text{def}}{=} \frac{1}{N_2} \sum_{i=1}^{N_2} \delta_{x_i^-} \quad \forall x_i^- \in S_2, \quad (6.11)$$

where $\delta_x(\cdot)$ is the probability distribution on $\mathcal{X}$ concentrated at point x . Let $\hat{\mathbb{E}}_{N_1}$ and $\hat{\mathbb{E}}_{N_2}$ be expectations w.r.t. $\hat{\mathbb{P}}_{N_1}$ and $\hat{\mathbb{P}}_{N_2}$ respectively and conditional on all other variables.

6.2.1 Bounds for a General Class of Binary Classifiers

We restrict the function space $\mathcal{F}$ to a measurable space of functions f from $\mathcal{X}$ to $[-1, 1]$. We are interested in bounding an asymmetric error of f using the following empirical margin-based quantity:

$$\hat{J}_\lambda(f, \delta_1, \delta_2) \stackrel{\text{def}}{=} \lambda_1 \hat{\mathbb{P}}_{N_1} [f(x) \leq \delta_1] + \lambda_2 \hat{\mathbb{P}}_{N_2} [f(x) \geq -\delta_2], \quad (6.12)$$

for some δ_1 and δ_2 in $(0, 1]$. Similarly to [36], let Φ_L be the set of functions $\phi : \mathbb{R} \rightarrow [0, 1]$ satisfying Lipschitz condition L , such that $1_{[x \leq 0]} \leq \phi(x)$. Denote $\bar{\Phi}_L = \{\phi(-x) : \phi \in \Phi_L\}$. Our first result is given as follows:

Theorem 6.2.1. *Suppose $\phi_1 \in \Phi_{L_1}$ and $\phi_2 \in \bar{\Phi}_{L_2}$ for some Lipschitz conditions (L_1, L_2) . Given $t > 0$, each of the following statements is true for every $f \in \mathcal{F}$, and with probability at least $1 - \exp(-2t^2)$ over a sample of N_1 positive examples and N_2 negative*

examples:

$$J_\lambda(f) \leq \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] + 4 \sum_{i=1}^2 \lambda_i L_i R_{N_i}(\mathcal{F}) + \dot{\lambda} t, \quad (6.13)$$

$$J_\lambda(f) \leq \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] + \sqrt{2\pi} \sum_{i=1}^2 \lambda_i L_i G_{N_i}(\mathcal{F}) + \dot{\lambda} t + \ddot{\lambda}, \quad (6.14)$$

where $\dot{\lambda} \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^2 \lambda_i^2 / N_i}$ and $\ddot{\lambda} \stackrel{\text{def}}{=} 2 \sum_{i=1}^2 \frac{\lambda_i}{\sqrt{N_i}}$.

Proof. The proof is substantially based on the proof of Theorem 1 of Koltchinskii and Panchenko's paper in [36]. Let $\tilde{\phi}_i(x) = \phi_i(x) - \phi_i(0)$. We have

$$\mathcal{J}_\lambda(f) \leq \sum_i \lambda_i \mathbb{E} [\phi_i(f)] \quad (6.15)$$

$$\begin{aligned} &\leq \sum_i \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] \sup_{f' \in \mathcal{F}} \left\{ \sum_i \lambda_i \left(\mathbb{E} [\phi_i(f')] - \hat{\mathbb{E}}_{N_i} [\phi_i(f')] \right) \right\} \\ &= \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] + \sum_i \lambda_i (\mathbb{E} [\phi_i(0)] - \hat{\mathbb{E}}_{N_i} [\phi_i(0)]) \\ &\quad + \sup_{f' \in \mathcal{F}} \left\{ \sum_i \lambda_i \left(\mathbb{E} [\tilde{\phi}_i(f')] - \hat{\mathbb{E}}_{N_i} [\tilde{\phi}_i(f')] \right) \right\} \end{aligned} \quad (6.16)$$

$$\begin{aligned} &= \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] \\ &\quad + \sup_{f' \in \mathcal{F}} \left\{ \sum_i \lambda_i \left(\mathbb{E} [\tilde{\phi}_i(f')] - \hat{\mathbb{E}}_{N_i} [\tilde{\phi}_i(f')] \right) \right\} \end{aligned} \quad (6.17)$$

$$= \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} [\phi_i(f)] + \sup_{f' \in \mathcal{F}} Q f', \quad (6.18)$$

where the second last equation holds because $\mathbb{E} [\phi_i(0)] = \hat{\mathbb{E}}_{N_i} [\phi_i(0)] = \phi_i(0)$, and operator Q is defined as $Q f' \stackrel{\text{def}}{=} \sum_i \lambda_i \left(\mathbb{E} [\tilde{\phi}_i(f')] - \hat{\mathbb{E}}_{N_i} [\tilde{\phi}_i(f')] \right)$.

When example $x_i^+ \in S_1$ changes its value, $Q f'$ changes by at most λ_1 / N_1 . Similarly, when $x_i^- \in S_2$ changes its value, $Q f'$ changes by at most λ_2 / N_2 . Applying McDiarmid's inequality (see [52] or theorem B.2.4 in appendix B) on $Q f'$, we get that for $t > 0$, with

probability at most $\exp\left(\frac{-2t^2}{\sum_i \sum_{j=1}^{N_i} \lambda_i^2 / N_i^2}\right) = \exp\left(\frac{-2t^2}{\sum_i \lambda_i^2 / N_i}\right)$,

$$Qf' \geq \mathbb{E}[Qf'] + t. \quad (6.19)$$

Equivalently, by inverting (6.19), we have, with probability at least $1 - \exp(-2t^2)$,

$$Qf' \leq \mathbb{E}[Qf'] + t \sqrt{\sum_{i=1}^2 \lambda_i^2 / N_i} = \mathbb{E}[Qf'] + \lambda t. \quad (6.20)$$

Next, we have

$$\mathbb{E} \left[\sup_{f' \in \mathcal{F}} Qf' \right] \leq \mathbb{E} \left[\sum_i \lambda_i \sup_{f' \in \mathcal{F}} \left\{ \mathbb{E} [\tilde{\phi}_i(f')] - \hat{\mathbb{E}}_{N_i} [\tilde{\phi}_i(f')] \right\} \right] \quad (6.21)$$

$$= \sum_i \lambda_i \mathbb{E} \left[\sup_{g \in \tilde{\phi}_i \circ \mathcal{F}} \left\{ \mathbb{E}[g] - \hat{\mathbb{E}}_{N_i}[g] \right\} \right], \quad (6.22)$$

where $\tilde{\phi}_i \circ \mathcal{F}$ is defined as $\tilde{\phi}_i \circ \mathcal{F} \stackrel{\text{def}}{=} \{x \rightarrow \tilde{\phi}_i(f'(x)) : f' \in \mathcal{F}\}$. The symmetrization inequality in [75] (page 108-109, 177-179) gives

$$\mathbb{E} \left[\sup_{g \in \tilde{\phi}_i \circ \mathcal{F}} \left\{ \mathbb{E}[g] - \hat{\mathbb{E}}_{N_i}[g] \right\} \right] \leq 2R_{N_i}(\tilde{\phi}_i \circ \mathcal{F}) \leq \sqrt{2\pi} G_{N_i}(\tilde{\phi}_i \circ \mathcal{F}). \quad (6.23)$$

Since function $\tilde{\phi}_i/L_i$ is a contraction and $\tilde{\phi}_i(0) = 0$, the Rademacher comparison inequality (Theorem 4.12 in [41]) implies

$$R_{N_i}(\tilde{\phi}_i \circ \mathcal{F}) \leq 2L_i R_{N_i}(\mathcal{F}). \quad (6.24)$$

Combining (6.18), (6.20), (6.22), (6.23), and (6.24), we get (6.13).

