AdaBoost and Histograms for Fast Face Detection

A N D E R S   J Ö R G E N S E N

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2006

AdaBoost and Histograms for Fast Face Detection

A N D E R S   J Ö R G E N S E N

Master's Thesis in Computer Science (20 credits)
at the School of Engineering Physics
Royal Institute of Technology year 2006
Supervisor at CSC was Jan-Olof Eklundh
Examiner was Jan-Olof Eklundh

TRITA-CSC-E 2006:110
ISRN-KTH/CSC/E--06/110--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

A popular learning method for solving classification problems is AdaBoost, which combines an ensemble of weak classifiers into a strong classifier. In this report the real version of AdaBoost is used on the face detection problem and the weak classifiers are modeled by simple histograms.

A face detector of the Viola-Jones type is built to detect frontal faces in unconstrained images. The real-time performance of the system is attributed to how AdaBoost selects a small number of critical visual features and how more and more complex classifiers are combined in a cascade.

Results are demonstrated on a real-world test set and show that similar detection rates can be achieved with less than half the number of weak classifiers needed for standard AdaBoost.

Snabb detektion av ansikten med AdaBoost och histogram

Sammanfattning

En populär inlärningsmetod för klassificeringsproblem är AdaBoost, som kombinerar en grupp av svaga klassificerare till en robust klassificerare. I denna rapport tillämpas den reella versionen av AdaBoost på problemet att detektera ansikten, där de svaga klassificerarna representeras av histogram.

En ansiktsdetektor av Viola-Jones-typ har skapats för att detektera bilder av ansikten tagna framifrån. Systemets snabba beteende beror på metoden AdaBoost väljer ut ett fåtal utslagsgivande visuella kännetecken, samt hur allt mer komplexa klassificerare är ordnade i en serie.

Resultaten demonstreras genom en mängd autentiska bilder och visar att likvärdiga detektionsförhållanden kan uppnås med färre än hälften av antalet svaga klassificerare som behövs för standardversionen av AdaBoost.

Contents

Chapter 1

Introduction

Face detection systems identify faces in images and video sequences using computers. An ideal face detection system should be able to identify and locate all faces regardless of their positions, scale, orientation, lightning conditions, expressions and so on. Due to the large intra-class variations in facial appearances, face detection has been a challenging problem in the field of computer vision.

There are many closely related problems with numerous applications of face detection:

- *Face localisation* aims to determine image position of a single face; a simplified detection problem with the assumption that an input image contains only one face.
- *Facial feature detection* is to detect the presence and location of features, such as eyes, nose, nostrils, eyebrows, mouth, lips, ears, etc., with the assumption that there is only one face in an image.
- *Face recognition* or *face identification* compares an input image against a database and reports a match, if any.
- *Face authentication* verifies the claim of the identity of an individual in an input image.
- *Face tracking* methods continuously estimate the location and possibly the orientation of a face in an image sequence in real time.
- *Facial expression recognition* concerns identifying the affective states (happy, sad, disgusted, etc.) of humans.

Evidently, face detection is the first step in any automated system which solves the above problems and a robust and effective face detector system is essential.

Face detection can be performed based on several different cues: skin colour (for faces in color images), motion (for faces in videos), facial/head shape and facial appearances, or a combination of them. However, detecting faces in black and white, still images with unconstrained, complex backgrounds is a complicated task. So far learning-based approaches have been most effective and have therefore attracted much attention the last years. Recently, Viola and Jones [18] introduced an impressive face detection system capable of detecting frontal-view faces in real time. The desirable properties are partly attributed to the used AdaBoost learning algorithm.

AdaBoost, from adaptive boosting, was rapidly made popular in the machine learning community when it was presented by Freund and Schapire [2] about 10 years ago. The AdaBoost algorithm sequentially constructs a classifier as a linear combination of “weak” classifiers. More recently attention has shifted to a refinement of the original *Discrete AdaBoost*. One such example is the *Real AdaBoost* algorithm, by Schapire and Singer [13], which incorporates a measure of confidences to the predictions of each weak classifier.

1.1 Problem Description

This Master’s project was performed at the *Center for Machine Perception* (CMP) at the *Czech Technical University* in Prague. At CMP, face detection is one of the main topics of the on-going research. Based mainly on the work of Viola and Jones a face detector program using Discrete AdaBoost has been developed and some improvements have been suggested, e.g. [15, 16].

Based on the results by Schapire and Singer [13] the task of this Master’s project is to implement and evaluate the Real AdaBoost algorithm on the face detection problem using histograms to model the weak classifiers. Special attention has been payed to how this affects the complexity of the classifiers and the effectiveness of the system.

1.2 Overview

The ensuing chapters of this report are organised as follows: Chapter 2 explains the AdaBoost algorithm and how its discrete and real versions are derived by minimising the training error. Chapter 3 introduces the face detection problem and describes the adopted face detector system used. Chapter 4 describes the setup of the experiments carried out and the results are presented in Chapter 5. Finally, Chapter 6 summarises the results and the conclusions of the report.

1.3 Acknowledgement

This project was carried out on a scholarship from the research exchange program between CVAP, KTH and CMP, CTU founded by an institutional grant from STINT, The Swedish Foundation for International Cooperation in Research and Education. This support is gratefully acknowledged.

Chapter 2

AdaBoost

In many problem domains, combining the predictions of several models often results in a model with improved performance. Boosting is one such method that has shown great promise.

The AdaBoost algorithm is a relatively new algorithm proposed by Freund and Schapire [2]. It is a descendant of the weighted majority algorithm by Littlestone and Warmuth [8] and the boost-by-majority algorithm by Freund [1]. All three algorithms construct an ensemble of classifiers and use a voting mechanism for the classification. In a wide variety of classification problems, their weighting scheme and final classifier merge have proven to be an efficient method for reducing bias and variance, and improving misclassification rates.

2.1 The Boosting Algorithm

AdaBoost takes as input a training set $\mathcal{S} = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ where each instance, x_i, belongs to a domain or instance space $\mathcal{X}$, and each label y_i belongs to a finite label space $\mathcal{Y}$. Here we will only focus on the binary case when $\mathcal{Y} = \{-1, +1\}$.

Each round, $t = 1, \ldots, T$, AdaBoost calls a given *weak* or *base learning algorithm* which accepts as an input a sequence of training examples $\mathcal{S}$ along with a distribution or set of weights over the training example, $D_t(i)$. Given such an input the weak learner computes a weak classifier, h_t. In general, h_t has the form $h_t : \mathcal{X} \to \mathbb{R}$. We interpret the sign of $h_t(x)$ as the predicted label to be assigned to instance x. Once the weak classifier has been received, AdaBoost chooses a parameter $\alpha_t \in \mathbb{R}$ that intuitively measures the importance that it assigns to h_t.

The idea of boosting is to use the weak learner to form a highly accurate prediction rule by calling the weak learner repeatedly on different distributions over the training examples. Initially, all weights are set equally, but each round the weights of incorrectly classified examples are increased so that those observations that the previously classifier poorly predicts receive greater weight on the next iteration.

A generalised version of Freund and Schapire's AdaBoost algorithm is shown in Algorithm 1.

Algorithm 1 A generalised version of AdaBoost

```
Given $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
Iterate $t = 1, \ldots, T$:
    1. Train weak learner using distribution $D_t$
    2. Get weak classifier $h_t : \mathcal{X} \to \mathbb{R}$
    3. Choose $\alpha_t \in \mathbb{R}$
    4. Update:
           $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
       where $Z_t$ is a normalisation factor (chosen so that $D_{t+1}$ will be a distribution).
Output the final classifier:
    $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

2.2 Analysis

As Algorithm 1 shows, the weights $D_t(i)$ are updated and normalised on each round. The normalisation factor takes the form

$$Z_t = \sum_{i=1}^{m} D_t(i) \mathrm{e}^{-\alpha_t y_i h_t(x_i)}$$

and it can be verified that Z_t measures exactly the ratio of the new to the old value of the exponential sum

$$\sum_{i=1}^{m} \exp\Big(-y_i \sum_{j=1}^{t} \alpha_j h_j(x_i)\Big)$$

on each round, so that $\prod_t Z_t$ is the final value of this sum. We will see below that this product plays a fundamental role in the analysis of AdaBoost.

2.2.1 Training Error

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error, i.e. the fraction of mistakes on the training set. If we let

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

so that $H(x) = \operatorname{sign}(f(x))$. Also, for any predicate π, let $[\![\pi]\!]$ be 1 if π holds and 0 otherwise. Then the following bound on the training error of H holds [13]:

Theorem 2.1. *Assuming the notation in Algorithm 1, the following bounds holds on the training error of H:*

$$\frac{1}{m}\sum_{i=1}^{m} [\![H(x_i) \neq y_i]\!] \leq \prod_{t=1}^{T} Z_t.$$

Proof. By unraveling the update rule, we have that

$$\begin{aligned} D_{T+1}(i) &= \frac{\exp\left(-\sum_t \alpha_t y_i h_t(x_i)\right)}{m \prod\limits_t Z_t} \\ &= \frac{\exp\left(-y_i f(x_i)\right)}{m \prod\limits_t Z_t}. \end{aligned} \tag{2.1}$$

Moreover, if $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $\exp(-y_i f(x_i)) \geq 1$. Thus,

$$[\![H(x_i) \neq y_i]\!] \leq \exp(-y_i f(x_i)). \tag{2.2}$$

Combining Eqs. (2.1) and (2.2) gives the stated bound on training error, since

$$\begin{aligned} \frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \\ &= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i) \\ &= \prod_t Z_t. \end{aligned}$$

$\square$

The important consequence of Theorem 2.1 is that, in order to minimise the training error, a reasonable approach might be to greedily minimise the bound given in the theorem by minimising Z_t on each round of boosting. Following the idea by Schapire and Singer [13] we will see two different ways of choosing the weights α_t and the weak classifier, h_t.

2.2.2 Choosing Parameters for Discrete AdaBoost

In Freund and Schapire's [2] original Discrete AdaBoost the algorithm each round selects the weak classifier, h_t, that minimises the weighted error

$$\epsilon_t = \sum_i D_t(i)[\![h_t(x_i) \neq y_i]\!] = \sum_i D_t(i) \left(\frac{1 - y_i h_t(x_i)}{2} \right) \tag{2.3}$$

on the training set.

We will here show how to choose the coefficients, α_t, when the weak classifiers are restricted to the discrete values $\{-1, +1\}$. Recall that we want to minimise Z_t, which we can rewrite as:

$$\begin{aligned} Z_t &= \sum_i D_t(i) \mathrm{e}^{-\alpha_t y_i h_t(x_i)} \\ &= \sum_i D_t(i) \left(\frac{1 + y_i h_t(x_i)}{2} \mathrm{e}^{-\alpha_t} + \frac{1 - y_i h_t(x_i)}{2} \mathrm{e}^{\alpha_t} \right). \end{aligned} \tag{2.4}$$

Using Eq. (2.3) we can analytically choose α_t by minimising this expression, which yields:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{2.5}$$

Plugging into Eq. (2.4) this choice gives

$$Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}.$$

We have thus proved the following corollary of Theorem 2.1:

Corollary 2.1 (Freund & Schapire)**.** *Assume each $h_t \in \{-1, +1\}$ and that we choose*

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

where

$$\epsilon_t = \Pr_{i \sim D_t}[h_f(x_i) \neq y_i] = \sum_i D_t(i)[\![h_t(x_i) \neq y_i]\!].$$

Then the training error of the final classifier, H, is as most

$$2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1 - \epsilon_t)}.$$

Algorithm 2 Discrete AdaBoost

