Boosting Classifier & AdaBoost

Weak classifier

VS

Strong classifier

Outline

- Overview of Boosting classifier
- The AdaBoost Algorithm
- How and why AdaBoost works?
- AdaBoost for Face Detection

PAC学习模型

- PAC学习模型

  机器学习中，训练样本再大也不能代表某类事物本身，所以从训练样本中学习得到“规则”不能对某类事物完全适用，总有失效的情况出现，所以机器学习的目标是概率逼近正确学习！

- **1984年 Valiant提出PAC（**Probably Approximately Correct）学习模型文中提出强学习和弱学习两个概念。

强学习 VS 弱学习

强学习：令 S 为包含 N 个数据点 $(x_1, y_1), \cdots, (x_N, y_N)$ 的样本集，其中 x_n 是按照某种固定但未知的分布 $D(x)$ 随机独立抽取的. $y_n = f(x_n)$，f 属于某个已知的布尔函数集 F. 如果对于任意的 D，任意的 $f \in F$，任意的 $0 \leqslant \epsilon, \delta \leqslant 1/2$，学习算法生成一个满足 $Pr[h(x) \neq f(x)] \leqslant \epsilon$ 的估计 h 的概率大于 $1 - \delta$，并且学习算法的运行时间与 $1/\epsilon, 1/\delta$ 成多项式关系. 则称这种学习算法为强学习算法.

弱学习：其定义与强学习算法类似，只是弱学习算法中只需存在某对 ϵ，δ 满足上述条件即可.

Kearns 和 Valiant 提出了弱学习算法与强学习算法间的等价问题[3]，即是否能把弱学习算法转化为强学习算法？如果两者等价，那么只要找到一个比随机猜测略好的弱学习算法就可以直接将其提升为强学习算法，而不必直接去找很难获得的强学习算法. 在文献[4]中 Kearns 和 Valiant 证明只要有足够的数据，弱学习算法就能通过集成的方式生成任意高精度的估计.

弱学习 - >强学习

- **Valiant的贡献**

  Valiant指出弱学习转换为强学习的可行性！

  实际运用中，人们根据生产经验可以较为容易的找到弱学习方法，但是很多情况下要找到强学习方法是不容易的。有时候人们倾向于通过先找到弱学习然后把它转换为强学习的方式获取强学习方法，而Valiant证明了这种方式的可行性。

怎样实现弱学习转为强学习

- 核心思想：**通过组合使弱学习互补。**

  学习是不适定问题，在有限的样本上，不同的学习方法得到不同的“规则”，并在不同的情况下失效，没有一种学习算法总是在任何领域产生最好的分类效果。

例子

学习算法A在a情况下失效，学习算法B在b情况下失效，那么在a情况下可以用B算法，在b情况下可以用A算法解决。这说明通过某种合适的方式把各种算法组合起来，可以提高准确率。

为实现弱学习互补，面临两个问题：

（1）怎样获得不同的弱分类器？

（2）怎样组合弱分类器？

怎样获得不同的弱分类器？

- 使用不同的弱学习算法得到不同基学习器
  参数估计、非参数估计…
- 使用相同的弱学习算法，但用不同的超参数
  K-Mean不同的K，神经网络不同的隐含层…
- 相同输入对象的不同表示
  不同的表示可以凸显事物不同的特征
- 使用不同的训练集
  装袋（bagging）
  提升（boosting）

怎样组合弱分类器

- 多专家组合

  一种并行结构，所有的弱分类器都给出各自的预测结果，通过“组合器”把这些预测结果转换为最终结果。 eg.投票（voting）及其变种、混合专家模型

- 多级组合

  一种串行结构，其中下一个分类器只在前一个分类器预测不够准（不够自信）的实例上进行训练或检测。 eg. 级联算法（cascading）

弱学习小结

- bagging在给定样本上随机抽取（有放回）训练子集，在每个训练子集上用不稳定的学习算法训练分类不同弱分类器。
- boosting在前一个弱分类器错分的实例在后续的弱分类器上得到更大的重视。
- 从训练子集的获取方式上看：
  bagging靠“运气”，boosting有“依据”！

所谓不稳定学习算法是指训练集很小的变化会引起所产生的分类器变化很大，即学习算法高方差。例如，决策树。

AdaBoost的先验知识

1990年，Schapire[5]最先构造出一种多项式级的算法，即最初的Boosting算法．这种算法可以将弱分类规则转化成强分类规则．一年后，Freund[6]提出了一种效率更高的Boosting算法．1993年，Drucker和Schapire[7]第一次以神经网络作为弱学习器，应用Boosting算法来解决实际的ORC问题．由于早期的Boosting算法在解决实际问题时要求事先知道弱学习算法学习正确率的下限，这实际上很难做到．1995年，Freund和Schapire提出了Adaboost(Adaptive Boosting)算法[8]，这种算法的效率和原来Boosting算法的效率一样，但不需要任何关于弱学习器性能的先验知识，因此可以非常容易地应用到实际问题中．