To prove (6.14), we use the proof of Theorem 1 in [36], in which equation (2.8) (page 10) gives

$$G_{N_i}(\tilde{\phi}_i \circ \mathcal{F}) \leq L_i G_{N_i}(\mathcal{F}) + \sqrt{2/(\pi N_i)}. \quad (6.25)$$

Similarly, (6.14) is obtained by combining (6.18), (6.20), (6.22), (6.23), and (6.25). $\square$

Theorem 6.2.1 gives a bound that holds for a single choice of (ϕ_1, ϕ_2) , which in turns is constrained by a single pair of Lipschitz conditions (L_1, L_2) . We now give a theorem that, with high probability, the bound holds simultaneously for *all* pairs of Lipschitz conditions.

Theorem 6.2.2. *Given $t > 0$, $a_1 > \sqrt{0.5}$, $a_2 > \sqrt{0.5}$, $b_1 > 1$, and $b_2 > 1$, with probability at least $1 - \zeta(2a_1^2)\zeta(2a_2^2)\exp(-2t^2)$ over a sample of N_1 positive examples and N_2 negative examples, for every $f \in \mathcal{F}$, each of the following statements is true:*

$$J_\lambda(f) \leq \dot{\lambda}t + \inf_{\delta_1, \delta_2 \in (0,1]} \left\{ \hat{J}_\lambda(f, \delta_1, \delta_2) + 4 \sum_{i=1}^2 \frac{\lambda_i b_i}{\delta_i} R_{N_i}(\mathcal{F}) + \dot{\lambda} \sum_{i=1}^2 a_i \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}} \right\}, \quad (6.26)$$

$$J_\lambda(f) \leq \dot{\lambda}t + \ddot{\lambda} + \inf_{\delta_1, \delta_2 \in (0,1]} \left\{ \hat{J}_\lambda(f, \delta_1, \delta_2) + \sqrt{2\pi} \sum_{i=1}^2 \frac{\lambda_i b_i}{\delta_i} G_{N_i}(\mathcal{F}) + \dot{\lambda} \sum_{i=1}^2 a_i \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}} \right\}, \quad (6.27)$$

where $\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$ is the Riemann zeta function, and $\dot{\lambda}$ and $\ddot{\lambda}$ are defined as in theorem 6.2.1.

Proof. We extend the technique that both Schapire *et al.* [67] and Koltchinskii and Panchenko [36] used for proving their bounds. First, consider two sequences $\eta_{i,k} = b_i^{-k}$ for $i \in \{1, 2\}$ where $k = 1, 2, \dots$. To prove (6.26), we use (6.13) with t replaced by $t + \sum_{i=1}^2 a_i \sqrt{\log k_i}$ for some positive integers k_1 and k_2 , and with ϕ_i replaced by $\phi_i(\cdot/\eta_{i,k_i})$ for some functions ϕ_i to be chosen later. Iterating k_1 and k_2 , and uniting the

bounds altogether, we get:

$$\mathbb{P} \left[\exists f \in \mathcal{F} : J_\lambda(f) \geq \dot{\lambda} t + \right. \quad (6.28)$$

$$\left. \inf_{k_1, k_2 \geq 1} \left\{ \sum_i \lambda_i \hat{\mathbb{E}}_{N_i} \left\{ \phi_i \left(\frac{f}{\eta_{i, k_i}} \right) \right\} + 4 \sum_i \frac{\lambda_i L_i}{\eta_{i, k_i}} R_{N_i}(\mathcal{F}) + \dot{\lambda} \sum_i a_i \sqrt{\log k_i} \right\} \right] \\ \leq \sum_{k_1=1}^{\infty} \sum_{k_2=1}^{\infty} e^{-2(t + \sum_i a_i \sqrt{\log k_i})^2} \leq \sum_{k_1=1}^{\infty} \sum_{k_2=1}^{\infty} e^{-2(t^2 + \sum_i a_i^2 \log k_i)} \quad (6.29)$$

$$= \exp(-2t^2) \sum_{k_1=1}^{\infty} k_1^{-2a_1^2} \sum_{k_2=1}^{\infty} k_2^{-2a_2^2} = \zeta(2a_1^2) \zeta(2a_2^2) \exp(-2t^2). \quad (6.30)$$

Notice that for any $\delta_i \in (\eta_{i, k_i}, \eta_{i, k_i-1}]$, we have

$$\frac{1}{\eta_{i, k_i}} \leq \frac{b_i}{\delta_i}, \quad (6.31)$$

$$\hat{\mathbb{E}}_{N_i} \left[\phi_i \left(\frac{f}{\eta_{i, k_i}} \right) \right] \leq \hat{\mathbb{E}}_{N_i} \left[\phi_i \left(\frac{f}{\delta_i} \right) \right], \quad (6.32)$$

$$\sqrt{\log k_i} = \sqrt{\log \log_{b_i} \frac{1}{\eta_{i, k_i}}} \leq \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}}. \quad (6.33)$$

Therefore, (6.30) implies:

$$\zeta(2a_1^2) \zeta(2a_2^2) \exp(-2t^2) \geq \mathbb{P} \left[\exists f \in \mathcal{F} : J_\lambda(f) \geq \dot{\lambda} t + \right. \quad (6.34) \\ \left. \inf_{\delta_1, \delta_2 \in (0, 1]} \left\{ \sum_{i=1}^2 \lambda_i \hat{\mathbb{E}}_{N_i} \left[\phi_i \left(\frac{f}{\delta_i} \right) \right] + 4 \sum_i \frac{\lambda_i b_i L_i}{\delta_i} R_{N_i}(\mathcal{F}) + \dot{\lambda} \sum_i a_i \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}} \right\} \right].$$

Next, similarly to Koltchinskii and Panchenko [36], we define function $\varphi(x)$ as follows:

$$\varphi(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x \leq 0 \\ 1 - x & \text{if } 0 < x < 1 \\ 0 & \text{if } x \geq 1 \end{cases} \quad (6.35)$$

Note that $\varphi(x) \in \Phi_1$ and $\varphi(x) \leq 1_{[x \leq 1]}$ for all $x \in \mathbb{R}$. Now we replace $(\phi_1(x), \phi_2(x))$

with $(\varphi_1(x), \varphi_2(-x))$ which yields

$$\lambda_1 \hat{\mathbb{E}}_{N_1} \left[\varphi_1 \left(\frac{f}{\delta_1} \right) \right] + \lambda_2 \hat{\mathbb{E}}_{N_2} \left[\varphi_2 \left(\frac{-f}{\delta_2} \right) \right] \leq \hat{J}_\lambda(f, \delta_1, \delta_2). \quad (6.36)$$

Thus, (6.34) in turn implies (6.26).