```
Given $(x_1, y_1), \ldots, (x_m, y_m)$   where   $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
Iterate $t = 1, \ldots, T$:
    1. Find $h_t = \arg\min_{h_j} \epsilon_j$   where   $\epsilon_j = \sum_{i=1}^{m} D_t(i)[\![h_t(x_i) \neq y_i]\!]$
    2. Set $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
    3. Update $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
Output final classifier:
    $H(x) = \text{sign}\left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$
```

The bound on the error given in Corollary 2.1 can also be written in the form

$$\epsilon_{\text{train}} \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2 \right)$$

where we define $\gamma_t = 1/2 - \epsilon_t$. Thus if each base classifier is slightly better than random so that $\gamma_t \geq \gamma$ for some $\gamma > 0$, then the training error drops exponentially fast in T. This bound, combined with the bounds on generalisation error given below, proves that AdaBoost is indeed a boosting algorithm in the sense that it can efficiently convert a weak learning algorithm into a strong learning algorithm, which can generate a hypothesis with an arbitrary low error rate, given sufficient data.

We summarise the results from this section in Algorithm 2.

2.2.3 Domain-partitioning Weak Classifiers

As we have seen, the early studies of AdaBoost focused on finding a weak classifier with a small number of errors with respect to the given distribution over the training samples. Theorem 2.1 suggests, however, that a different criterion can be used. Following Schapire and Singer [13], we can instead attempt to greedily minimise the upper bound on the training error by minimising Z_t on each round. Thus, the weak learner should attempt to find a weak classifier, h_t, which minimises

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)).$$

This expression can be simplified by folding α_t into h_t. In other words, by assuming, without loss of generality, that the weak learner can freely scale any weak classifier, h_t, by any constant factor $\alpha_t \in \mathbb{R}$. Then the weak learner's goal now is to minimise

$$Z_t = \sum_i D_t(i) \exp(-y_i h_t(x_i)). \tag{2.6}$$

Different algorithms, i.e. gradient-based algorithms, can be modified to minimise this loss function directly. We will here show how histograms can be used based on the criterion for finding good weak classifiers which make their predictions based on a partitioning of the domain $\mathcal{X}$. To be more specific, each such classifier is associated with a partition of $\mathcal{X}$ into disjoint blocks $X_1, \ldots, X_N$ which cover all of $\mathcal{X}$ and for which $h(x) = h(x')$ for all $x, x' \in X_j$, where we from now on omit the subscript t when clear from context. In other words, h's prediction depends only on which block X_j a given instance falls into.

Suppose that we have a partition $X_1, \ldots, X_N$ of the space. The question is what prediction should be made for each block of the partition. That is, how to find a function $h : \mathcal{X} \to \mathbb{R}$, which respects the given partition and minimises Eq. (2.6).

Let $c_j = h(x)$ for $x \in \mathcal{X}_j$. The goal is to find appropriate choices for c_j. For each j and for $b \in \{-1, +1\}$, let

$$W_b^j = \Pr{}_{i \sim D}[x_i \in X_j \wedge y_i = b] = \sum_{i=1}^{m} D(i) [\![x_i \in X_j \wedge y_i = b]\!]$$

be the weighted fraction of examples which fall in block j with label b. Then Eq. (2.6) can be rewritten

$$\begin{aligned} Z &= \sum_j \sum_{i: x_i \in X_j} D(i) \exp(-y_i c_j) \\ &= \sum_j \left(W_{+1}^j \mathrm{e}^{-c_j} + W_{-1}^j \mathrm{e}^{c_j} \right), \end{aligned} \tag{2.7}$$

which is minimised when

$$c_j = \frac{1}{2} \ln \left(\frac{W_{+1}^j}{W_{-1}^j} \right). \tag{2.8}$$

Plugging into (2.7) gives

$$Z = 2 \sum_j \sqrt{W_{+1}^j W_{-1}^j}. \tag{2.9}$$

Note that the sign of c_j is equal to the majority class within block j and c_j will be close to zero if there is a roughly equal split of positive and negative examples in block j. Likewise, c_j will be far from zero if one label strongly predominates.

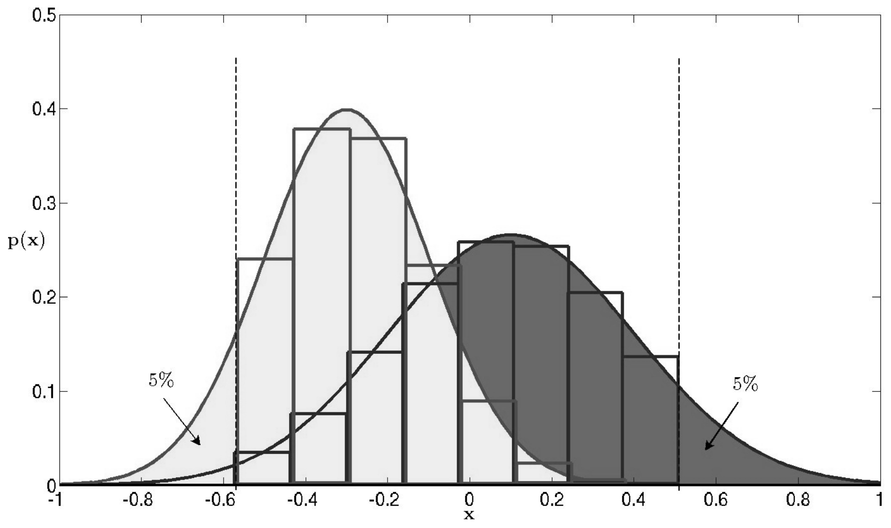


Figure 2.1: *Example of the distribution of two general classes. The prediction of the weak classifier for each block is based on the ratio the two bins within that block. Instances outside the interval of focus are predicted as in the closest block.*

In many real world situations, instances of one class often strongly dominates at the end of the domain. Therefore, it is often better to focus on the subspace where the data is distributed more evenly. In the experiments presented later 5% of the data at each end of the domain has been rejected and histograms generated on the rest of the data. This is depicted in Figure 2.1 and the Real AdaBoost algorithm is summarised in Algorithm 3.

Smoothing the predictions The scheme presented above requires that we predict as in Eq. (2.8) on block j. It may well happen that W^j_{+1} or W^j_{-1} is very small or even zero, in which case c_j will be very large or infinite in magnitude. In practise, such large predictions may cause numerical problems. In addition, there may be theoretical reasons to suspect that large, overly confident predictions will increase the tendency to over-fit.

To limit the magnitudes of the predictions we used the smoothed values

$$c_j = \frac{1}{2}\ln\left(\frac{W^j_{+1}+\varepsilon}{W^j_{-1}+\varepsilon}\right) \tag{2.10}$$

for some appropriately small positive value of ε. Because W^j_{+1} and W^j_{-1} are both