原理

- AdaBoost的核心思想

  “关注”被错分的样本，“器重”性能好的弱分类器

- 怎么实现

  （1）不同的训练集→调整样本权重

  （2）“关注”→增加错分样本权重

  （3）“器重”→好的分类器权重大

  （4）样本权重间接影响分类器权重

AdaBoost & Its Applications

Overview

Introduction

AdaBoost

Adaptive Boosting

A learning algorithm

Building a strong classifier a lot of weaker ones

AdaBoost Concept

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

weak classifiers

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

strong classifier

slightly better than random

Weaker Classifiers

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

weak classifiers

slightly better than random

- Each weak classifier learns by considering one simple feature
- T most beneficial features for classification should be selected
- How to
  - define features?
  - select beneficial features?
  - train weak classifiers?
  - manage (weight) training samples?
  - associate weight to each weak classifier?

The Strong Classifiers

How good the strong one will be?

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

weak classifiers

strong classifier

slightly better than random

AdaBoost & Its Applications

The AdaBoost Algorithm

Freund 等在 1995 年提出了 Discrete AdaBoost 学习算法[46]，具体算法步骤如下：

1. 对于训练集合 $(x_1, y_1), \cdots, (x_N, y_N)$，$g_j(x_i)$ 代表第 i 个训练图像的第 j 个特征，$y_i \in (1,0)$ 分别表示正例、反例样本。

2. 初始化权重 $D_1(x_i) = \frac{1}{N}$，$i = 1,2,\cdots,N$，其中 N 是正例样本、反例样本的总数。

3. 寻找 T 个弱分类器 $h_t(t = 1,\cdots,T)$

   (1) 对于每个样本中第 j 个特征，可以得到一个弱分类器 h_j，即可以得到阈值 θ_j 和方向 p_j，使得错误率 $\varepsilon_j = \sum_{i=1}^{N} D_t(x_i)\left|h_j(x_i) - y_i\right|$ 达到最小，而弱分类器 h_j 为：

$$h_j(x) = \begin{cases} 1 & p_j g_j(x) < p_j \theta_j \\ 0 & \text{其它} \end{cases} \qquad (2.2.1)$$

   其中，p_j 决定不等式方向，只有 ± 1 两种情况。

   (2) 在所有得到的弱分类器中，找出一个具有最小误差 ε_t 的弱分类器 h_t。

   (3) 对所有样本的权重进行更新：

$$D_{t+1}(x_i) = \frac{D_t(x_i)\beta_t^{1-e_i}}{Z_t} \qquad (2.2.2)$$

   其中，$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$，$Z_t$ 是使 $\sum_{i=1}^{N} D_{t+1}(x_i) = 1$ 的归一化因子，如果 x_i 被 h_t 正确分类，则 $e_i = 0$，反之 $e_i = 1$。

4. 最后得到的强分类器为：

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0 & \text{其它} \end{cases} \qquad (2.2.3)$$

   其中，$\alpha_t = \ln(1/\beta_t)$。

原始AdaBoost

1995年Freund 提出AdaBoost算法，1999年Schapire在一篇会议论文上对Freund的AdaBoost重新表述，基本原理不变但是更易理解，下面以Schapire的版本介绍AdaBoost。

Schapire AdaBoost Algorithm

Given: m examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For t = 1 to T

1. Train learner $\boldsymbol{h_t}$ with min error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
2. Compute the hypothesis weight $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
3. For each example i = 1 to m

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

Output

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

The weight **Ada**pts. The bigger ε_t becomes the smaller α_t becomes.

Boost example if incorrectly predicted.

Z_t is a normalization factor.

Linear combination of models.

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

$D_t(i)$:probability distribution of x_i's at time t

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

minimize weighted error

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

for minimize exponential loss

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Give error classified patterns more chance for learning.