The bound in (6.27) can be obtained by using a similar argument, with the bound in (6.13) being replaced by the one in (6.14). $\square$

Remark The use of b_1 and b_2 was briefly discussed in [36], they control the tradeoff between the second summand and the third summand of the expressions inside the infimum functions. Finding b_i minimizing these expressions are convex minimization problems, but no known analytical solution exists. One can tune a_1 and a_2 to control between the third summand and the probability of validity of a bound. Conditions $a_i > \sqrt{0.5}$ are required to ensure that $\zeta(2a_i^2) < \infty$. The bounds in [36, 67] address one error rate rather than a combination of two error rates as in ours, and correspond to the case that $b_1 = 2$ and $a_1 = 1$.

By using a counter example, Koltchinskii and Panchenko showed in Proposition 1 in [36] that, without further information, the term $R_N(\mathcal{F})/\delta$ of their bounds cannot be improved upon. This implies that $R_{N_i}(\mathcal{F})/\delta_i$ in our bounds cannot be improved upon either, by using the same argument as in [36].

6.2.2 Bounds for Convex Combinations of Classifiers

We now consider applications of the bounds developed in the previous section to convex combinations of classifiers. Given a base class $\mathcal{H}$ of simple (weak) functions $h : \mathcal{X} \rightarrow [-1, 1]$, many voting algorithms, including boosting and bagging methods, produce a complex classifier that is a convex combination of base functions, *i.e.*, belonging to the convex hull of $\mathcal{H}$. In the proof of Theorem 12 in [36], Koltchinskii and Panchenko

showed that if $\mathcal{F} = \text{conv}(\mathcal{H})$ then

$$R_N(\mathcal{F}) = R_N(\mathcal{H}). \quad (6.37)$$

Applying this to theorem 6.2.2, we get an analogous version to the bound of Koltchinskii and Panchenko [36], but applied to an asymmetric error rather than to the generalization error of a convex combination of classifiers:

Theorem 6.2.3. *Under the conditions of theorem 6.2.2, with probability at least $1 - \zeta(2a_1^2)\zeta(2a_2^2)\exp(-2t^2)$, every $f \in \text{conv}(\mathcal{H})$ satisfies*

$$J_\lambda(f) \leq \dot{\lambda}t + \inf_{\delta_1, \delta_2 \in (0,1]} \left\{ \hat{J}_\lambda(f, \delta_1, \delta_2) + 4C \sum_{i=1}^2 \frac{\lambda_i b_i}{\delta_i} R_{N_i}(\mathcal{H}) + \dot{\lambda} \sum_{i=1}^2 a_i \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}} \right\}, \quad (6.38)$$

To make a connection with the traditional bounds of Schapire *et al.* [66, 67], one uses the well-known inequality [75]:

$$R_N(\mathcal{H}) \leq C \sqrt{\frac{V}{N}}, \quad (6.39)$$

where C is a constant and V is the VC-dimension of $\mathcal{H}$, to get:

Corollary 6.2.4. *Under the conditions of theorem 6.2.2, with probability at least $1 - \zeta(2a_1^2)\zeta(2a_2^2)\exp(-2t^2)$, every $f \in \text{conv}(\mathcal{H})$ satisfies*

$$J_\lambda(f) \leq \dot{\lambda}t + \inf_{\delta_1, \delta_2 \in (0,1]} \left\{ \hat{J}_\lambda(f, \delta_1, \delta_2) + 4C \sum_{i=1}^2 \frac{\lambda_i b_i \sqrt{V}}{\delta_i \sqrt{N_i}} + \dot{\lambda} \sum_{i=1}^2 a_i \sqrt{\log \log_{b_i} \frac{b_i}{\delta_i}} \right\}, \quad (6.40)$$

where C is a constant and V is the VC-dimension of $\mathcal{H}$.

6.3 Discussion and Conclusions

We developed a new set of bounds on an asymmetric error of a binary classifier and on that of a convex combination of classifiers, based on the two class-conditional empirical margin distributions.

The bounds show that, unlike traditional methods where there is no difference between the margin of a positive example and that of a negative example, the penalties associated with the margins are different for different classes. This suggests that traditional learning methods that treat margins of the two positive and negative classes equally (*e.g.*, AdaBoost and asymmetric boosting methods) are sub-optimal in learning an asymmetric classifier.

Our bounds are based on the first set of bounds of Koltchinskii and Panchenko [36], which improve the bounds of Schapire *et al.* [66, 67] in boosting and bagging, and those of Bartlett [4] in neural network learning. These bounds treat all classifiers in the function space $\mathcal{F}$ equally, and they do not take into account any property of the classifier f itself. In a special case when $\lambda_1 = \mathbb{P}[y = 1]$ and $\lambda_2 = \mathbb{P}[y = -1]$, an asymmetric error is precisely equal to the generalization error. In this case, our bounds are slightly tighter than those of Koltchinskii and Panchenko because they take into account the prior probabilities of the two classes.

Koltchinskii *et al.* [37, 38, 39], however, have developed a number of new bounds in boosting under more restricted assumptions on (subsets of) the function space $\mathcal{F}$, and the classifier f itself. These bounds require a number of inequalities which do not hold when an asymmetric error is used. Deriving bounds on asymmetric error that are analogous to the new bounds of Koltchinskii *et al.* is a potential direction for future work.

Chapter 7

Conclusions and Future Work

7.1 Summary of Research

The central theme of the research done in this thesis is the application of asymmetric boosting to significantly improve face detectors in learning time, learning capacity, detection speed, and detection's accuracy.

In chapter 3, we presented an integrated framework for the *online* learning of *asymmetric* boosted classifiers. Previous methods addressed either the *online* learning of a symmetric boosted classifier, or the offline learning of an *asymmetric* classifier, but not both. In addition to integrating the two lines of research, our method seeks to balance the skewness of the labels presented to the weak classifiers, allowing them to be trained more equally, and hence resulting in a small gain in accuracy. In terms of learning online, we enforced more conditions to the weights of the examples presented to weak classifiers, resulting in a faster convergence speed than the traditional online learning algorithm [56] used for object detection problems. Our experiments showed that, by replacing traditional methods with our algorithm, about a 0-10% increase in accuracy and about 5-30% gain in learning speed were observed, depending on the problem the algorithms were applied to.

In chapter 4, we presented a fast method to train and select Haar-like features for training of a weak classifier in face detection cascades. It trains a weak classifier in time $O(N + T)$ (in seconds), rather than $O(NT \log N)$ (in minutes) in traditional methods. The reduction in the training time complexity, from the *product* of two large quantities, N and T , to the *sum* of those two, leads to a dramatic reduction in the training time. While conventional methods trained a face detector in days or even weeks, our method trained a face detector in just a few hours. In compared with the currently fastest implementation of boosting [86], taking about 12 seconds per weak classifier using 40,000 Haar-like features but requiring almost 1GB of memory, this method trained a weak classifier using nearly 300,000 Haar-like features in just 6 seconds, while requiring less than 30MB of memory and maintaining a compelling performance. Besides, this method suffered a very minimal increase in the training time with very large increases in members of Haar-like features, enjoying a significant gain in accuracy. By substantially reducing the training time, the method empowers researchers to much more quickly experiment and explore solutions to other important research issues in this area.