```
Algorithm 3 Domain-partitioned Real AdaBoost
Given $(x_1, y_1), \ldots, (x_m, y_m)$   where   $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
Iterate $t = 1, \ldots, T$:
    1. Find a partition, $X_1, \ldots, X_N$, of the domain $\mathcal{X}$
    2. Choose $h_t(x) = \frac{1}{2} \ln \frac{P(y = +1|x, D_t)}{P(y = -1|x, D_t)}$
       according to Eq. (2.8)
    3. Update $D_{t+1}(i) = D_t(i) \exp(-y_i h_t(x_i))$
       and normalise so that $\sum_{i=1}^{m} D_{t+1}(i) = 1$
Output final classifier:
    $H(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right)$
```

bounded between 0 and 1, this has the effect of bounding $|c_j|$ by

$$\frac{1}{2} \ln\left(\frac{1+\varepsilon}{\varepsilon}\right) \approx \frac{1}{2} \ln(1/\varepsilon).$$

Moreover, this smoothing only slightly weakens the value of Z since, plugging Eq. (2.10) into Eq. (2.7) gives

$$\begin{aligned}
Z &= \sum_j \left(W_{+1}^j \sqrt{\frac{W_{-1}^j + \varepsilon}{W_{+1}^j + \varepsilon}} + W_{-1}^j \sqrt{\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon}} \right) \\
&\leq \sum_j \left(\sqrt{W_{+1}^j (W_{-1}^j + \varepsilon)} + \sqrt{W_{-1}^j (W_{+1}^j + \varepsilon)} \right) \\
&\leq \sum_j \left(2\sqrt{W_{+1}^j W_{-1}^j} + \sqrt{\varepsilon W_{+1}^j} + \sqrt{\varepsilon W_{-1}^j} \right) \\
&\leq 2 \sum_j \sqrt{W_{+1}^j W_{-1}^j} + \sqrt{2N\varepsilon}. \qquad (2.11)
\end{aligned}$$

In the second inequality, $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ for $x, y \geq 0$ is used and in the last inequality we used the fact that

$$\sum_j (W_{+1}^j + W_{-1}^j) = 1 \qquad \Rightarrow \qquad \sum_j \left(\sqrt{W_{+1}^j} + \sqrt{W_{-1}^j} \right) \leq \sqrt{2N},$$

where N is the number of blocks in the partition.

Thus comparing Eqs. (2.9) and (2.11) we see that Z will not be greatly degraded by smoothing if we choose $\varepsilon \ll 1/(2N)$. In the experiments presented later ε of the order of $1/(2mN)$, where m is the number of training examples, is used.

2.2.4 Boosting and Logistic Regression

Classification generally is the problem of predicting the label y of an instance x with the intention of minimising the probability of an incorrect prediction. However, it is often useful to estimate the probability of a particular label. In statistics this problem has been extensively studied and *logistic regression models* are often used.

A recent paper of Friedman et al. [3] presents a statistical interpretation of the AdaBoost algorithm. In particular, they show how boosting algorithms result from building additive models using Newton updates of the exponential loss function, making a comparison between boosting and stepwise logistic regression methods. Their ideas were formalised in the *LogitBoost* algorithm.

We will in this section see, following the work from Friedman et al. [3], how the discrete and real versions of AdaBoost are related to logistic regression. We begin with a review of the logistic regression model.

The Logistic Regression Model As before, let $\mathcal{X}$ and $\mathcal{Y}$ be spaces of instances and labels, respectively. Although the overall goal still is classification, we focus on estimating probabilities which can be converted into classifications in the obvious way by thresholding. That is, given training data, we wish to build a rule that estimates the conditional probability that $y = +1$ given x when test example (x, y) is chosen according to some distribution $\mathcal{D}$. In logistic regression this is done by building a real-valued function $F : \mathcal{X} \to \mathbb{R}$ and estimate $P(y = +1|x)$ by $\sigma(F(x))$. Friedman et al. suggest using the symmetric logistic function

$$\sigma(z) = \frac{\mathrm{e}^z}{\mathrm{e}^z + \mathrm{e}^{-z}}.$$

In the sense of additive regression models $F(x)$ is a linear combination of base functions

$$F(x) = \sum_{j=1}^{M} f_j(x).$$

With such a model we attempt to find $F(x)$ by maximising the conditional likelihood of the data, or equivalently, minimising the negative log conditional likelihood

$$\sum_i \ln\left(1 + \mathrm{e}^{-2y_i F(x_i)}\right). \tag{2.12}$$

As we have seen in earlier section AdaBoost attempts to minimise another loss function, namely:

$$\sum_i \mathrm{e}^{-y_i F(x_i)}. \tag{2.13}$$

It can be verified that Eq. (2.12) is upper bounded by Eq. (2.13). In addition, if we add the constant $1 - \ln 2$ to Eq. (2.12), which does not affect its minimisation, the resulting function and Eq. (2.13) have identical Taylor expansions around zero up to second order; thus, their behavior near zero is very similar. Finally, it can be shown that, for any distributions over pairs (x, y), the expectations

$$E\left[\ln\left(1 + \mathrm{e}^{-2yF(x)}\right)\right]$$

and

$$E\left[\mathrm{e}^{-yF(x)}\right]$$

are minimised by a the same function $F(x)$. Lemma 2.1 shows that this function $F(x)$ is the symmetric logistic transform of $P(y = 1|x)$.

Lemma 2.1. $\sigma(F(x)) = E[\mathrm{e}^{-yF(x)}]$ *is minimised at*

$$F(x) = \frac{1}{2} \ln \frac{P(y = 1|x)}{P(y = -1|x)}$$

Hence

$$\begin{aligned} P(y = 1|x) &= \frac{\mathrm{e}^{F(x)}}{\mathrm{e}^{-F(x)} + \mathrm{e}^{F(x)}} \\ P(y = -1|x) &= \frac{\mathrm{e}^{-F(x)}}{\mathrm{e}^{-F(x)} + \mathrm{e}^{F(x)}} \end{aligned}$$

Proof. While E entails expectations over the joint distributions of y and x, it is sufficient to minimise the criterion conditional on x.

$$\begin{aligned} E[\mathrm{e}^{-yF(x)}|x] &= P(y = 1|x)\mathrm{e}^{-F(x)} + P(y = -1|x)\mathrm{e}^{F(x)} \\ \frac{\partial E[\mathrm{e}^{-yF(x)}|x]}{\partial F(x)} &= -P(y = 1|x)\mathrm{e}^{-F(x)} + P(y = -1|x)\mathrm{e}^{F(x)} \end{aligned}$$

Setting the derivative to zero the expression for $F(x)$ follows.

The conditional probabilities follows by some manipulation of the expression for $F(x)$ and the fact that $P(y = 1|x) + P(y = -1|x) = 1$. □

We will now state and prove two theorems from Friedman et al. [3] which shows that Discrete and Real AdaBoost are stage-wise estimation procedures for fitting an additive logistic regression model.

Theorem 2.2. *The Discrete AdaBoost in Algorithm 2 fits an additive logistic regression model by using adaptive Newton updates for minimising* $E[\mathrm{e}^{-yF(x)}]$.

Proof. Let $\sigma(F(x)) = E[\mathrm{e}^{-yF(x)}]$. Suppose we have a current estimate $F(x)$ and seek an improved estimate $F(x)+cf(x)$. For fixed c and x, expand $\sigma(F(x)+cf(x))$ to second order about $f(x)=0$:

$$\begin{aligned}\sigma(F+cf) &= E[\mathrm{e}^{-y(F(x)+cf(x))}]\\ &\approx E[\mathrm{e}^{-yF(x)}(1-ycf(x)+c^2f(x)^2/2)]\end{aligned}$$

Minimising point-wise with respect to $f(x) \in \{-1, +1\}$, we find

$$\begin{aligned}\hat{f}(x) &= \arg\min_f E_w[1-ycf(x)+c^2f(x)^2/2|x]\\ &= \arg\min_f E_w[(y-cf(x))^2|x)]\\ &= \arg\min_f E_w[(y-f(x))^2|x]\end{aligned}$$

where $w(y|x) = \exp(-yF(x)/E[\exp(-yF(x)])$ and the last equality follows by considering the two possible choices for $f(x)$. Thus, minimising a quadratic approximation to the criterion leads to a weighted least-squares choice of $f(x) \in \{-1, +1\}$ and this constitutes the Newton step.

Given $\hat{f}(x) \in \{-1, +1\}$, we can directly minimise $\sigma(F+c\hat{f})$ to determine c:

$$\begin{aligned}\hat{c} &= \arg\min_c E_w[\mathrm{e}^{-cy\hat{f}(x)}]\\ &= \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)\end{aligned}$$

where $\epsilon = E_w\left[[\![y \neq \hat{f}(x)]\!]\right]$.

Combining these steps we get the update for $F(x)$

$$F(x) \leftarrow F(x) + \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)\hat{f}(x)$$

In the next iteration the new contribution $\hat{c}\hat{f}(x)$ to $F(x)$ augments the weights:

$$w(y|x) \leftarrow w(y|x)\mathrm{e}^{-\hat{c}\hat{f}(x)y},$$

followed by a normalisation. Since $y\hat{f}(x) = 2 \times [\![y \neq \hat{f}(x)]\!] - 1$, we see that the update is equivalent to

$$w(y|x) \leftarrow w(y|x)\exp\left(\ln\left(\frac{1-\epsilon}{\epsilon}\right)[\![y \neq \hat{f}(x)]\!]\right)$$

Thus the function and weight updates are identical to those used in Discrete AdaBoost.

□

Theorem 2.3. *The Real AdaBoost algorithm fits an additive logistic regression model by stage-wise optimisation of* $\sigma(F(x)) = E[\mathrm{e}^{-yF(x)}]$.

Proof. Suppose we have a current estimate $F(x)$ and seek an improved estimate $F(x) + f(x)$ by minimising $\sigma(F(x) + f(x))$:

$$\begin{aligned}\frac{\partial\sigma(F(x)+f(x))}{\partial f(x)} &= -E\left[\mathrm{e}^{-yF(x)}y\mathrm{e}^{-yf(x)}|x\right]\\ &= -E\left[\mathrm{e}^{-yF(x)}[\![y=1]\!]\mathrm{e}^{-f(x)}|x\right] + E\left[\mathrm{e}^{-yF(x)}[\![y=-1]\!]\mathrm{e}^{f(x)}|x\right]\end{aligned}$$

Dividing through by $E\mathrm{e}^{-yF(x)}$ and setting the derivative to zero yields

$$\begin{aligned}\hat{f}(x) &= \frac{1}{2}\ln\frac{E_w([\![y=1]\!]|x)}{E_w([\![y=-1]\!]|x)}\\ &= \frac{1}{2}\ln\frac{P_w(y=1|x)}{P_w(y=-1|x)}\end{aligned}$$

where $w(y|x) = \exp(-yF(x))/E[\exp(-yF(x))|x]$.

$\square$

2.2.5 Generalisation Error

In learning problems like pattern classification and regression, we are of course interested in performance on examples *not* seen during training, i.e. in the generalisation error. A considerable amount of effort has been spent on obtaining good error bounds. Typically, such bounds take the form of a sum of two terms: some sample-based estimate of performance and a penalty term that is large for more complex models.

A classic theory for error bounds of binary classification methods was developed by Vapnik and Chervonenkis [17]. Their results relates the empirical classification error of a binary classifier, H, to the probability of the error $\Pr[y \neq H(x)]$, where $\Pr[\cdot]$ denotes the empirical probability on the training sample.

For AdaBoost, two methods of analysing the generalisation error have been proposed in the literature. In the track of Vapnik and Chervonenkis's theory, Freund and Schapire [2] showed how to bound the generalisation error of the final classifier in terms of its training error, the size m of the sample, the VC-dimension[1] d of the base classifier space and the number of rounds T of boosting. Specifically, the generalisation error, with high probability, is at most

$$\Pr[H(x) \neq y] + \mathcal{O}\left(\sqrt{\frac{Td}{m}}\right).$$

This bound suggests that boosting will over-fit if run for too many rounds, i.e. as T becomes too large. However, initial experiments observed indicate that AdaBoost

[1]The Vapnik-Chervonenkis (VC) dimension is a standard measure of the "complexity" of a space of binary functions.

tends not to over-fit, even when run for hundreds of rounds of boosting, the generalisation error continues to drop, or at least not increase, long after the training error had reached zero, clearly contradicting the spirit of the bound above.

In response to these empirical findings, Schapire et al. [12], gave an alternative analysis in terms of the *margins* achieved by the final classifier on the training examples. The margin of a labeled example (x, y) is defined to be $yf(x)$ and is positive if and only if H makes a correct prediction on the example. We further regard the magnitude of the margin as a measure of the “confidence” of H’s prediction. Schapire et al. show that larger margins imply lower generalisation error, regardless of the number of rounds. Moreover, they show that AdaBoost tends to increase the margins of the training examples.

In a first step to outline their work, we assume that each weak classifier h_t has bounded range. Recall that the final classifier has the form

$$H(x) = \operatorname{sign}(f(x)) \quad \text{where} \quad f(x) = \sum_t \alpha_t h_t(x).$$

Since the h_t’s are bounded and since we only care about the sign of f, we can re-scale the h_t’s and normalise the α_t’s allowing us to assume without loss of generality that each $h_t : \mathcal{X} \to [-1, +1]$, each $\alpha_t \in [0, 1]$ and $\sum_t \alpha_t = 1$. Let us also assume that each h_t belongs to a classifier space $\mathcal{H}$.

Schapire et al.’s result can be applied only in the special case that each $h \in \mathcal{H}$ has range $\{-1, +1\}$. However, Schapire and Singer [13] extends the theory to allow the weak classifier to be real-valued, which is of our interest.

Let us define d to be the *pseudo-dimension* of $\mathcal{H}$ and use $\Pr_{\mathcal{D}}[\cdot]$ to denote the probability of an event when the example (x, y) is chosen according to the distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$, and $\Pr_{\mathcal{S}}[\cdot]$ to denote the probability with respect to choosing an example uniformly at random from the training set. It can then be proofed that, with high probability, the generalisation error satisfies the following bound for all $\theta > 0$:

$$\Pr_{\mathcal{D}}[yf(x) \le 0] \le \Pr_{\mathcal{S}}[yf(x) \le \theta] + \mathcal{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

See for example Schapire and Singer [13] for a comprehensive proof. Note that this bound is entirely independent of T, the number of rounds of boosting. In addition, Schapire et al. proved that boosting is particularly aggressive at reducing the margin since it concentrates on the examples with the smallest margins, whether positive or negative.

Experiments have shown that as the margin increases, the generalisation performance becomes better on data sets with almost no noise. However, this performance has not been confirmed for noisy data, which has at least one of the following properties: 1) overlapping class probability distributions, 2) outliers, or 3) mislabeled patterns. All these types of noise appear very often in data analysis and implies that AdaBoost exhibits suboptimal generalisation ability and over-fitting behavior for such data.

2.3 Simulation Studies

In this section the performance of Real AdaBoost with different complexities of the histograms are compared on an artificially constructed problem.

The synthetic data used for the experiments consists of two randomly generated classes, where each instance, $\boldsymbol{x}_i$, belongs to the domain $\mathcal{X} \subseteq \mathbb{R}^2$. The data set for the first class was generated from a $N(\mathbf{0}, \boldsymbol{I})$ distribution, while each instance $\boldsymbol{x} = (r\cos\theta, r\sin\theta)$ from the second class was generated using $r \in N(4, 1)$ and $\theta \in U(0, 2\pi)$.

The domain is partitioned in 13 different ways; horizontally, vertically, diagonally and 9 radial partitions with different centre points. At each round of AdaBoost, histograms with N bins are generated for every partition and the predictions within each block is computed according to Eq. (2.10). The partition which minimises Z as in Eq. (2.11) is then chosen as weak classifier for that round. An example illustrating the data sets and the partitioned domain is shown in Figure 2.2, where a decision function is built up by Real AdaBoost to classify two sets of 100 data points.

Figure 2.3 compares the training and test error rates for Real AdaBoost with 4, 8 and 16 bins in the histogram, respectively. 1000 data points for the two classes were generated and AdaBoost was run 20 times and the results were averaged.

From Figure 2.3 it seems that using more complex histograms yields a better reduction of the training error. However, this fine partition of the space leads to overtraining of the system, something that is not experienced for the coarser partitions.

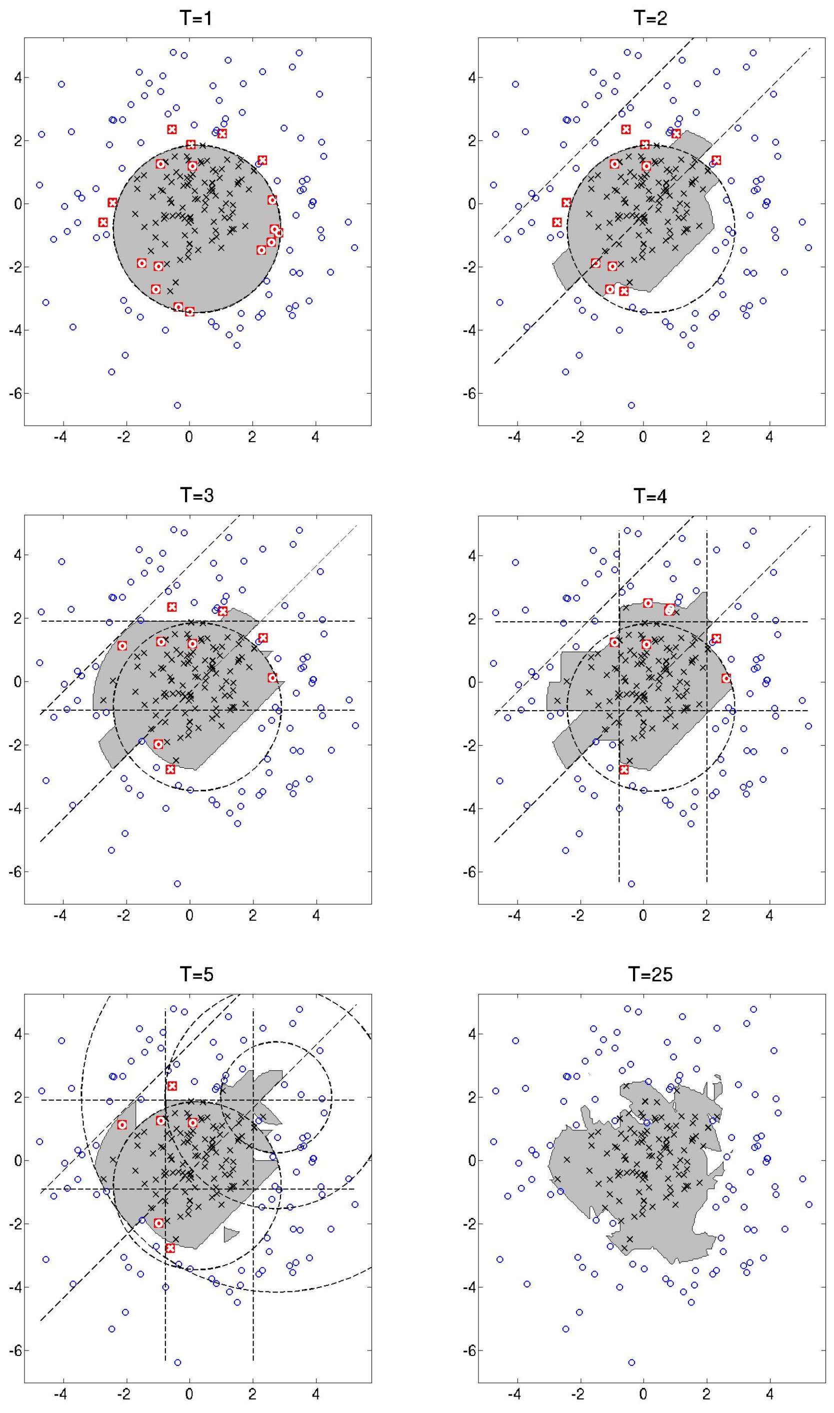
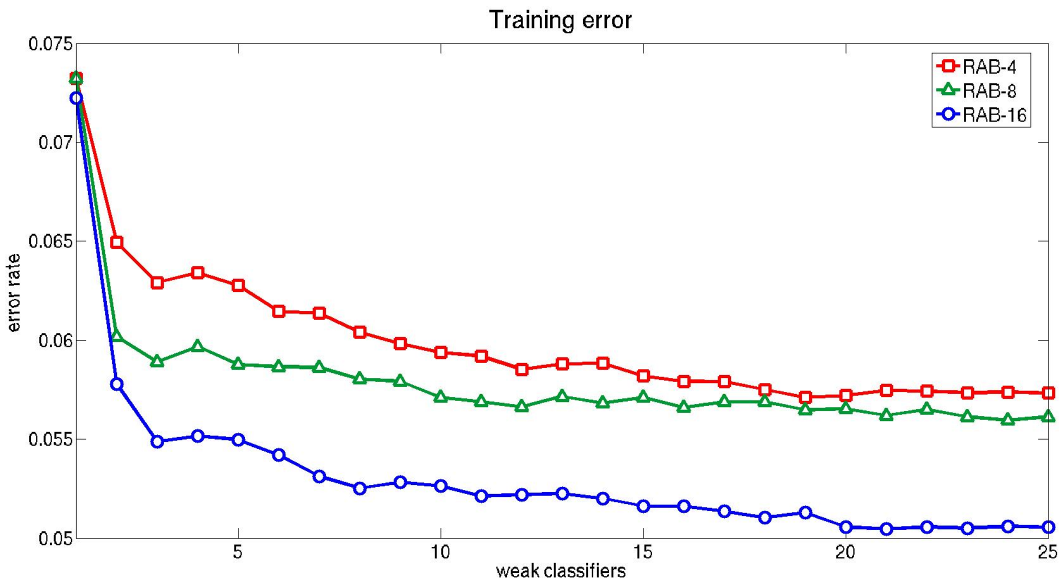