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

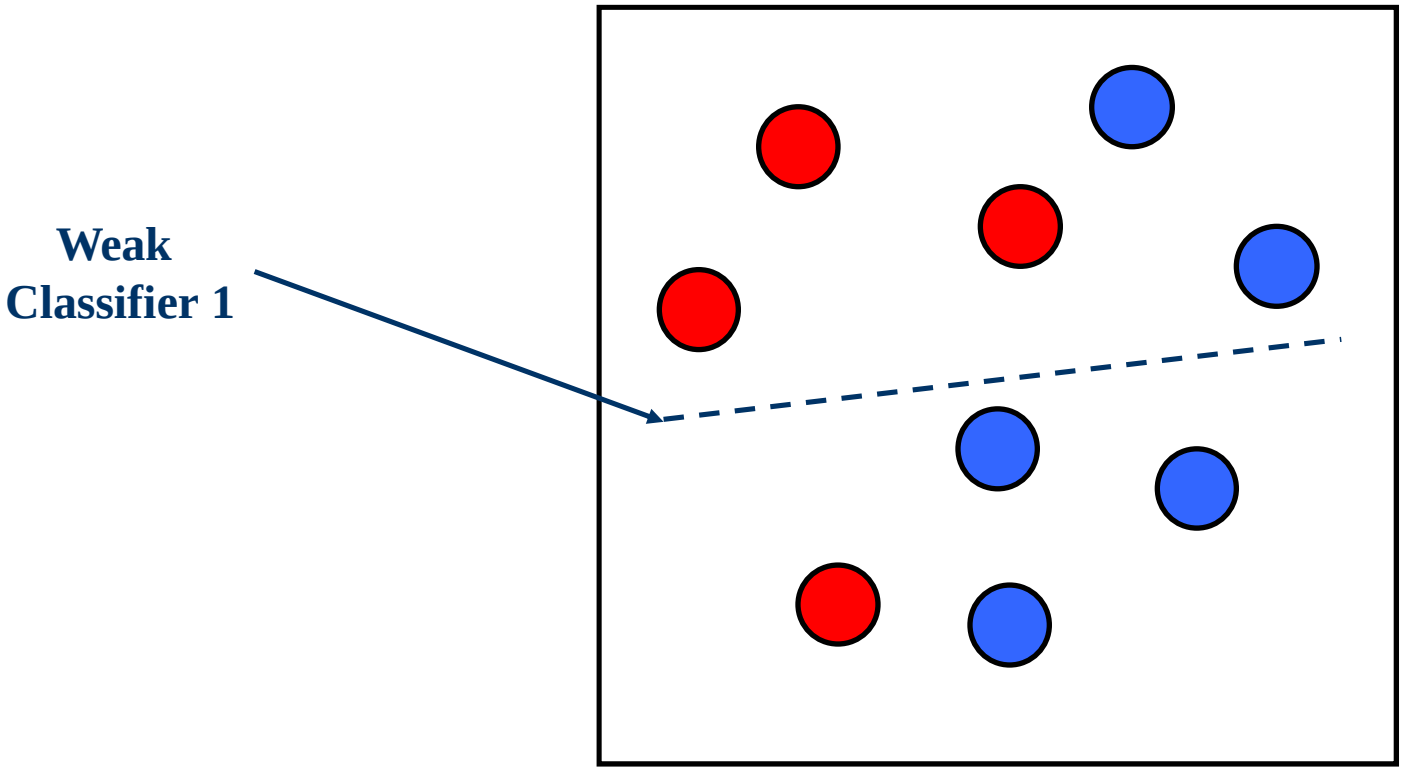
$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

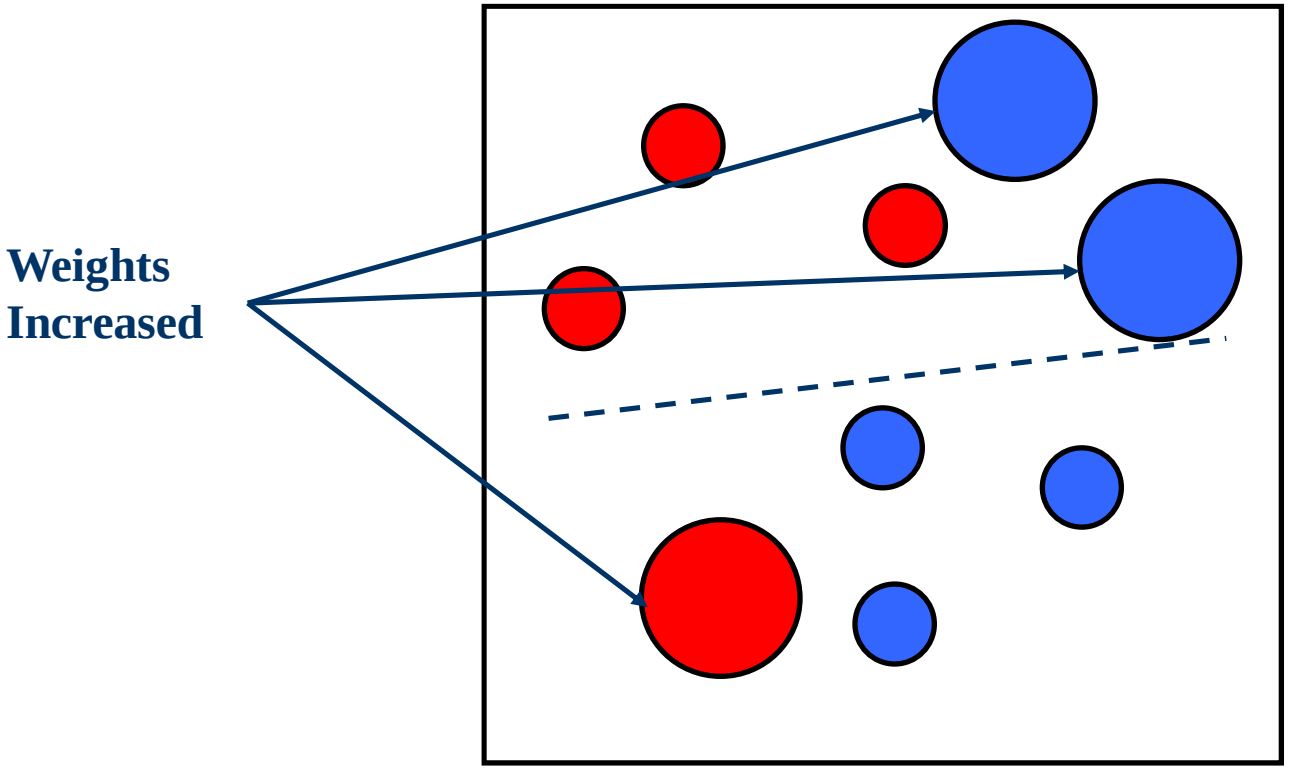
- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