In chapter 5, we proposed a method to train a multi-exit boosted classifier, which is a better model than traditional cascades, by combining the idea of propagating scores in [87, 89] and training with an asymmetric goal in [80]. More importantly, we showed how to properly select an asymmetric goal that achieves the desired error rates with a minimum number of weak classifiers, avoiding the need to run multiple ad hoc trials to discover the best boosted classifiers satisfying the operating point requirements. Experimental results showed not only significant reduction in the training time and the number of weak classifiers, but also better accuracy compared to conventional cascades and multi-exit boosted classifiers.

The success of AdaBoost and some of its variants has been recently explained, to some limited extent, by using margin-based bounds on the generalization error of the classifier learned by the algorithms. These bounds, however, are not applicable to asym-

metric boosted classifiers as the learning goals of AdaBoost and asymmetric boosting are different. The learning goal of the former is symmetric in the sense that there is no difference between a false positive and a false negative. For the latter, there is a clear distinction between a false positive and a false negative. In an attempt to understand the generalization ability of asymmetric boosting, in chapter 6, we developed the first margin-based bounds on the true asymmetric error of a classifier trained with an asymmetric goal. Our bounds show that, unlike symmetric bounds, the margins of positive examples and those of negative examples are of different importance. This suggests that, with a proper proportion (which is currently not clearly understood), a learning algorithm yielding margins asymmetrically for different classes may actually improve the generalization ability of the asymmetric boosted classifier that it learns.

7.2 Future Work

It is often the case with scientific investigations that more questions than answers are produced. The research described in this thesis has led to methods for improving learning in classification problems with imbalanced data like face detection. Some potential future research work includes:

7.2.1 Learning a Weak Classifier

Extending Haar-like features In chapter 4, we addressed the problem of *fast* training a large number of Haar-like features using statistics. Haar-like features are very useful for detection because they can be extracted *extremely* fast, in just a few hundreds of CPU clock cycles per feature. However, currently they are selected manually from a predefined set of Haar-like feature types. This raises a question of whether one can generalize this set to a set of all possible linear features that are sparse enough for fast extraction. Perhaps, one should also discuss about tradeoff between sparsity and extraction time.

Fast searching for Haar-like features If the above issue is solved, then searching for the *best* Haar-like feature becomes problematic because now we have to deal with a *large* set of features. Current linear scan solutions are not adequate. Can we search for the best feature, or perhaps a nearly best feature, in sub-linear time or even log time? Those are challenging open problems that can be addressed.

7.2.2 Learning a Boosted Classifier

Consistency and convergence rate of asymmetric boosting One area of research in computational learning theory is whether methods like AdaBoost are consistent, *i.e.*, whether its generalization error approaches Bayes error as the number of examples approaches infinity. Only recently did Bartlett and Traskin [5] prove that AdaBoost is consistent. This naturally leads to a question of whether asymmetric boosting methods are also consistent. Of course, in this case, the theoretical minimum error in question is not the original Bayes error, but is rather the Bayes asymmetric error described in (5.17) in chapter 5:

$$\varepsilon_\lambda = \int_x \min \left\{ \frac{\lambda}{\lambda + 1} \mathbb{P}[y = 1|x], \frac{1}{\lambda + 1} \mathbb{P}[y = -1|x] \right\} \mathbb{P}[x] dx. \quad (7.1)$$

Another aspect of consistency is the rate of convergence towards Bayes risk of boosting methods. For AdaBoost, the rate is rather pessimistic, $\log^{-1/2} t$ [5] where t is the number of iterations/weak classifiers. If asymmetric boosting is consistent, then is the convergence rate similar or worse?

Online asymmetric boosting using real-valued weak classifiers The underlying boosting method described in chapter 3 is a variant of Discrete AdaBoost in which weak classifiers produce discrete-valued outputs. With the recent development of boosting methods in which weak classifiers produce real-valued outputs (*i.e.*, Friedman *et al.* [21] and Schapire and Singer *et al.* [67]), one would naturally ask whether the proposed on-

line asymmetric boosting method can be extended to these algorithms. Our unpublished preliminary results show that deriving a framework is not an issue, but the stability of such an algorithm is the main challenge. If a real-valued weak classifier wrongly predicts an example, the next weak classifier in the sequence would learn the example with large weight. Our experiments show that these weights are much larger than those in the case of discrete-valued boosting. Hence, it takes much more iterations to converge a real-valued weak classifier than a discrete-valued one. Future work should consider this issue with care.

Sharper asymmetric bounds We proved in chapter 6 the first set of upper bounds for *asymmetric generalization error* based on the first bound of Koltchinskii and Panchenko [36]. Recently, Koltchinskii proved new and sharper bounds under additional but reasonable assumptions [39, 38, 37]. This naturally raises a question of whether our current asymmetric bounds can be sharpened based on their techniques. This is not easily foreseeable as the techniques, as admitted by the authors, are very hard to derive. Whether they can be applied to our case is still not clear.

7.2.3 Beyond Learning a Classifier

Designing desired error rates One issue we have not addressed in this thesis is how to choose desired error rates to asymmetrically train a boosted classifier. Cascades and multi-exit boosted classifiers can be considered as machines making a sequence of decisions whereby late decisions are harder to make than early ones. There is clearly a trade-off among three important factors of a boosted classifier: the detection rate, the false acceptance rate, and the number of weak classifiers. In chapter 5 we showed how to minimize the number of weak classifiers given bounds for false acceptance rate and false rejection rate. The remaining question is how to choose these sets of bounds over all the exit nodes of a multi-exit boosted classifier or a cascade. Some progress has been made to address this problem [9, 70] but so far satisfactory results and full understanding

have not been achieved.

Fast generating false positive examples for bootstrapping A less important problem when training with bootstrapping is that as more negative examples are rejected, it is harder to generate false positives for training subsequent boosted classifiers. At the moment, negative examples are first generated randomly from a large set of non-object images. The currently trained cascade or multi-exit boosted classifier is then used to filter all true negatives, leaving only false positives to train the next boosted classifier. The problem is, as more boosted classifiers are trained, the probability of generating a false positive approaches zero and the average time to actually achieve it goes to infinity (*i.e.*, inversely proportional to the probability). In the future, a different strategy to generate false positives must be invented to avoid this problem.

Online learning of a face detector An interesting yet challenging line of research that has become active recently is how to learn *online* a face/object detector [24, 33, 34, 58]. The problem itself is still open, but it has an interesting insight as follows. An offline learned face detector is usually trained with a fixed set of positive/negative examples. In order to achieve high accuracy, the provided training set has to be large and diverse enough to generalize to any possible user environment. However, when the detector is applied to a specific environment of a user, it is so often that the environment it works with is much smaller than what it has been trained for. If we can explore the user environment on the fly, it is theoretically possible to build/re-design a face detector to work specifically towards the user environment. The key point here is, because the environment is now much smaller than the one used for training in offline, we expect the resulting face detector is much less complex, yielding much faster detection speed.