Figure 2.2: *Example of binary classification using Real AdaBoost with $T = 1, 2, 3, 4, 5$ and 25 rounds of boosting. The weak classifiers consist of 13 partitions of the domain, each partitioned into 8 blocks. The dashed lines mark a change of sign of each weak classifier. Misclassified data are highlighted.*



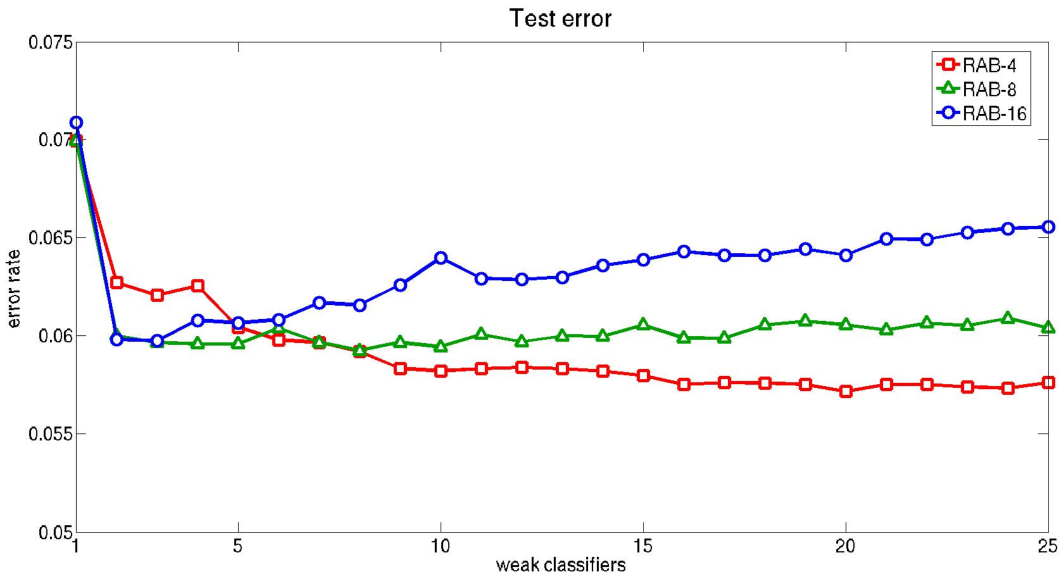


Figure 2.3: *Training error (top) and test error (bottom) on a toy problem using Real AdaBoost and histograms with 4, 8 and 16 bins, respectively. The results are averaged over 20 rounds.*

Chapter 3

Face Detection

Images containing faces are essential to intelligent vision-based human computer interaction and research efforts in face processing include face recognition, face tracking, pose estimation and expression recognition. However many reported methods assume that the faces in an image have been identified and localised. To build fully automated systems that analyse the information contained in face images, robust and efficient face detection algorithms are required.

Given a single image the goal of face detection is to identify all image regions which contain a face regardless of its position, orientation and the lightning condition. Such a problem is challenging because faces are nonrigid and have a high degree of variability in scale, location, orientation (up-right, rotated) and pose (frontal, profile). Facial expression, occlusion and lightning conditions also change the overall appearances of faces.

Two important characteristics for a trained face detector are its detection and error rates. The *detection rate* is defined as the ratio between the number of faces correctly detected and the number of faces determined by a human. In general, two types of errors can occur:

False negatives in which faces are missed, resulting in low detection rate.

False positives in which an image region is declared to be a face, but it is not.

The detection and false positive rates are normally related since one can often tune the parameters of the detection system to increase the detection rates while also increasing the number of false detections.

3.1 Previous research

The face detection problem is pretty old and many algorithms have been proposed. Therefore, only some of the latest approaches often used for comparison will be mentioned here. For a detailed survey of the older methods see e.g. Yang et. al [20].

Rowley et al. [11] built a neural network based face detector. A neural network is trained to classify a 20×20 image. For detection of faces of different scales and at different positions the image is repetitively sub-sampled and scanned throughout every location in every such image. Multiple responses of the detector are then merged. For detection of rotated faces, Rowley proposed a two stage algorithm. First, a neural network determines the rotation of the face (for non-face images, arbitrary rotation is returned), the image is derotated and in the second stage a frontal face detector is run. Similar approach was used for head pose, however better results were obtained by dividing the task into several subproblems for different poses. The Rowley's detector is quite accurate but is very slow. However, the two stage approach is an improvement over the previous approaches scanning the image at every position, scale and rotation.

Schneiderman et al. [14] adopted a fully Bayesian approach. The final decision rule is a simple likelihood ratio test. Both class conditional density functions are modeled as a product of a big number of likelihoods of a single visual attribute where the attributes are assumed independent. The likelihoods are modeled as histograms. The visual attributes used are based on the quantised wavelet coefficients to allow localisation of the attributes in space, scale and orientation. To collect a representative set of the visual attributes the AdaBoost algorithm is used. The detector is able to detect either frontal faces or profiles but is again relatively slow.

Yang et al. [20] use the SNoW architecture to build a classifier. The SNoW (Sparse Network of Winnows) architecture is similar to the perceptron but with very high number of inputs (possibly infinite). Few of them are "active" and the rest "inactive". The inputs correspond to the features in the example images. Measured values in an image determine which input features becomes active. Weights of connections between active inputs and output are summed and thresholded. If a prediction mistake is made the weights are increased or decreased, depending on the type of mistake (missed detection or false alarm). The weighted sum is similar to the AdaBoost algorithm, yet the feature set reduction is not so immense. The speed of the evaluation is higher but still does not allow a real-time performance.

Another very simple and almost directly Bayesian approach to the frontal face detection was proposed by Liu [9]. The face class is modeled as a multivariate normal distribution. The same model is applied to the non-face class, however only for the non-face samples close enough to the face class. Since the class conditional probabilities are known after training, the Bayes' decision rule is applied. For the classification of the rest of the non-face samples a simple threshold on the distance to the face class is used. The important property of the approach is the fast non-face samples classification, however, the detection is rather slow. Nevertheless, the reported results are very good.

The first really real-time face detector was proposed by Viola and Jones [19]. The Viola and Jones frontal face detector consists of several classifiers trained by the AdaBoost algorithm that are organised into a decision cascade. Each cascade stage classifier is set to reach a very high detection rate and an "acceptably" low false positive rate. Since it is trained on the data classified as a face by the previous stages, the final false positive rate is very low and the final detection rate remains high. Besides similar detection rates as the previous approaches, the main advantage of Viola and Jones algorithm is the real-time detection.

Later, Viola and Jones extended their work also to multi-view face detection. The used approach is similar to the work of Rowley et al.. A decision tree is trained to find a head pose and when the pose is known a face detector corresponding to this pose is evaluated. The paper demonstrates that the Viola and Jones approach can be extended to the multi-view face detection without any substantial speed reduction.

The Viola and Jones algorithm was also extended to the multi-view face detection by Li et al. [6]. Instead of using discrete AdaBoost, the real version was used and the algorithm was modified to exclude some of the already found weak classifiers to overcome non-monotonicity problem of the greedy selection method. To detect a face independent of the head pose, a pyramid of coarse to fine detectors is trained. The algorithm runs in real-time.

Only for the works of Viola and Jones, and Li et al. a real-time performance is reported. All other approaches are often more accurate but with a large penalty in speed. The main attribute contributing to the real-time performance of the methods is the sequential classifier evaluation. Even Rowley et al. used a simple version of sequential decision making in their two stage architecture and in the approach by Liu the non-face samples are classified as soon as possible. Nevertheless, even in the approaches performing in real-time, the lack of a deeper theory working properly with the time parameter is evident.

3.2 The Adopted Face Detection Approach

Viola and Jones [19] introduced an impressive face detection system capable of detecting faces in real-time with both high detection rate (about 90%) and very low false positive rates. The desirable properties are attributed especially to the efficiently computable features used, the AdaBoost learning algorithm and the cascade technique adopted for decision making.

3.2.1 Features

The detection procedure classifies images based on the value of simple scalar features. There are many motivations for using features rather than the pixels directly. The most common reason is that features can act to encode ad hoc domain knowledge that is difficult to learn using a finite quantity of training data, another reason is that the feature-based system operate much faster than a pixel-based system.

The features are similar to Haar basis functions. They operate on the grey level images and decision depends on the value of difference of sums computed over rectangular regions. Viola and Jones use three kinds of features depicted in Figure 3.1. The value of a *two-rectangular feature* is the difference between the sums of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent. A *three-rectangular feature* computes the sum within two outside rectangles subtracted from the sum in center rectangle. Finally a *four-rectangular feature* computes the difference between diagonal pairs of

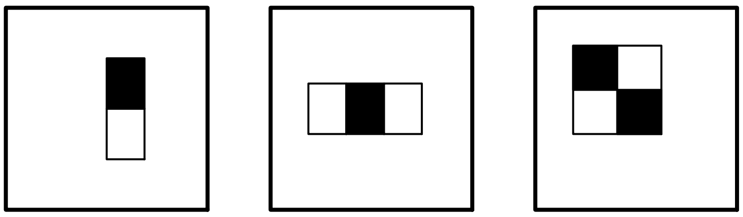


Figure 3.1: *Examples of the Viola and Jones features. The squares represents a face frame. The value of the filter based on a given feature is computed as the sum over filled rectangle(s) minus the sum over empty rectangle(s).*

rectangles. Two other types of features can easily be generated by rotating the first two types by 90°.

Rectangle features can be computed very efficiently by means of an auxiliary image, also referred to as an integral image. The integral image, I, at location $(x,\, y)$ is defined as the sum of the pixels of the rectangle ranging from the top left corner at (0, 0) to the bottom right corner at $(x,\, y)$:

$$I(x,\, y) = \sum_{x' \leq x,\, y' \leq y} i(x',\, y'),$$

where i is the input image in question. The integral image can be computed in one pass over the original image by using the following recurrence relation:

$$I(x,\, y) = I(x,\, y-1) + I(x-1,\, y) + i(x,\, y) - I(x-1,\, y-1)$$

with $I(-1,\, y) = I(x,\, -1) = I(-1,\, -1) = 0$.

Using the integral image any rectangular sum can be computed in four array references, as depicted in Figure 3.2. Since the two-rectangular features defined above involve adjacent rectangular sums they can be computed in only six references, eight in the case of three-rectangle features and nine for four-rectangle features.

As we have seen the scalar feature is a transformation from the n-dimensional image space to the real line. Given that the base resolution of the face detector is 24×24 pixels the image space is 576-dimensional. For each such detector window there are tens of thousands of different features ranging over all scales and positions within the window. Thus, the set of rectangular features is an over-complete set for the intrinsically low-dimensional image pattern x. However, as we will see in the next section only a very small number of these features are needed to form an effective classifier.

3.2.2 Learning the Classifier Function

Given a feature set and a training set of positive and negative images, different machine learning approaches could be used to learn a classification function. The

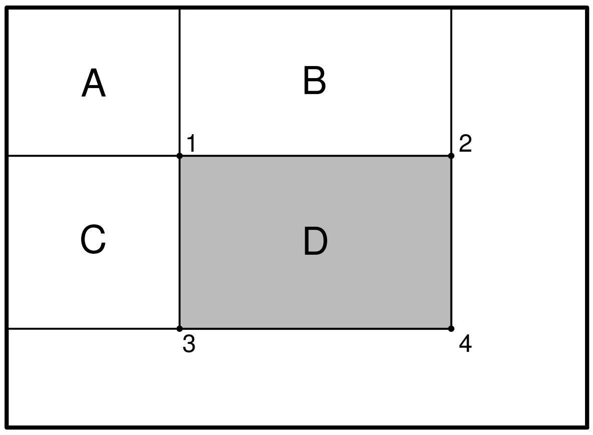


Figure 3.2: *Using the integral image representation of an image the sum of the pixels within rectangle* D *can be computed with only four array references. For example, the value of the integral image at location* 4 *is the sum of the pixels within the rectangles* A + B + C + D. *Thus, the sum within rectangle* D *can be computed as* $4 + 1 - (2 + 3)$.

system of Viola and Jones makes a successful application of the discrete version of AdaBoost to learn the classification function. Specifically, AdaBoost is adapted to solve the following three fundamental problems in one boosting procedure:

Learning effective features from a large feature set. The conventional AdaBoost procedure can be easily interpreted as a greedy feature selection process. Consider the general problem of boosting, in which a large set of classification functions are combined using a weighted majority vote. The challenge is to associate a large weight with each good classification function and a smaller with poor functions. Drawing an analogy between weak classifiers and features, AdaBoost is an effective procedure for searching out a small number of good features.

Constructing weak classifiers each of which is based on one of the selected features. One practical method for completing this analogy is to restrict the weak learner to the set of classification functions each of which depend on a single feature. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples. From the theory in sections 2.2.2 and 2.2.3 this implies choosing the feature which minimise the error ϵ_t or Z_t, for the discrete or real version of AdaBoost, respectively.

Boosting the weak classifiers into a stronger classifier. AdaBoost combines the selected features as a linear combination and provides a strong theoretical backbone for the bound of the training error.

The way Real AdaBoost is adopted to solve these problems is described in Algorithm 4.

3.2.3 Cascade Building

In order to detect a face in an image we need to examine all possible sub-windows and determine whether they contain a face or not. This is done for all possible positions and for different scales. In a regular image of 320×240 pixels there are over 500 000 sub-windows. In order to reduce the total running time of the system we need to radically bound the average time that the system spends on each sub-window. For this purpose, Viola and Jones [19] suggested using a cascade of classifiers. The idea of the cascade is based on the observation that we need very few features to create a classifier that accepts almost 100% of the positive examples while rejecting many (20-50%) of the false examples. Linking many such classifiers one after another will create a cascade of classifiers that separates true from false examples almost perfectly, see Figure 3.3.

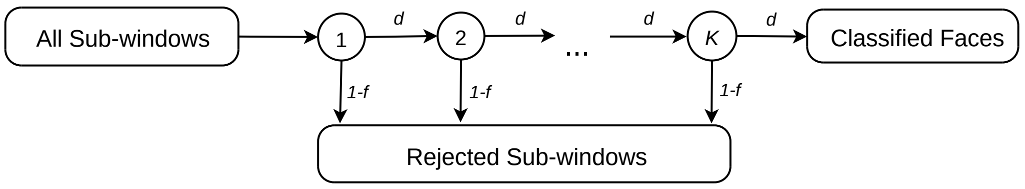


Figure 3.3: *Schematic depiction of a cascade of classifiers with K stages. Each classifier is trained to achieve a detection rate d and a false negative rate f. A sub-window is considered to be a face only if it passes all the K classifiers.*

The structure of the cascade reflects the fact that within any single image an overwhelming majority of sub-windows are negative. As such, the cascade attempts to reject as many negatives as possible at the earliest stage possible. While a positive instance will trigger the evaluation of every classifier in the cascade, this is an exceedingly rare event.

The building process of the cascade is described in Algorithm 5. Inputs to the algorithm are the desired false positive rate, f, the detection rate, d, of the cascade stages and the final positive rate, f_{final}, of the cascade. Each stage of the cascade is trained by AdaBoost with the number of features used being increased until the target detection and false positive rates can be met for this level. Because AdaBoost attempts only to minimise errors and is not specifically designed to achieve high detection rates at the expense of larger false positive rates, a threshold, θ, for the strong classifier is introduced. That is, let the new strong classifier be

$$H(x) \begin{array}{lcl} \geq \theta & \Rightarrow & x \text{ is a face} \\ < \theta & \Rightarrow & x \text{ is a non-face} \end{array}$$

Algorithm 4 The AdaBoost Algorithm for Learning a Classifier Function