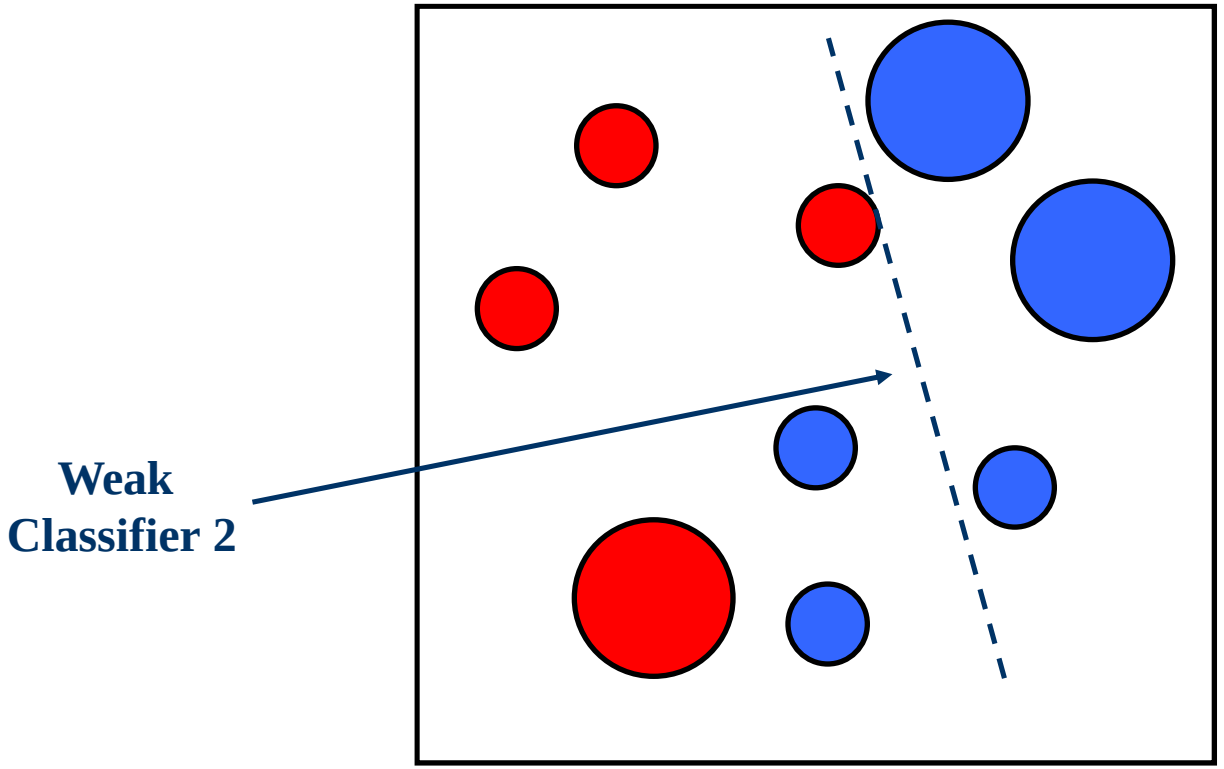
Boosting illustration



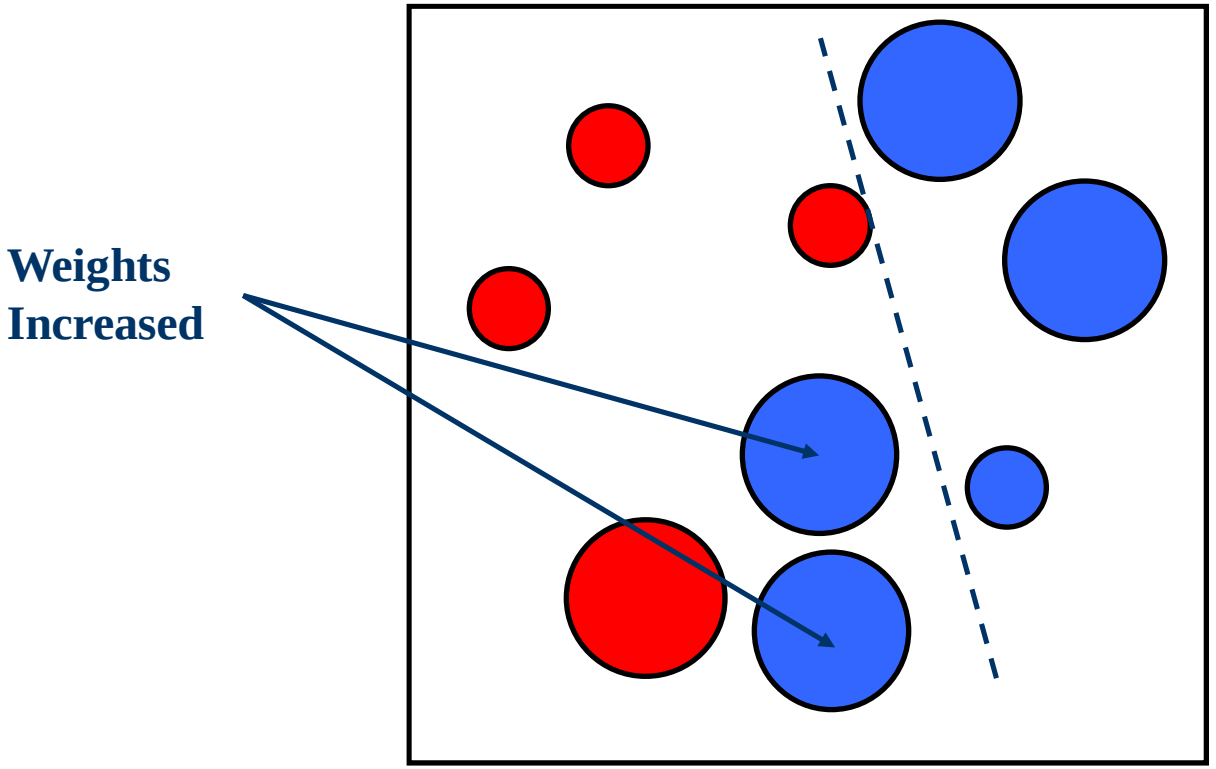
Boosting illustration



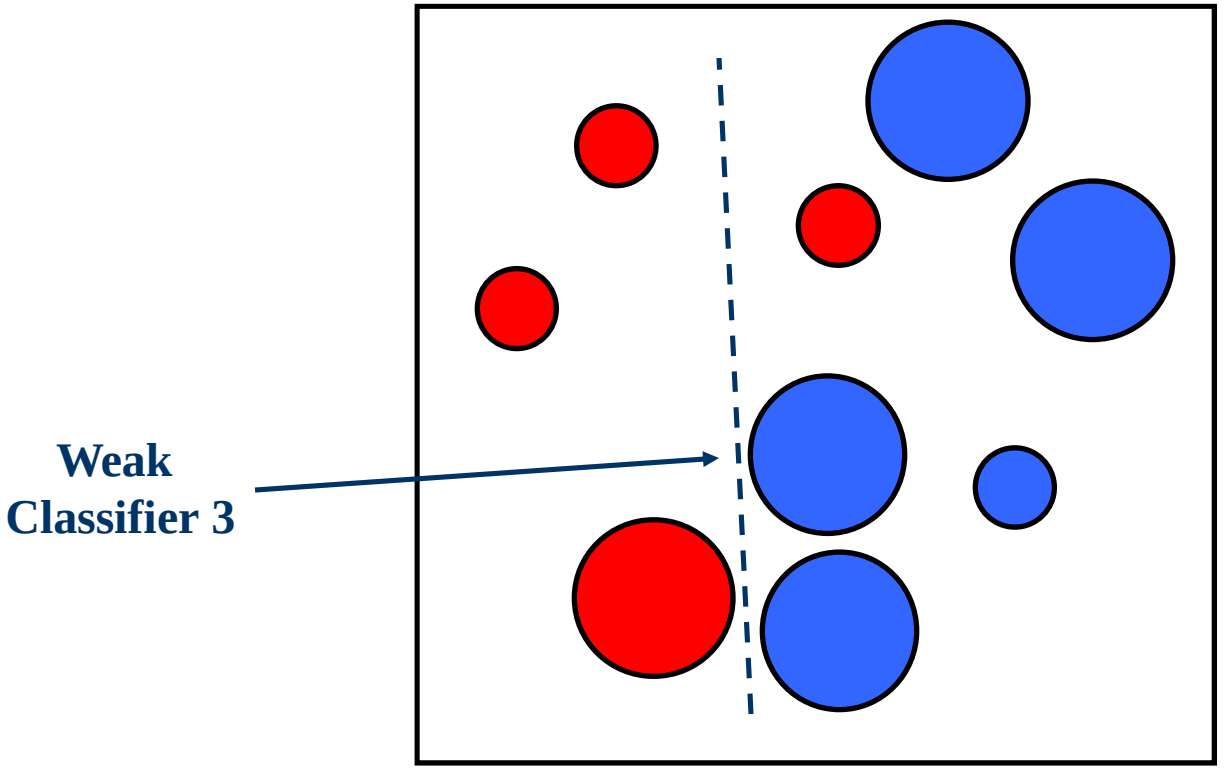
Boosting illustration



Boosting illustration

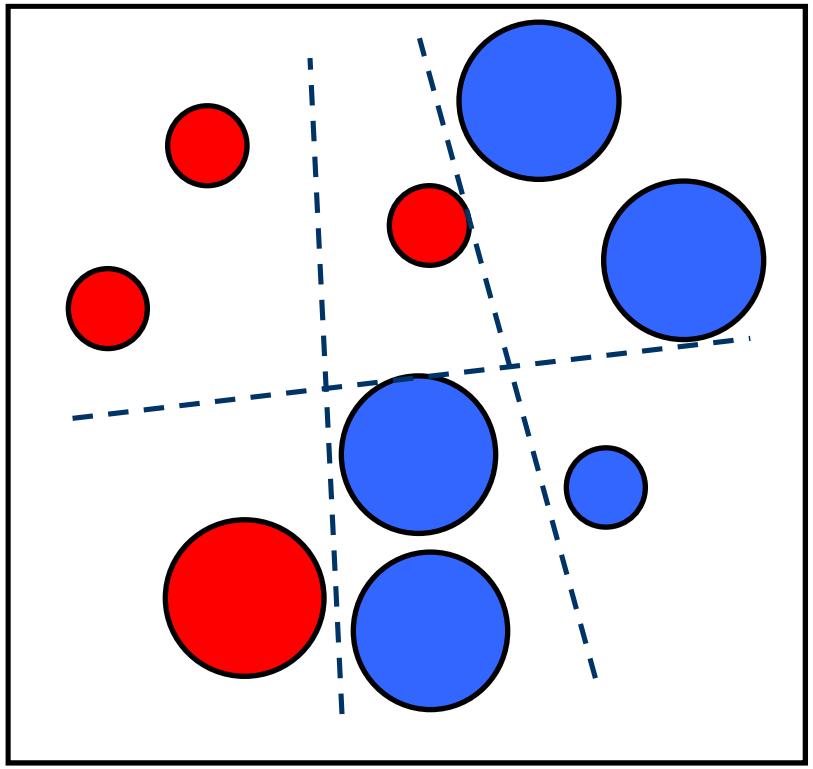


Boosting illustration



Boosting illustration

Final classifier is a combination of weak classifiers



AdaBoost & Its Applications

How and why AdaBoost works?

What goal the AdaBoost wants to reach?

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

What goal the AdaBoost wants to reach?

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ wh

Initialization: $D_1(i) = \frac{1}{}$

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

- Find classifier $h_t : X \to \{-1,$ D_t, i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

They are goal dependent.

Goal

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

Minimize exponential loss

$$loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$$

Goal