Pioneered work has shown some promising results [58, 60]. However, the problem is far from being solved. The main issue is how to online-learn a cascade or a multi-exit boosted classifier, which is a lot more difficult than online-learning a boosted classifier alone due to the involvement of a sequence of decisions. However, throughout this thesis,

all the chapters address many key issues for online-learning a face detector. Chapter 3 shows how we can learn online a boosted classifier with an asymmetric goal. The technique developed in chapter 4 can be used to dramatically speed up the online learning of a weak classifier, by learning online blocks of examples for instance. Chapter 5 proves that the asymmetric goal used for training can be properly chosen prior to training so that the number of weak classifiers can be minimized. And chapter 6 explains the relationship between the empirical asymmetric error of a boosted classifier and its generalization ability.

With these achievements, perhaps the problem of online-learning a face/object detector may be solvable in a very near future.

Appendix A

Derivations for Training a Haar-like Feature Classifier

This chapter shows how we can train a feature classifier under class-conditional Gaussian distribution assumptions by using closed-form solutions, implying running in constant time. For simplicity, we omit subscripts m and t . Thus, a feature classifier $\eta(v)$ assumes $v|c \sim \mathcal{N}(\mu_c, \sigma_c^2)$ for every $c \in \mathcal{Y}$, where v is the random variable representing the observed feature value when the image sub-window is weighted by $w_m(\cdot)$. Denote by $\hat{\mathbb{P}}$ the weighted distribution presented to the classifier. To train the classifier, one finds a threshold $\hat{\theta} \in \mathbb{R}$ and a parity $\hat{p} \in \{-1, 1\}$ such that

$$\hat{\theta}, \hat{p} = \arg \min_{\theta, p} \left\{ \hat{\mathbb{P}}[v > \theta \wedge y = -p] + \hat{\mathbb{P}}[v < \theta \wedge y = p] \right\}. \quad (\text{A.1})$$

We can try both cases of the parity and select the one with a smaller error. Hence

without loss of generality, we assume $\hat{p} = 1$ and rewrite the error function:

$$\epsilon(\theta) \stackrel{\text{def}}{=} \hat{\mathbb{P}}[v > \theta \wedge y = -1] + \hat{\mathbb{P}}[v < \theta \wedge y = 1] \quad (\text{A.2})$$

$$= \hat{\mathbb{P}}[y = -1] \hat{\mathbb{P}}[v > \theta | y = -1] + \hat{\mathbb{P}}[y = 1] \hat{\mathbb{P}}[v < \theta | y = 1] \quad (\text{A.3})$$

$$= \frac{\hat{\mathbb{P}}[y = -1]}{\sqrt{2\pi}\sigma_{-1}} \int_{v=\theta}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{v - \mu_{-1}}{\sigma_{-1}}\right)^2\right) dv + \frac{\hat{\mathbb{P}}[y = 1]}{\sqrt{2\pi}\sigma_1} \int_{v=-\infty}^{\theta} \exp\left(-\frac{1}{2}\left(\frac{v - \mu_1}{\sigma_1}\right)^2\right) dv. \quad (\text{A.4})$$

Let $a_{-1} = \frac{\hat{\mathbb{P}}[y=-1]}{\sqrt{2\pi}\sigma_{-1}}$ and $a_1 = \frac{\hat{\mathbb{P}}[y=1]}{\sqrt{2\pi}\sigma_1}$. Differentiating $\epsilon(\theta)$ w.r.t. θ we get:

$$\epsilon'_\theta(\theta) = a_1 \exp\left(-\frac{1}{2}\left(\frac{\theta - \mu_1}{\sigma_1}\right)^2\right) - a_{-1} \exp\left(-\frac{1}{2}\left(\frac{\theta - \mu_{-1}}{\sigma_{-1}}\right)^2\right). \quad (\text{A.5})$$

$\epsilon'_\theta(\theta) = 0$ if and only if:

$$a_1 \exp\left(-\frac{1}{2}\left(\frac{\theta - \mu_1}{\sigma_1}\right)^2\right) = a_{-1} \exp\left(-\frac{1}{2}\left(\frac{\theta - \mu_{-1}}{\sigma_{-1}}\right)^2\right) \quad (\text{A.6})$$

$$\Leftrightarrow 2 \log a_1 - \left(\frac{\theta - \mu_1}{\sigma_1}\right)^2 = 2 \log a_{-1} - \left(\frac{\theta - \mu_{-1}}{\sigma_{-1}}\right)^2 \quad (\text{A.7})$$

$$\Leftrightarrow \left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_{-1}^2}{\sigma_{-1}^2}\right) \theta^2 - 2\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_{-1}}{\sigma_{-1}^2}\right) \theta + \left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_{-1}^2} + 2 \log a_{-1} - 2 \log a_1\right) = 0, \quad (\text{A.8})$$

where the last equation is achieved by re-arranging the terms. Let

$$A \stackrel{\text{def}}{=} \frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_{-1}^2}{\sigma_{-1}^2}, \quad (\text{A.9})$$

$$B \stackrel{\text{def}}{=} 2\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_{-1}}{\sigma_{-1}^2}\right), \quad (\text{A.10})$$

$$C \stackrel{\text{def}}{=} \frac{1}{\sigma_1^2} - \frac{1}{\sigma_{-1}^2} + 2 \log a_{-1} - 2 \log a_1. \quad (\text{A.11})$$

Solving $\epsilon'_\theta(\theta) = 0$ becomes solving a quadratic equation

$$A\theta^2 + B\theta + C = 0. \tag{A.12}$$

Denote by Θ the set of solutions of (A.12). Depending on the values of (A, B, C) , there are at most two closed-form solutions. Since we do not know if any $\theta \in \Theta$ corresponds to a local minimum of $\epsilon(\theta)$, we just have to select $\hat{\theta}$ from:

$$\hat{\theta} = \arg \min_{\theta \in \Theta \cup \{-\infty, +\infty\}} \{\epsilon(\theta)\}. \tag{A.13}$$

Appendix B

Probability Tools

This chapter recalls facts from probability theory that are used throughout the thesis, sometimes without explicitly mentioning it.

B.1 Basic Tools

We denote by A and B some events and by x some real-valued random variable.

- Union:

$$\mathbb{P}[A \text{ or } B] \leq \mathbb{P}[A] + \mathbb{P}[B]. \quad (\text{B.1})$$

- Inclusion:

$$\text{If } A \Rightarrow B \text{ then } \mathbb{P}[A] \leq \mathbb{P}[B]. \quad (\text{B.2})$$

- Inversion: If $\mathbb{P}[x > t] \leq F(t)$ then with probability at least $1 - \delta$,

$$x \leq F^{-1}(\delta). \quad (\text{B.3})$$

- Expectation:

$$\text{If } x \geq 0 \text{ then } \mathbb{E}[x] = \int_0^\infty \mathbb{P}[x \geq t] dt. \quad (\text{B.4})$$

- Jensen: Given function f convex,

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]. \quad (\text{B.5})$$

B.2 Concentration Inequalities

Recently, new tools have appeared in probability theory. These tools called concentration inequalities allow to bound the deviation of a random function from its expectation, just knowing the sensitivity of this function to the change or removal of one of its arguments. We give a list of some of these inequalities here. We refer to [7] for a comprehensive review of concentration inequalities.