```
Given example images $(x_1, y_1), \ldots, (x_m, y_m)$ where $y_i = -1, +1$ for negative and
    positive examples, respectively.

Initialise weights $D_1(i) = 1/m$.

Iterate $t = 1, \ldots, T$:

    1. Normalise the weights,
           $$D_t(i) = \frac{D_t(i)}{\sum_{i=1}^{m} D_t(i)}$$
       so that $D_t$ is a probability distribution.
    2. For each feature, $k$:
       - Compute a histogram, with $N$ bins, using the values of the feature
         extracted from the training images.
       - Train a classifier, $h_k$, using the histogram and
           $$c_j = \frac{1}{2} \ln \left( \frac{\sum_{i=1}^{m} D_t(i)[\![x_i \in \mathrm{Bin}(j) \wedge y = +1]\!] + \varepsilon}{\sum_{i=1}^{m} D_t(i)[\![x_i \in \mathrm{Bin}(j) \wedge y = -1]\!] + \varepsilon} \right)$$
         where $c_j = h(x)$ for $x \in \mathrm{Bin}(j)$ and $\varepsilon$ is a small positive number.
       - Compute
           $$Z_k = \sum_{j=1}^{N} \sum_{i:x_i \in \mathrm{Bin}(j)} D(i) \exp(-y_i c_j).$$
    3. Choose the classifier, $h_t$, with the smallest $Z_t$.
    4. Update the weights:
           $$D_{t+1}(i) = D_t(i) \exp\left(-y_i h_t(x_i)\right)$$

Output the final classifier:
    $$H(x) = \mathrm{sign}\left( \sum_{t=1}^{T} h_t(x) \right)$$
```

Reducing the threshold yields a classifier with more false positives and a higher detection rate. The rates are determined by testing the current detector on a validation set.

Subsequent classifiers are trained using those examples which pass through all the previous stages. As a result, the second classifier faces a more difficult task than the first. The examples which make it through the first stage are “harder” than typical examples.

Algorithm 5 Training algorithm for building a cascade detector

```
Input: Allowed false positive rates $f$, detection rate $d$ and final false positive rate $f_{\text{final}}$.