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

Minimize exponential loss

$$loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$$

Maximize the margin $yH(x)$

Goal

Minimize $loss_{\exp}[H(x)] = E_{x,y}\left[e^{-yH(x)}\right]$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t(x)}\right] = E_x\left[E_y\left[e^{-yH_t(x)} \mid x\right]\right]$$

$$= E_x\left[E_y\left[e^{-y[H_{t-1}(x)+\alpha_t h_t(x)]} \mid x\right]\right]$$

$$= E_x\left[E_y\left[e^{-yH_{t-1}(x)} e^{-y\alpha_t h_t(x)} \mid x\right]\right]$$

$$= E_x\left[e^{-yH_{t-1}(x)}\left[e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))\right]\right]$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$

$\alpha_t = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t(x)}\right] = E_x\left[e^{-yH_{t-1}(x)}\left[e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))\right]\right]$$

Set $\frac{\partial}{\partial \alpha_t} E_{x,y}\left[e^{-yH_t(x)}\right] = 0$

$$\Rightarrow E_x\left[e^{-yH_{t-1}(x)}\left[\underbrace{-e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))}_{0}\right]\right] = 0$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$

$\alpha_t = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow \alpha_t = \frac{1}{2}\ln\frac{P(y = h_t(x))}{P(y \neq h_t(x))} \quad \Rightarrow \alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

$P(x_i, y_i) = D_t(i)$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x\left[e^{-yH_{t-1}(x)}\left[\underbrace{-e^{-\alpha_t}P(y = h_t(x)) + e^{\alpha_t}P(y \neq h_t(x))}_{0}\right]\right] = 0$$

$\alpha_t = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

Define $H_t(x) =$

Then, $H(x) = H$

$$\Rightarrow \alpha_t = \frac{1}{2} \ln \frac{P(y = h_t(x))}{P(y \neq h_t(x))} \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

$$P(x_i, y_i) = D_t(i)$$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x \left[e^{-yH_{t-1}(x)} \left[\underbrace{-e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))}_{0} \right] \right] = 0$$