First of all, for a non-negative random variable x , (B.4) yields *Markov's inequality*:

$$\mathbb{P}[x \geq t] \leq \frac{\mathbb{E}[x]}{t}, \quad (\text{B.6})$$

for all $t > 0$.

It follows that if ϕ is a monotonically increasing non-negative function, we can replace x with $\phi(x)$:

$$\mathbb{P}[\phi(x) \geq \phi(t)] \leq \frac{\mathbb{E}[\phi(x)]}{\phi(t)}. \quad (\text{B.7})$$

Applying $\phi(x) = x^2$ and replacing x with $|x - \mathbb{E}[x]|$, we obtain *Chebyshev's inequality*:

$$\mathbb{P}[|x - \mathbb{E}[x]| \geq t] = \mathbb{P}[|x - \mathbb{E}[x]|^2 \geq t^2] \leq \frac{\mathbb{E}[|x - \mathbb{E}[x]|^2]}{t^2} = \frac{\text{Var}\{x\}}{t^2}. \quad (\text{B.8})$$

In another application, if we take $\phi(x) = \exp(sx)$ for some positive number s , then

for any random variable x , we have

$$\mathbb{P}[x \geq t] = \mathbb{P}[\exp(sx) \geq \exp(st)] \leq \frac{\mathbb{E}[\exp(sx)]}{\exp(st)}. \quad (\text{B.9})$$

We find an $s > 0$ that minimizes the upper bound. This is the basis of *Chernoff's bounding method*.

Chernoff's bounding method is especially convenient for bounding tail probabilities of sums of independent random variables. Given N independent real-valued random variables $x_1, \dots, x_N$, let $S_N = \sum_{i=1}^N x_i$. Chernoff's bound becomes

$$\begin{aligned} & \mathbb{P}[S_N - \mathbb{E}[S_N] \geq t] \\ & \leq \exp(-st) \mathbb{E} \left[\exp \left(s \sum_{i=1}^N (x_i - \mathbb{E}[x_i]) \right) \right] \\ & = \exp(-st) \prod_{i=1}^N \mathbb{E}[\exp(s(x_i - \mathbb{E}[x_i]))] \quad (\text{by independence}). \end{aligned} \quad (\text{B.10})$$

Now the problem of finding tight bounds comes down to finding a good upper bound for the moment generating function of the random variables $x_i - \mathbb{E}[x_i]$. There are many ways of doing this. A classical and most celebrated version is due to Hoeffding [28] which we state without proof.

Lemma B.2.1 (Hoeffding's lemma [28]). *Let x be a random variable with $\mathbb{E}[x] = 0$, $a \leq x \leq b$. Then for $s > 0$,*

$$\mathbb{E}[\exp(sx)] \leq \exp(s^2(b-a)^2/8).$$

Combining this lemma with (B.10) yields:

Theorem B.2.2 (Hoeffding's inequality [28]). *Let $x_1, \dots, x_N$ be independent (but not necessarily identically distributed) random variables with $x_i \in [a_i, b_i]$ almost surely. Let*

$S_N = \sum_{i=1}^N x_i$. For any $t > 0$ we have

$$\mathbb{P}[S_N - \mathbb{E}[S_N] \geq t] \leq \exp\left(-2t^2 / \sum_{i=1}^N (b_i - a_i)^2\right)$$

and

$$\mathbb{P}[S_N - \mathbb{E}[S_N] \leq -t] \leq \exp\left(-2t^2 / \sum_{i=1}^N (b_i - a_i)^2\right).$$

An application of Hoeffding's inequality for binomial random variables was proved by Chernoff [10]:

Corollary B.2.3 (Chernoff's bound [10]). *Assume $x_1, \dots, x_N$ are i.i.d. random variables. Let $p = \mathbb{E}[x_i]$, $x_i \in \{0, 1\}$ and $\varepsilon > 0$. Then*

$$\mathbb{P}\left[\frac{1}{n} \sum x_i \geq p + \varepsilon\right] \leq \exp(-2\varepsilon^2 N)$$

and

$$\mathbb{P}\left[\frac{1}{n} \sum x_i \leq p - \varepsilon\right] \leq \exp(-2\varepsilon^2 N).$$

About two decades after Hoeffding's inequality was published, McDiarmid [52] derived a stronger and more general inequality:

Theorem B.2.4 (McDiarmid's inequality [52]). *Let $x_1, \dots, x_N$ be independent (but not necessarily identically distributed) random variables taking values in a set $\mathcal{A}$, and assume that $f : \mathcal{A}^N \rightarrow \mathbb{R}$ is a function satisfying*

$$\sup_{x_1, \dots, x_N, \hat{x}_i} |f(x_1, \dots, x_N) - f(x_1, \dots, x_{i-1}, \hat{x}_i, x_{i+1}, \dots, x_N)| \leq c_i$$

for every $1 \leq i \leq N$. Then, for every $\varepsilon > 0$:

$$\mathbb{P}[f(x_1, \dots, x_N) - \mathbb{E}[f(x_1, \dots, x_N)] \geq \varepsilon] \leq \exp\left(-2\varepsilon^2 / \sum_{i=1}^N c_i^2\right).$$

McDiarmid gave a proof based on the theory of martingale, and an extension of Hoeffding's lemma:

Lemma B.2.5. *Suppose for random variable v and random vector $\mathbf{y}$, we have $\mathbb{E}[v|\mathbf{y}] = 0$ almost surely and $f(\mathbf{y}) \leq v \leq f(\mathbf{y}) + c$, for some function f and constant c . Then, for any $s > 0$,*

$$\mathbb{E}[\exp(sv) | \mathbf{y}] \leq \exp(s^2 c^2 / 8).$$

Proof. of lemma B.2.5: By applying Hoeffding's lemma with x replaced by $v|\mathbf{y}$, a replaced by $f(\mathbf{y})$, and b replaced by $f(\mathbf{y}) + c$, we get the result. $\square$

Proof. of McDiarmid's inequality: Let $\mathbf{x} = (x_1, \dots, x_N)$ and a discrete-time stochastic process $z_2, \dots, z_n$ where $z_k = \mathbb{E}[f(\mathbf{x}) | x_1, \dots, x_k]$. Obviously, $z_2 = \mathbb{E}[f(\mathbf{x})]$ and $z_n = f(\mathbf{x})$.