Initialise: $F_0 = 1$, $D_0 = 1$

Do until $F_i < f_{\text{final}}$
    1. Train a classifier with AdaBoost until $f_{\text{reached}} < f$ and $d_{\text{reached}} > d$ on the validation set.
    2. $F_{i+1} = F_i \times f_{\text{reached}}$
    3. $D_{i+1} = D_i \times d_{\text{reached}}$
    4. Discard misclassified faces and generate new data from non-face images.
```

3.2.4 Summary of the System

To summarise this section, the construction of the detection system is done as follows:

1. Simple *features* are designed. There are a large number of candidate features.
2. A small subset of them are selected and the corresponding *weak classifiers* are learned using AdaBoost as described in Algorithm 4.
3. The *strong classifier* is constructed as a linear combination of the weak ones as the output of AdaBoost learning.
4. A *detector* is composed of one or a cascade of strong classifiers.

Chapter 4

Experiments

Two different face detectors have been constructed to compare the performance of the real AdaBoost algorithm with the discrete version used by Viola and Jones. One single strong classifier and one cascade of strong classifiers were trained and tested on a real-world test set.

4.1 Training Data Set

Face images are taken from the MPEG7 face data set [4]. The data set contains face images of variable quality, different facial expressions and taken under a wide range of lightning conditions, with uniform or complex background. The pose of the heads is generally frontal with slight rotation in all directions. The eyes and the nose tip are aligned in all images. The data set contains 3 175 images.

Pose variability was added synthetically to the data. The images were randomly rotated by up to 5°, shifted up to one pixel and the bounding box was scaled by a factor of $1 \pm 0,05$. Two data sets, training and validation, of the same size as the original data set were created by the perturbations.

Non-face images were collected from the web. Images of diverse scenes were included. The data set contains images of animals, plants, countryside, man-made objects, etc.. More than 3 000 images were collected and from them non-face sub-windows were randomly generated.

Examples of faces and non-faces used for training are shown in Figure 4.1.

4.2 The Training Process

During the training process, the non-face part of the training and validation data is updated for each stage. It consists of two sets of 5 000 regions from the non-face images. For the first stage random regions are used, for the later stages, the non-face images are scanned with the cascade built so far and misclassified regions are chosen. The face set remains almost the same over the whole training, only the few faces misclassified at some stage are removed.

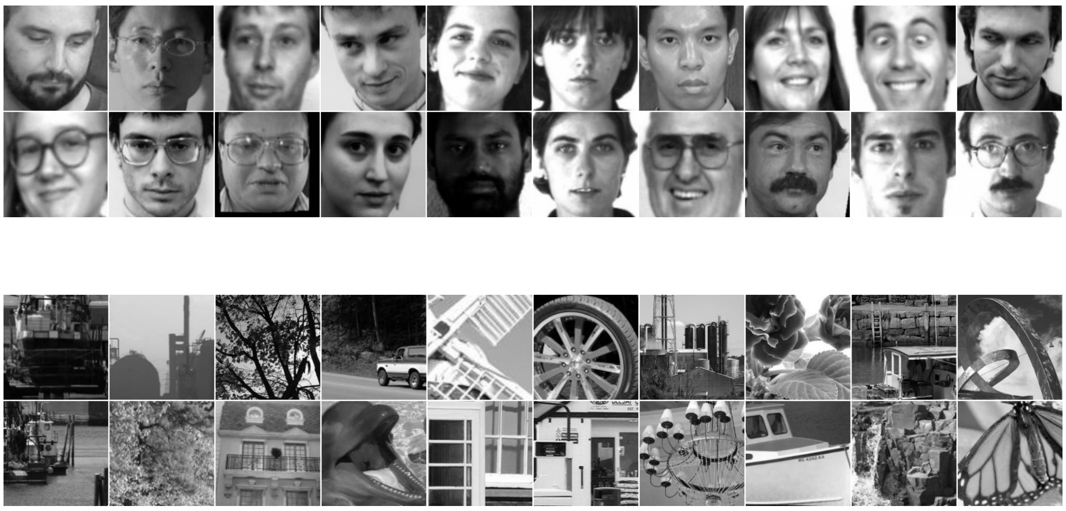


Figure 4.1: *Examples of faces and non-faces used for training.*

The process is driven by the stage false positive, detection and final false positive rates. In the reported experiment the values for each stage were set to $f = 0.4$ for the false positive rate and $d = 0.999$ detection rate, with an overall false positive rate of $f_{\text{final}} = 0.0001$ for the whole cascade detector. Because the overall false positive rate are a product of the rates of the individual stages this means that the number of stages, K, needed, is given by:

$$f^K < f_{\text{final}}.$$

With the values given above this means the the final cascade will need 11 stages ($0.4^{11} = 4.2 \times 10^{-5} < 0.0001$).

4.3 Scanning the Final Detector

Image Pre-processing To minimise the effect of different lightning conditions all example sub-windows used for training were variance normalised. The variance of an image can be computed quickly using a pair of integral images and the the fact that:

$$\sigma^2 = \mu^2 - \frac{1}{N}\sum x^2,$$

where σ is the standard deviation, μ is the mean, and x is the pixel value within the sub-window. The mean of a sub-window can be computed using the integral image. The sum of squared pixels is computed using an integral image of the image squared. During scanning the effect of image normalisation is achieved by scaling the feature values rather than operating on the pixels.

Scanning the Image The final detector, with a base resolution of 24×24 pixels, is scanned across the image at multiple scales and locations. Because the features can be evaluated at any scale with the same cost, scaling is achieved by scaling the detector itself rather than scaling the image. In the results presented here a scale factor of 1.25 is used. The detector is scanned across location by shifting the window a number of pixels Δ. This shifting process is affected by the scale of the detector; if the current scale is s the window is shifted by $[s\Delta]$ pixels, where $[\cdot]$ is the rounding operator.

The choice of Δ affects the speed as well as the accuracy of the detector. For the experiments presented here a step size of 2 pixels has been used.

Post-processing Since the final detector is insensitive to small changes in translation and scale, multiple detections will usually occur around each face and some types of false positives in a scanned image. It is therefore useful to post-process the detected sub-windows in order to combine overlapping detections into a single detection.

As a measure of how "good" a detection is, the magnitude of the strong classifier, $|H(x)|$, is used as a confidence. If two detections overlap and the relative width or height of the intersection is more than 10% the detection with highest confidence is used.

Chapter 5

Results

In this section results for two different trained face detectors are presented. The first system consists of one large single strong classifier and the second system uses the cascade building technique discussed earlier. For the two systems, three detectors were trained using Real AdaBoost (RAB-N), with $N = 4$, 8 and 16 bins in the histogram, respectively. A reference detector using Viola and Jones's setup with Discrete AdaBoost (DAB) and decision stumps was trained.

The main objective of the experiments is to demonstrate detection speedup in comparison with the classical Viola-Jones approach, rather than improvement of the detection rate per se.

One way for describing the performance of a detector system is by constructing its ROC (Receiver Operating Characteristic) curve. A ROC curve is created by adjusting the threshold, θ, for the final stage classifier from $+\infty$ to $-\infty$. A threshold value of $+\infty$ yield a detector with zero detection rate and false positive rate. By reducing the threshold, the detection rate and the false positive rate increase. In effect, a threshold of $-\infty$ is equivalent to removing that layer.