$D_{t+1} = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

Define $H_t(x) =$

Then, $H(x) = H$

$$\Rightarrow \alpha_t = \frac{1}{2} \ln \frac{P(y = h_t(x))}{P(y \neq h_t(x))} \quad \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

$$P(x_i, y_i) = D_t(i)$$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x \left[e^{-yH_{t-1}(x)} \underbrace{\left[-e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x)) \right]}_{0} \right] = 0$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t}\right] = E_{x,y}\left[e^{-yH_{t-1}}e^{-y\alpha_t h_t}\right] \approx E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h_t + \frac{1}{2}\alpha_t^2 y^2 h_t^2\right)\right]$$

$$\Rightarrow h_t = \arg\min_h E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2 y^2 h^2\right)\right] \qquad y^2h^2 = 1$$

$$\Rightarrow h_t = \arg\min_h E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right] | x\right]$$

Minimize $loss_{\exp}[H(x)] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow h_t = \arg\max_h E_x\left[1\cdot h(x)e^{-H_{t-1}(x)}\cdot P(y=1\mid x) + (-1)\cdot h(x)e^{H_{t-1}(x)}\cdot P(y=-1\mid x)\right]$$

$$\Rightarrow h_t = \arg\max_h E_x\left[E_y\left[e^{-yH_{t-1}}(yh)\right]\mid x\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}(-y\alpha_t h)\right]\mid x\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right]\mid x\right]$$

Minimize $loss_{\exp}[H(x)] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow h_t = \arg\max_h E_x\left[1\cdot h(x)e^{-H_{t-1}(x)}\cdot P(y=1\mid x) + (-1)\cdot h(x)e^{H_{t-1}(x)}\cdot P(y=-1\mid x)\right]$$

$$\Rightarrow h_t = \arg\max_h E_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}[yh(x)]$$ maximized when $y = h(x) \quad \forall x$

$$\Rightarrow h_t(x) = sign\left(E_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}[y\mid x]\right)$$

$$\Rightarrow h_t(x) = sign\left(P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=1\mid x) - P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=-1\mid x)\right)$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

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

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

At time t $x, y \sim e^{-yH_{t-1}(x)} P(y \mid x)$

$$\Rightarrow h_t(x) = sign\left(P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=1\mid x) - P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=-1\mid x)\right)$$

$D_{t+1} = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

Define $H_t(x) =$

Then, $H(x) = H$

At time t $\quad x, y \sim e^{-yH_{t-1}(x)} P(y \mid x)$

At time 1 $\quad x, y \sim P(y \mid x)$ $\qquad P(y_i \mid x_i) = 1 \Rightarrow D_1(1) = \frac{1}{Z_1} = \frac{1}{m}$

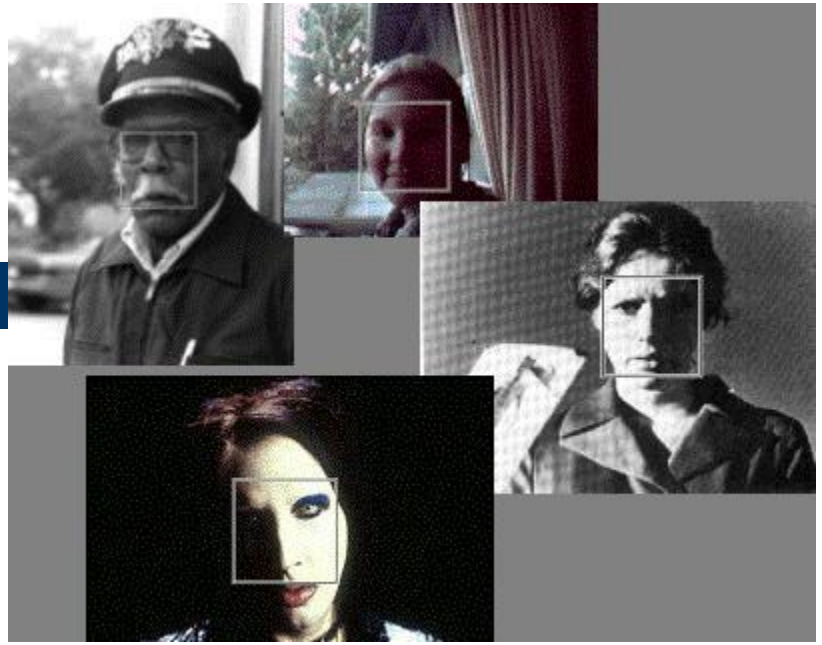
At time t+1 $\quad x, y \sim e^{-yH_t(x)} P(y \mid x) \equiv D_t e^{-\alpha_t y h_t(x)}$