Sequence z_k is, by definition, a martingale w.r.t. sequence x_k since

$$\begin{aligned} \mathbb{E}[z_k | x_1, \dots, x_{k-1}] &= \mathbb{E}[\mathbb{E}[f(\mathbf{x}) | x_1, \dots, x_k] | x_1, \dots, x_{k-1}] \\ &= \mathbb{E}[f(\mathbf{x}) | x_1, \dots, x_{k-1}] = z_{k-1}. \end{aligned} \quad (\text{B.11})$$

Let $u_k = \sup_v z_k |_{x_k=v}$ and $l_k = \inf_v z_k |_{x_k=v}$. We have

$$l_k \leq z_k \leq u_k. \quad (\text{B.12})$$

Moreover,

$$\begin{aligned} u_k - l_k &\leq \sup_{l,u} \{ \mathbb{E}[f(\mathbf{x}) | x_1, \dots, x_k = u] - \mathbb{E}[f(\mathbf{x}) | x_1, \dots, x_k = l] \} \\ &= \sup_{l,u} \mathbb{E}[\{f(\mathbf{x})|_{x_k=u} - f(\mathbf{x})|_{x_k=l}\} | x_1, \dots, x_{k-1}] \\ &\leq \sup_{l,u} \mathbb{E}[c_k | x_1, \dots, x_{k-1}] \\ &= c_k. \end{aligned} \quad (\text{B.13})$$

Applying lemma B.2.5 with $v = z_k - z_{k-1}$ and $\mathbf{y} = (x_1, \dots, x_{k-1})$, noticing that $\mathbb{E}[v|\mathbf{y}] = \mathbb{E}[z_k|\mathbf{y}] - z_{k-1} = 0$ and $(l_k - z_{k-1}) \leq v \leq (l_k - z_{k-1}) + c_k$, we obtain:

$$\mathbb{E}[\exp(s(z_k - z_{k-1})) | x_1, \dots, x_{k-1}] \leq \exp(s^2 c_k^2 / 8). \quad (\text{B.14})$$

Now, starting with Chernoff's bounding method, for $s > 0$ we have:

$$\begin{aligned} \mathbb{P}[f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})] \geq \varepsilon] &\leq e^{-s\varepsilon} \mathbb{E}[\exp(s(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})]))] \\ &= e^{-s\varepsilon} \mathbb{E}\left[\exp\left(s \sum_{k=1}^n (z_k - z_{k-1})\right)\right] \end{aligned} \quad (\text{B.15})$$

Notice that:

$$\begin{aligned} &\mathbb{E}\left[\exp\left(s \sum_{k=1}^n (z_k - z_{k-1})\right)\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\exp\left(s \sum_{k=1}^n (z_k - z_{k-1})\right) | x_1, \dots, x_{n-1}\right]\right] \\ &= \mathbb{E}\left[\exp\left(s \sum_{k=1}^{n-1} (z_k - z_{k-1})\right) \mathbb{E}[\exp(s(z_n - z_{n-1})) | x_1, \dots, x_{n-1}]\right] \\ &\leq \mathbb{E}\left[\exp\left(s \sum_{k=1}^{n-1} (z_k - z_{k-1})\right) \exp(s^2 c_n^2 / 8)\right] \\ &= \exp(s^2 c_n^2 / 8) \mathbb{E}\left[\exp\left(s \sum_{k=1}^{n-1} (z_k - z_{k-1})\right)\right] \end{aligned} \quad (\text{B.16})$$

Applying the same argument $n - 1$ times, we get:

$$\begin{aligned} e^{-s\varepsilon} \mathbb{E}\left[\exp\left(s \sum_{k=1}^n (z_k - z_{k-1})\right)\right] &\leq e^{-s\varepsilon} \prod_{k=1}^n \exp(s^2 c_k^2 / 8) \\ &= \exp\left(-s\varepsilon + \frac{s^2}{8} \sum_k c_k^2\right). \end{aligned} \quad (\text{B.17})$$

Minimizing the right-hand side of (B.17) w.r.t. s , we get the optimal point $s = \frac{4\varepsilon}{\sum_k c_k^2}$, at which the equation equals $\exp(-2\varepsilon^2 / \sum_k c_k^2)$. $\square$

Bibliography

- [1] Y. Abramson, B. Steux, and H. Ghorayeb. Yet even faster (YEF) real-time object detection. *International Journal of Intelligent Systems Technologies and Applications*, 2(2/3):102–112, 2007.
- [2] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, 1999.
- [3] S. Baluja, M. Sahami, and H. A. Rowley. Efficient face orientation discrimination. In *International Conference on Image Processing*, pages 589–592, 2004.
- [4] P. L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.
- [5] P. L. Bartlett and M. Traskin. Adaboost is consistent. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 105–112. MIT Press, Cambridge, MA, USA, 2007.
- [6] B. Bhanu. Automatic target recognition: State of the art survey. *IEEE Trans. on Aerospace and Electric Systems*, 22(4):364–379, 1986.
- [7] S. Boucheron, G. Lugosi, and O. Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning*, pages 208–240, 2003.
- [8] L. D. Bourdev and J. Brandt. Robust object detection via soft cascade. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*

- (*CVPR 2005*), volume 2, pages 236–243, San Diego, CA, USA, 2005.
- [9] S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Towards optimal training of cascaded detectors. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision*, volume 3951 of *Lecture Notes in Computer Science*, pages 325–337, Graz, Austria, 2006.
- [10] H. Chernoff. A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.
- [11] A. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 782–787, 1997.
- [12] I. Craw, D. Tock, and A. Bennett. Finding face features. In *Computer Vision - ECCV'92, Second European Conference on Computer Vision*, pages 92–96, 1992.
- [13] Y. Dai and Y. Nakano. Face-texture model based on sgld and its application in face detection in a color scene. *Pattern Recognition*, 29(6):1007–1017, 1996.
- [14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pages 886–893, 2005.
- [15] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, February 1997.
- [16] 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 *12th International Conference on Interactive Technologies and Sociotechnical Systems, VSMM 2006*, pages 128–137, Xi'an, China, 2006.
- [17] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

- [18] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: Misclassification cost-sensitive boosting. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, pages 97–105, Bled, Slovenia, 1999. Morgan Kaufmann.
- [19] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53(1-2):71–109, 2003.
- [20] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96)*, pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.
- [21] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.
- [22] D. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.
- [23] K. Goto and R. van de Geijn. High-performance implementation of the level-3 blas. Technical report, The University of Texas at Austin, Department of Computer Sciences, 2006.
- [24] H. Grabner and H. Bischof. On-line boosting and vision. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, volume 1, pages 260–267, New York, NY, USA, 2006.
- [25] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [26] S. Hettich and S. D. Bay. The UCI KDD archive. University of California, Irvine, Dept. of Information and Computer Sciences, 1999.
- [27] E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001.

- [28] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58:13–30, 1963.
- [29] X. Hou, C.-L. Liu, and T. Tan. Learning boosted asymmetric classifiers for object detection. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, volume 1, pages 330–338, New York, NY, USA, 2006.
- [30] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *10th IEEE International Conference on Computer Vision (ICCV 2005)*, volume 1, pages 446–453, Beijing, China, 2005.
- [31] C. Huang, H. Ai, Y. Li, and S. Lao. Learning sparse features in granular space for multi-view face detection. In *Seventh IEEE International Conference on Automatic Face and Gesture Recognition (FG 2006)*, pages 401–407, Southampton, UK, 2006.
- [32] C. Huang, H. Ai, Y. Li, and S. Lao. High-performance rotation invariant multiview face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4):671–686, 2007.
- [33] C. Huang, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Incremental learning of boosted face detector. In *11th IEEE International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil, 2007.
- [34] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 696–701, San Diego, CA, USA, 2005.
- [35] M. Jones and P. A. Viola. Fast multi-view face detection. Technical report, Mitsubishi Electric Research Laboratories, 2003.