5.1 Test Data Set

The system was tested on the MIT+CMU data set. This data set has been widely used for comparison of face detectors [11, 14, 19] and consists of three large sets of images, which are completely distinct from the training sets. Test Set A was collected by Rowley et al. [11] at CMU, and consists of 42 scanned photographs, newspaper pictures, images collected from the internet, and digitised television pictures. These images contain 169 frontal views of faces. Test Set B consists of 23 images containing 155 faces, it was created and used by Sung and Poggio [10] at the AI/CBCL Lab at MIT, to measure the accuracy of their system. Test Set C is similar to Test Set A, but contains some images with more complex backgrounds and without any faces, to more accurately measure the false detection rate. It contains 65 images with 183 faces.

The data set contains several disputable faces. Thus, the results reported in the papers differ not only by the detection rates but by a subset used for the algorithm evaluation. Moreover, no standard evaluation procedure is given and consequently the results are influenced by the method of successful detection recognition. However, the results presented below are based on the entire data set of 130 images with 507 faces.

Detection rates are reported in percent, while the false positives are specified by their absolute numbers in order to make the results comparable with related work on this data set.

5.2 A Single Strong Classifier

Four large single strong classifiers constructed of 200 features were trained. While a cascade of strong classifiers is needed to achieve a very low false positive rate for face detection, a single classifier is easier to analyse for comparison of the effectiveness of the two boosting algorithms.

The training and validation sets are both composed of 3175 faces and 5000 non-faces as in section 4.1. The error rates for the training and validation sets of the different experiments are shown in Figure 5.1.

The following observations can be made from these curves:

1. Given the same number of learned features or weak classifiers, RAB always achieves lower error rates than DAB for both training and validation.

2. At least for the first 30 weak classifiers, the error rates for the *validation* set is lower for RAB than that of DAB on the *training* set.

3. RAB needs fewer weak classifiers than DAB in order to achieve the same error rate.

4. Using a finer partition of the feature space; that is, using more bins in the histogram, yields better error rates.

Figure 5.2 shows ROC curves for the four systems. As we can see, all RAB systems give better detection rates than DAB. One interesting observation, that we also noticed on the toy problem in section 2.3, is that RAB with 16 bins, which reduces the test error best, does not give best performance on the test set.

5.3 A Cascade of Classifiers

This set of experiments compares classification performance for the four cascades of classifiers. From the ROC curves in Figure 5.3 we see that the performance is similar for the four systems. Comparing the results with the ROC curves for the single strong classifiers in Figure 5.2, we see that the accuracy is not significantly different for RAB, but the speed of the cascaded classifiers are many times faster.

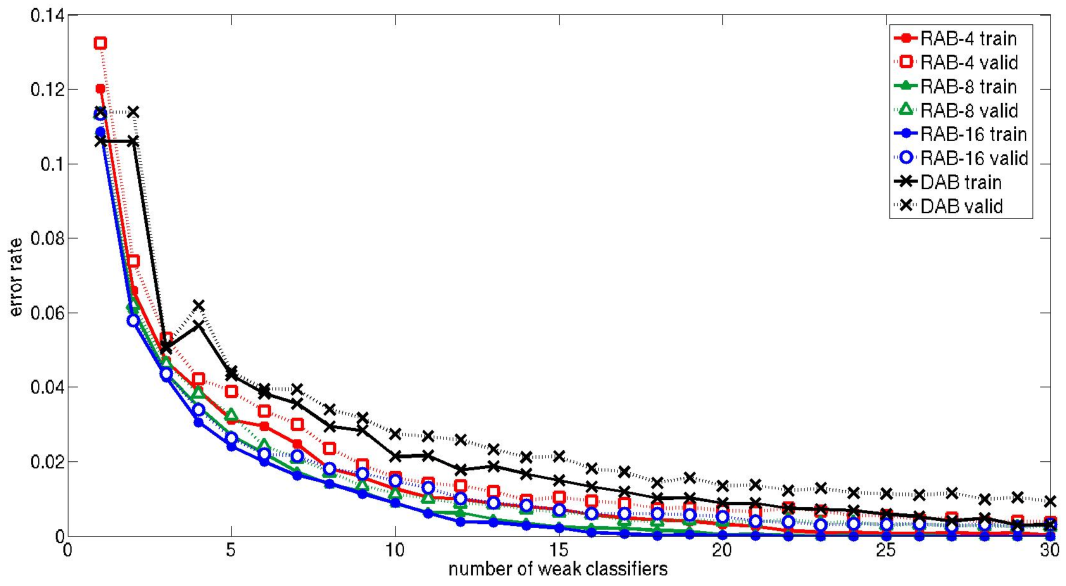


Figure 5.1: *Error rates for the training and validation set for the Real AdaBoost (RAB–N) and Discrete AdaBoost (DAB). Only results for the first 30 out of 200 weak classifiers are shown. Experiments were made with N=4, 8 and 16 bins in the histograms*

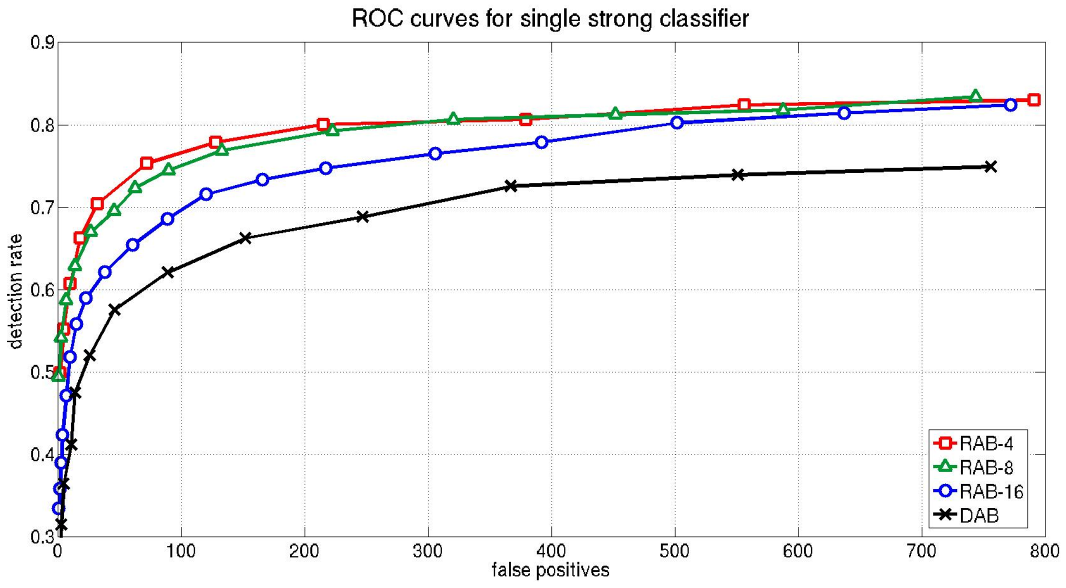


Figure 5.2: *ROC curves for a single strong classifier on the MIT+CMU test set.*

Table 5.1: *Test results on the CMU+MIT test set.*

Stage	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Number of weak classifiers											
DAB	10	11	18	23	21	33	37	40	47	71	62
RAB–4	3	5	7	14	15	12	18	21	24	23	31
RAB–8	3	5	6	8	10	13	14	13	19	17	23
RAB–16	3	5	6	9	10	13	13	14	14	14	18
Total number of weak classifiers											
DAB	10	21	39	62	83	116	153	193	240	311	373
RAB–4	3	8	15	29	44	56	74	95	119	142	173
RAB–8	3	8	14	22	32	45	59	72	91	108	131
RAB–16	3	8	14	23	33	46	59	73	87	101	119
Number of evaluated sub-windows											
DAB	15 971 027	5 072 185	1 956 501	788 091	321 270	132 782	59 495	24 255	11 712	6 281	3 796
RAB–4	15 971 027	4 873 322	2 096 639	938 840	372 891	163 561	67 276	28 963	14 059	7 129	3 938
RAB–8	15 971 027	5 727 715	2 418 409	984 185	410 431	169 671	75 299	35 478	15 577	8 029	4 212
RAB–16	15 971 027	5 702 784	2 228 347	898 097	394 832	177 214	79 173	50 312	21 440	12 277	6 779
False negatives											
DAB	0	1	2	2	6	13	26	41	55	69	85
RAB–4	0	0	0	1	4	10	20	26	38	49	67
RAB–8	0	0	1	2	4	7	19	31	40	57	70
RAB–16	0	0	2	2	2	7	10	17	25	37	51
False positives											
DAB	4 993 201	1 910 903	763 120	306 876	123 289	52 862	19 733	8 259	3 590	1 494	602
RAB–4	4 805 088	2 059 024	920 571	360 941	155 282	61 971	25 277	11 115	4 746	1 907	734
RAB–8	5 654 265	2 373 000	961 064	396 392	161 322	69 501	31 164	12 206	5 288	2 118	909
RAB–16	5 629 749	2 189 348	875 850	381 133	167 942	72 645	44 753	17 375	8 929	4 051	1 599

Table 5.1 compares each stage of the four cascade classifiers in terms of the number of weak classifiers, the total number of weak classifiers, the number of evaluated sub-windows and the number of false negatives and false positives on the MIT+CMU test set.

It can be observed that the complexity of the stages increases gradually, except for a few small fluctuations. RAB needs fewer weak classifiers at each stage than DAB and the number of bins in the histogram seems to further reduce the number of weak classifiers needed, even though the results for 8 and 16 bins are almost the same for the first 8 stages. In total RAB needs $55 - 70\%$ fewer weak classifiers than DAB, where the higher percentage is for 16 bins in the histogram.

We see that the fewer weak classifiers used at each stage the more regions are classified as a potential face, which implies higher detection and false positive rates. To compare the speed of the cascades, the number of weak classifiers evaluated on the MIT+CMU data set was measured. All regions have to be evaluated by the first stage classifier. The number of evaluations in the first stage is consequently a product of the number of scanned sub-windows and the length of the first stage classifier. The same holds for higher stages, but only regions not rejected by the previous stages are evaluated. The results for all stages of the cascades are visualised in Figure 5.4. We see that the number of evaluated weak classifiers is about 60% fewer for RAB than for DAB, no matter how many bins are used.

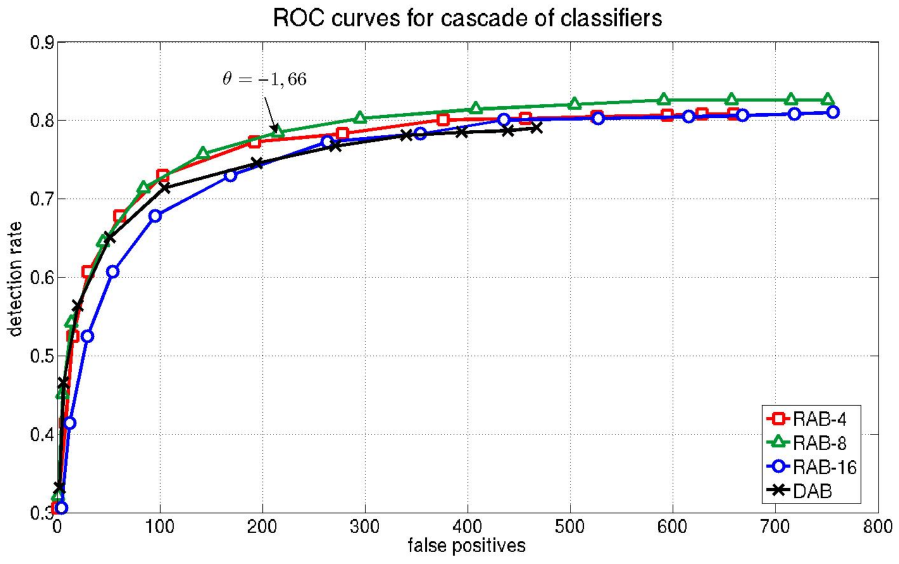


Figure 5.3: *ROC curves for a cascade of classifiers on the MIT+CMU test set.*

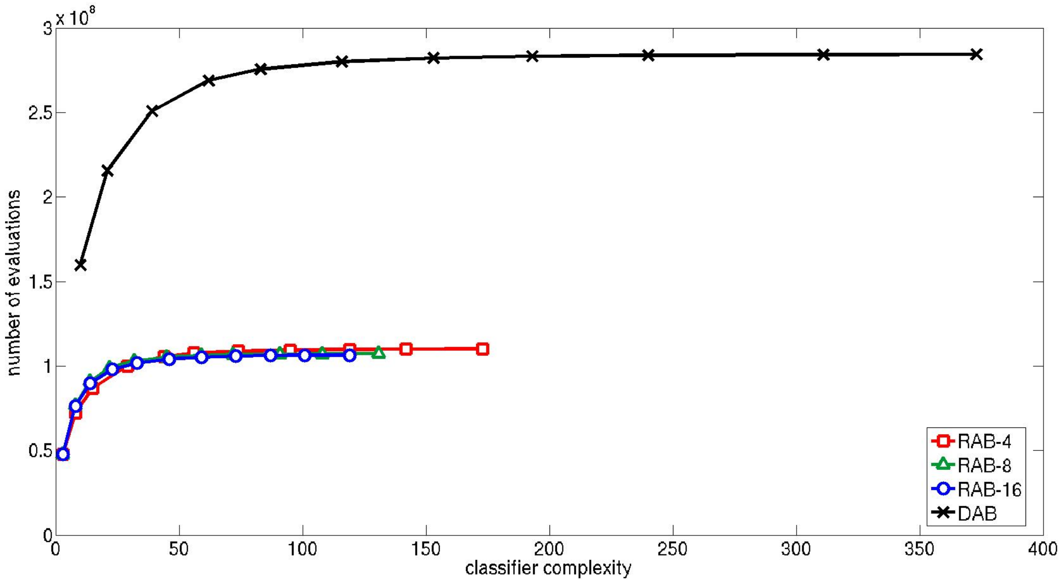


Figure 5.4: *Number of evaluated weak classifiers on the MIT+CMU data set for four cascades of classifiers.*

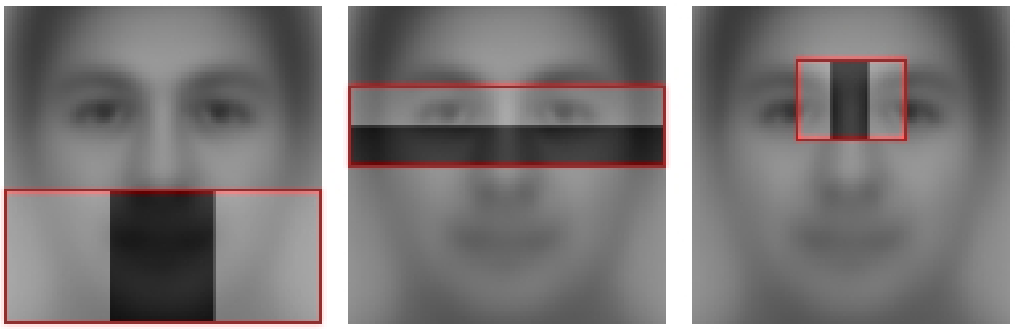


Figure 5.5: *Examples of three features selected early by AdaBoost. The first feature measures the difference in intensity between the region of the mouth and the jaw. The second feature focuses on the observation that the eye region is often darker than the cheeks. The third feature compares the intensities in the eye regions to the intensity of the bridge of the nose.*

In Appendix A a selection of the detected faces on the MIT+CMU test set are presented. The results are for the face detector using Real AdaBoost and histograms with 8 bins, where the final stage classifier was thresholded using $\theta = -1.66$ (see Figure 5.3), which yields a detection rate of $78,5\%$ and 215 false positives. The three features chosen in the first stage for this system are depicted in Figure 5.5.

Chapter 6

Discussion

6.1 Summary and Conclusions

In this Master's thesis the Real AdaBoost algorithm has been thoroughly studied. We have shown how it is derived by minimising an upper bound of the training error for each round of boosting. We have also seen how simple histograms representing the training data can be used to assign confidence to the predictions of the weak classifiers.

The boosting algorithm has been evaluated on the fast frontal face detection problem. The detection system uses Viola and Jones's approach and its real-time performance is attributed especially to the efficient features used, the AdaBoost learning algorithm and the cascade of classifiers for decision making.

Experiments have been made for histograms with 4, 8 and 16 bins. We have seen that Real AdaBoost speeds up the classification by minimising the training error more aggressively than its discrete version. This leads to shorter classifiers and hence to faster classification. In fact, Real AdaBoost needs only $30 - 45\%$ of the number of weak classifiers needed by Discrete AdaBoost and the results shows that the more bins used in the histograms, the fewer weak classifiers are needed.

The shorter classifiers lead to better detection rates but also to more false detections at each stage of the cascade. Hence, subsequent stages are given a harder task. However, due to the short classifiers the total number of weak classifiers evaluated is about 60% fewer, no matter how many bins used.

The reduction of the number of weak classifiers can be important in areas where the weak classifiers are expensive to compute or to implement, e.g on smart cards or other special purpose hardware.

6.2 Further Improvements

When Viola and Jones presented their real-time face detector system, it aroused great interest in the Computer vision community and extensive research has been performed and further improvements suggested.

Two improvements which would be interesting to implement and explore are:

An extended set of features Although frontal faces exhibit little diagonal structure, improvements have been observed by extending the set of Haar-like features to also include the set of 45 degree rotated features [7].

Multi-view face detection Stan Li et. al [5, 6] designed a pyramid of detectors for multi-view face detection. A coarse-to-fine view-partition is used where the out-of-plane head rotations are partitioned into increasingly smaller subspaces and a face detector is trained for each view.

Bibliography

[1] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1990.

[2] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995.

[3] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Technical Report, Dept. of Statistics, Stanford University*, July 1998.

[4] ISO/IEC JTC 1/SC 29/WG 11 Moving Picture Experts Group.

[5] Stan Z. Li and ZhenQiu Zhang. FloatBoost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112– 1123, 2004.

[6] S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, volume 4, pages 67–81. Springer-Verlag, 2002.

[7] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *MRL Technical Report*, May 2002.

[8] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[9] C. Liu. A bayesian discriminating features method for face detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 25, pages 725–740, June 2003.

[10] T. Poggio and K. Sung. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[11] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[12] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[13] R.E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[14] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 746–751, June 2000.

[15] J. Šochman and J. Matas. AdaBoost with totally corrective updates for fast face detection. In *Proceeding of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 445–450, May 2004.

[16] J. Šochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 150–157. IEEE Computer Society, June 2005.

[17] V.N. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, Dec 2001.

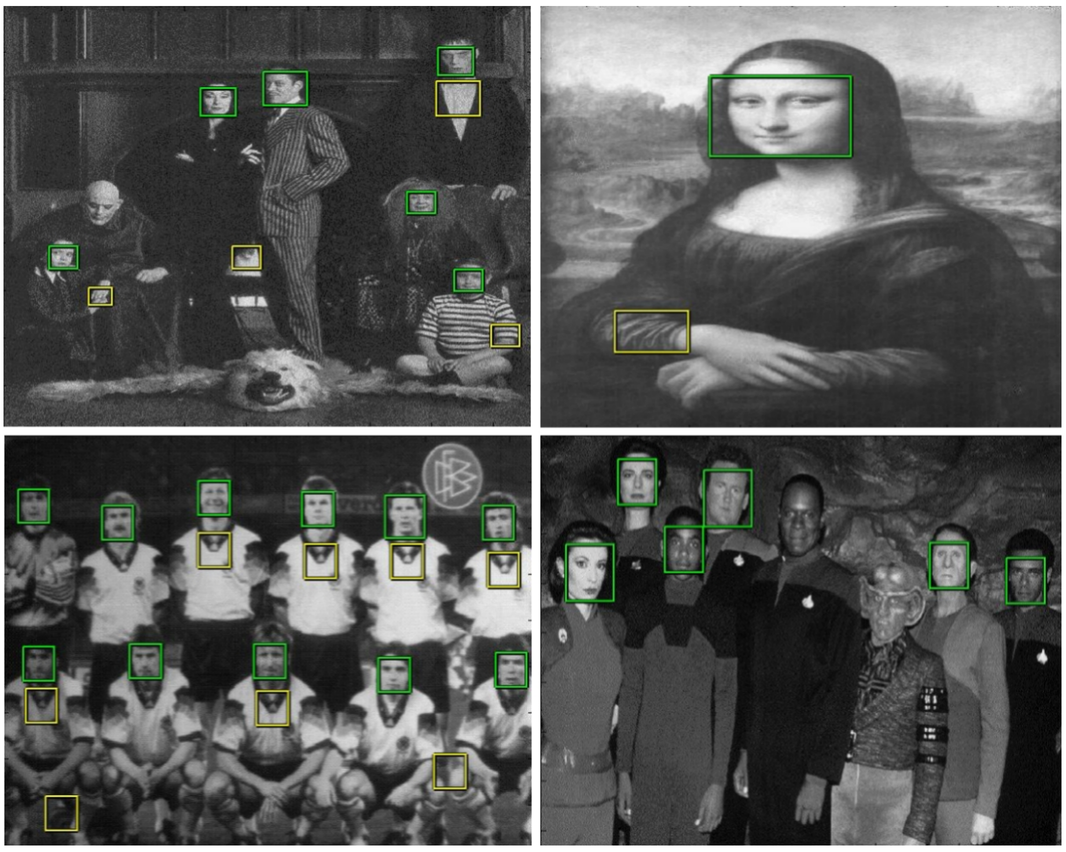
[19] P. Viola and M. Jones. Robust real-time object detection. *IEEE ICCV Workshop Statistical and Computational Theories of Vision*, July 2001.

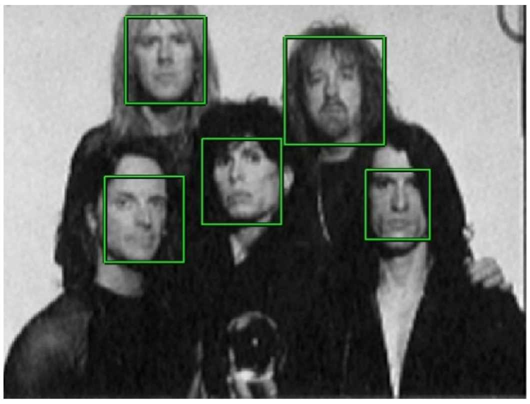
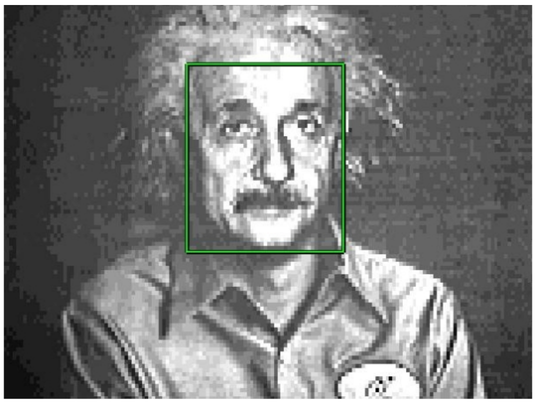
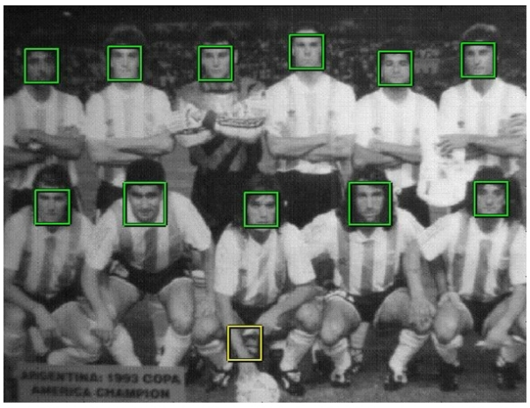
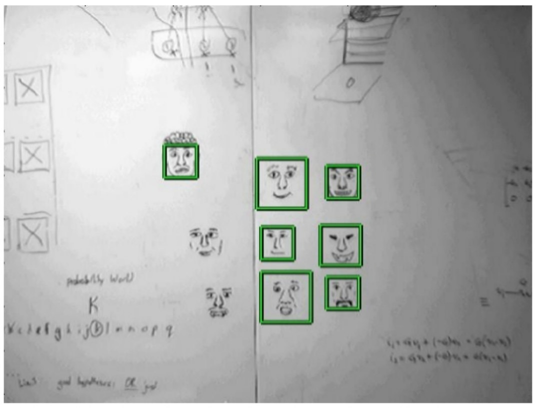
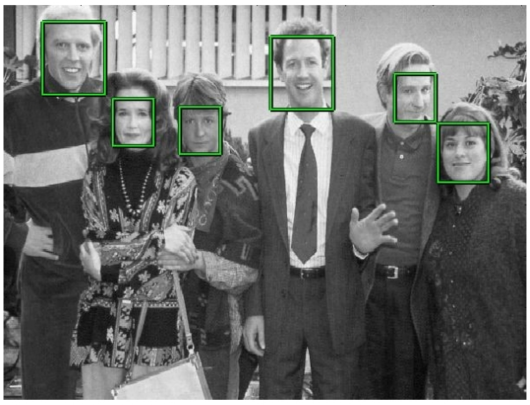
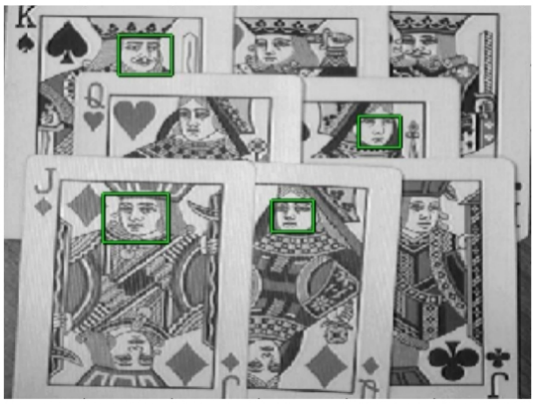
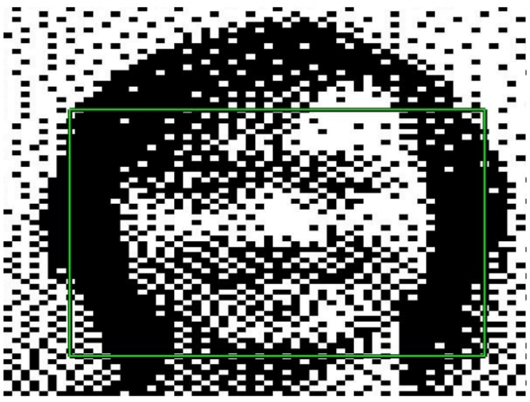
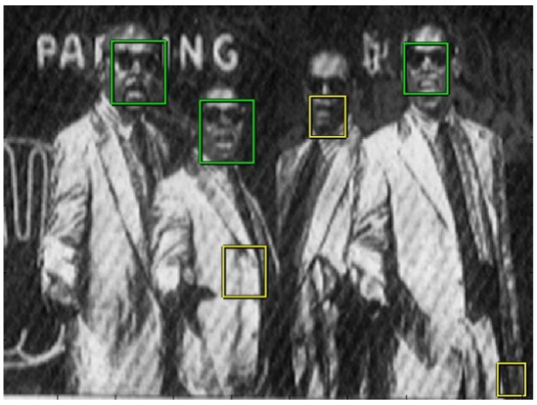
[20] M.H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

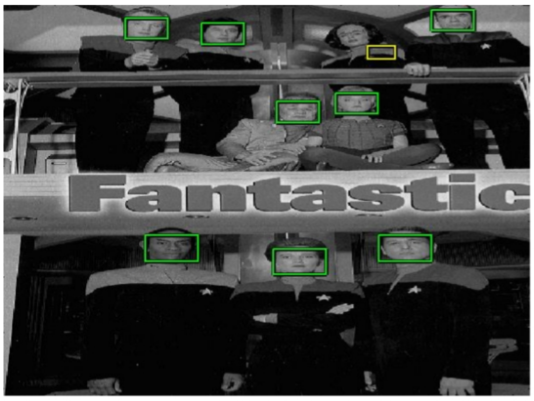
Appendix A

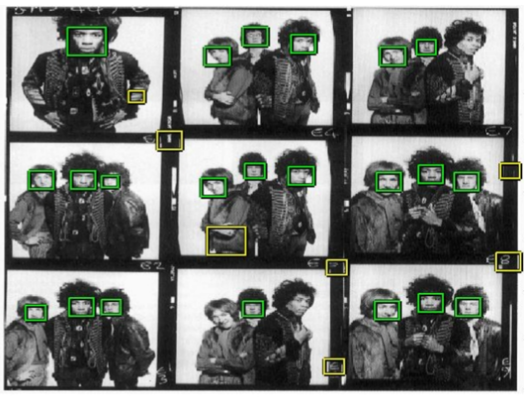
Examples of Detected Faces

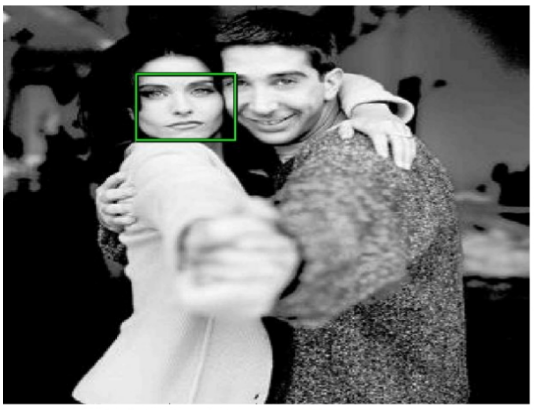
In this appendix a selection of faces detected on the MIT+CMU are presented. The face detector is using Real AdaBoost and histograms with 8 bins. The final stage classifier is thresholded using $\theta = -1.66$ which yields a detection rate of $78,5\%$ and 215 false positives. For ease of presentation the images have been resized.

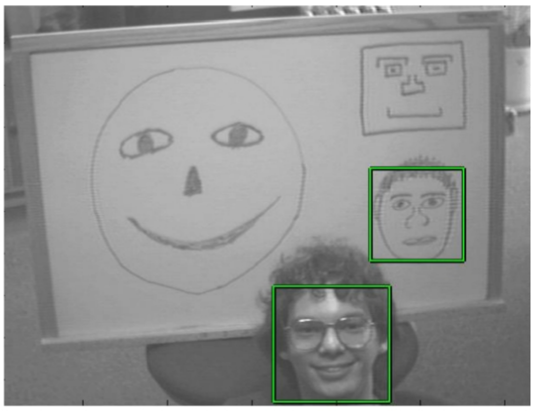


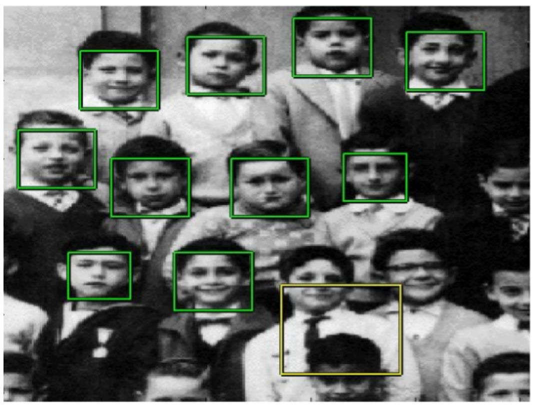



ARGENTINA: 1993 COPA
AMERICA CHAMPION






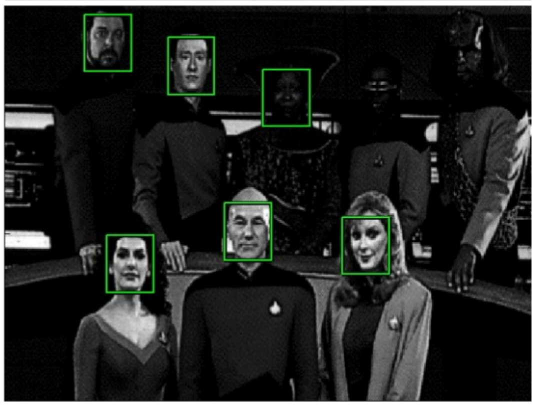

Fantastic

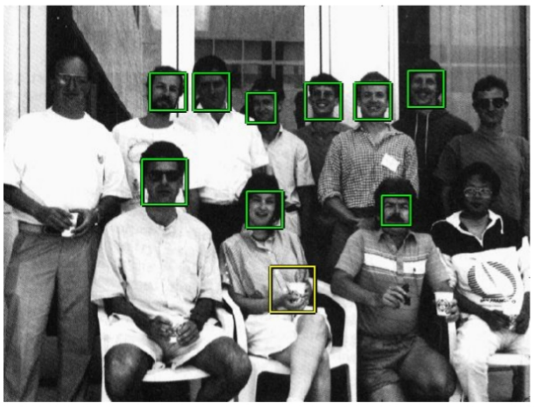


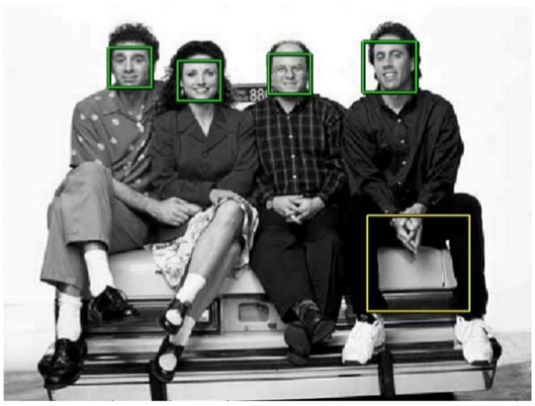












TRITA-CSC-E 2006:110
ISRN-KTH/CSC/E--06/110--SE
ISSN-1653-5715