- [36] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50, 2002.
- [37] V. Koltchinskii and D. Panchenko. Complexities of convex combinations and bounding the generalization error in classification. *The Annals of Statistics*, 33(4):1455–1496, 2005.
- [38] V. Koltchinskii, D. Panchenko, and S. Andonova. Generalization bounds for voting classifiers based on sparsity and clustering. In *16th Annual Conference on Computational Learning Theory*, pages 492–505, 2003.
- [39] V. Koltchinskii, D. Panchenko, and F. Lozano. Bounding the generalization error of convex combinations of classifiers: Balancing the dimensionality and the margins. *Annals of Applied Probability*, 13(1):213–252, 2003.
- [40] A. Lanitis, C. Taylor, and T. Cootes. Automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401, June 1995.
- [41] M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Springer, 1991.
- [42] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using labeled random graph matching. In *Proceedings of the Fifth International Conference on Computer Vision (ICCV 95)*, pages 637–644, 1995.
- [43] M. S. Lew. Information theoretic view-based and modular face detection. In *2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, pages 198–203, 1996.
- [44] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision - ECCV 2002, 7th European Conference on Computer*

- Vision*, volume 2353 of *Lecture Notes in Computer Science*, pages 67–81, Copenhagen, Denmark, 2002.
- [45] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In B. Michaelis and G. Krell, editors, *25th DAGM Symposium*, volume 2781 of *Lecture Notes in Computer Science*, pages 297–304, Magdeburg, Germany, 2003.
- [46] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of the 2002 International Conference on Image Processing (ICIP 2002)*, volume 1, pages 900–903, Rochester, NY, USA, 2002.
- [47] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110, 2004.
- [48] H. Luo. Optimization design of cascaded classifiers. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 480–485, San Diego, CA, USA, 2005.
- [49] Y. Ma and X. Ding. Robust real-time face detection based on cost-sensitive adaboost method. In *2003 International Conference on Multimedia and Expo*, volume 2, pages 465–468, Beijing, China, 2003.
- [50] A. Martinez and R. Benavente. The ar face database. Technical Report 24, CVC, U.A.B., 1998.
- [51] H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, pages 609–619, 2007.
- [52] C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, 1989.

- [53] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision*, pages 128–142, 2002.
- [54] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. In *10th IEEE International Conference on Computer Vision (ICCV 2005)*, volume 2, pages 1619–1626, Beijing, China, 2005.
- [55] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 130–136, 1997.
- [56] N. C. Oza. Online bagging and boosting. In *International Conference on Systems, Man, and Cybernetics, Special Session on Ensemble Methods for Extreme Environments*, volume 3, pages 2340–2345, 2005.
- [57] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV-98)*, pages 555–562, 1998.
- [58] M.-T. Pham and T.-J. Cham. Detection caching for faster object detection. In *Proc. IEEE International Workshop on Modeling People and Human Interaction (PHI'05)*, Beijing, China, 2005.
- [59] M.-T. Pham and T.-J. Cham. Fast training and selection of haar features using statistisc in boosting-based face detection. In *11th IEEE International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil, 2007.
- [60] M.-T. Pham and T.-J. Cham. Online learning asymmetric boosted classifiers for object detection. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN, USA, 2007.

- [61] M.-T. Pham, V.-D. D. Hoang, and T.-J. Cham. Detection with multi-exit asymmetric boosting. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK, USA, 2008.
- [62] T. Poggio and K. K. Sung. Finding human faces with a gaussian mixture distribution-based face model. In *Second Asian Conference on Computer Vision (ACCV '95)*, pages 437–446, 1995.
- [63] A. N. Rajagopalan, K. S. Kumar, J. Karlekar, R. Manivasakan, M. M. Patil, U. B. Desai, P. G. Poonacha, and S. Chaudhuri. Finding faces in photographs. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV-98)*, pages 640–645, 1998.
- [64] P. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In R. Chellappa, J. Ferryman, and T. Tan, editors, *Proc. 2nd Joint IEEE Workshop on Video Surveillance and Performance Evaluation on Tracking and Surveillance*, pages 223–230, 2005.
- [65] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *1998 Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pages 38–44, Santa Barbara, CA, USA, 1998.
- [66] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.
- [67] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
- [68] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR*, pages 1746–1759, 2000.

- [69] J. Sochman and J. Matas. Waldboost: Learning for time constrained sequential detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 2, pages 150–156, 2005.
- [70] J. Sun, J. M. Rehg, and A. F. Bobick. Automatic cascade training with perturbation bias. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 276–283, Washington, DC, USA, 2004.
- [71] K. K. Sung. *Learning and Example Selection for Object and Pattern Detection*. PhD thesis, Massachusetts Institute of Technology, 1996.
- [72] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.
- [73] M. Talagrand. New concentration inequalities in product spaces. *Invent. Math.*, 126:505–563, 1996.
- [74] M. Talagrand. A new look at independence. *The Annals of Probability*, 24:1–34, 1996.
- [75] A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, November 2000.
- [76] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [77] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *9th IEEE International Conference on Computer Vision (ICCV 2003)*, pages 281–288, Nice, France, 2003.
- [78] M. Vidyasagar. *A Theory of Learning and Generalization: With Applications to Neural Networks and Control Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

- [79] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume 1, pages 511–518, Kauai, HI, USA, 2001.
- [80] P. A. Viola and M. J. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems 14*, pages 1311–1318. MIT Press, Cambridge, MA, USA, 2002.
- [81] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal on Computer Vision*, 57(2):137–154, 2004.
- [82] A. Wald. *Sequential Analysis (Dover Phoenix Editions)*. Dover Publications, dover ed edition, June 2004.
- [83] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FGR 2004)*, pages 79–84, Seoul, Korea, 2004.
- [84] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal on Computer Vision*, 75(2):247–266, 2007.
- [85] B. Wu and N. Ram. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *11th IEEE International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil, 2007.
- [86] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):369–382, 2008.
- [87] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances*

- in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, USA, 2004.
- [88] R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *11th IEEE International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil, 2007.
 - [89] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *9th IEEE International Conference on Computer Vision (ICCV 2003)*, volume 1, pages 709–715, 2003.
 - [90] G. Yang and T. S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.
 - [91] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.
 - [92] K. C. Yow and R. Cipolla. Detection of human faces under scale, orientation and viewpoint variations. In *2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, pages 295–300, 1996.
 - [93] K. C. Yow and R. Cipolla. Scale and orientation invariance in human face detection. In *Proceedings of the British Machine Vision Conference 1996*, 1996.
 - [94] K. C. Yow and R. Cipolla. Feature-based human face detection. *Image Vision Comput.*, 15(9):713–735, 1997.
 - [95] K. C. Yow and R. Cipolla. Enhancing human face detection using motion and active contours. In *Third Asian Conference on Computer Vision (ACCV '98)*, volume 1, pages 515–522, 1998.