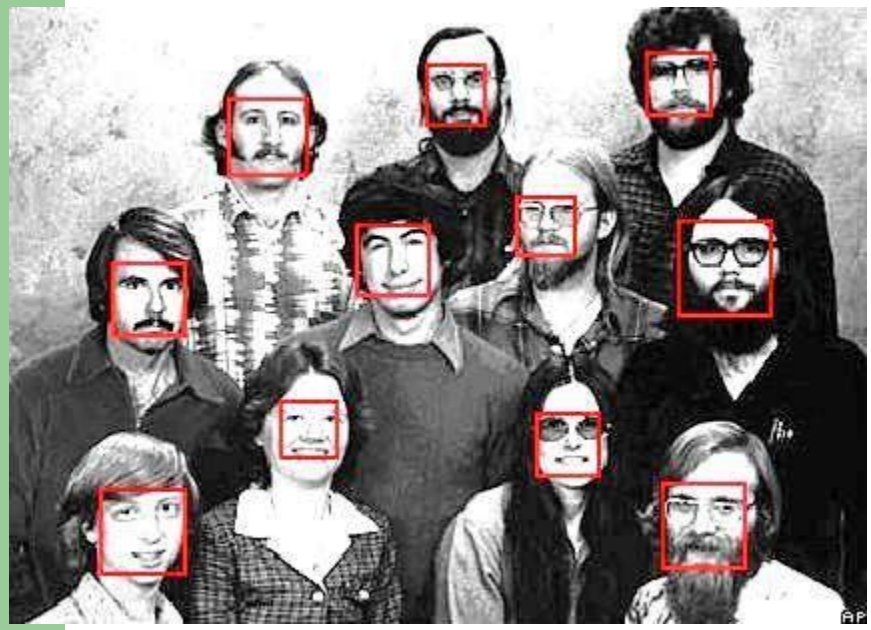
$$\Rightarrow D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}, \; Z_t \text{ is for normalization}$$

AdaBoost & Its Applications

AdaBoost for Face Detection

The Task of Face Detection





Many slides adapted from P. Viola

Basic Idea

- Slide a window across image and evaluate a face model at every location.

Challenges

- Slide a window across image and evaluate a face model at every location.
- Sliding window detector must evaluate tens of thousands of location/scale combinations.
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

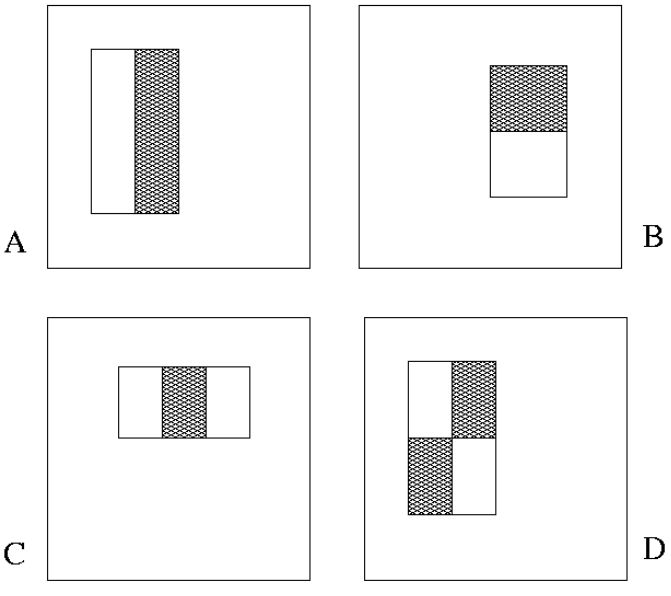
The Viola/Jones Face Detector

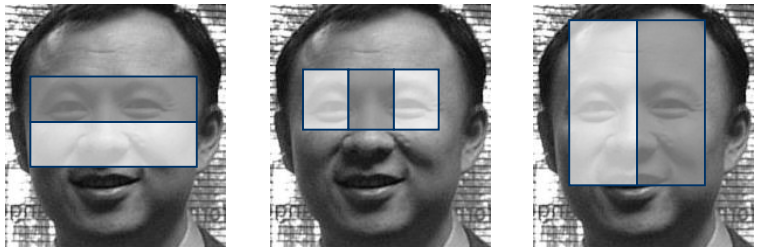
- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Image Features



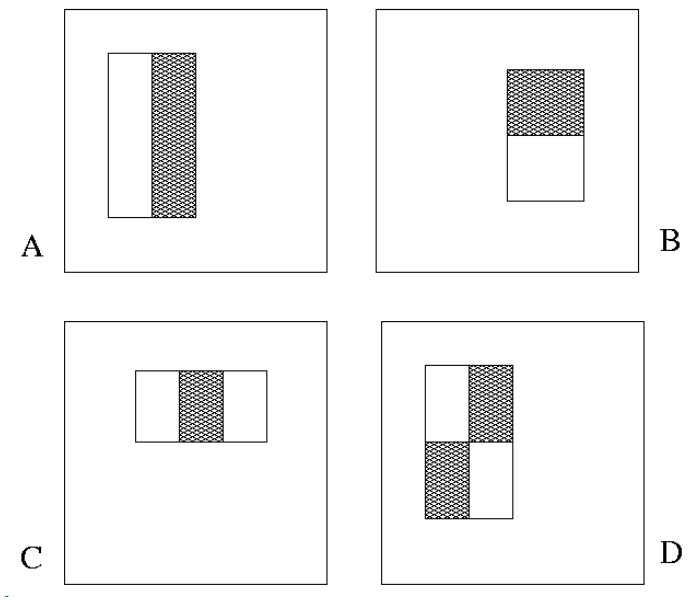


Rectangle filters

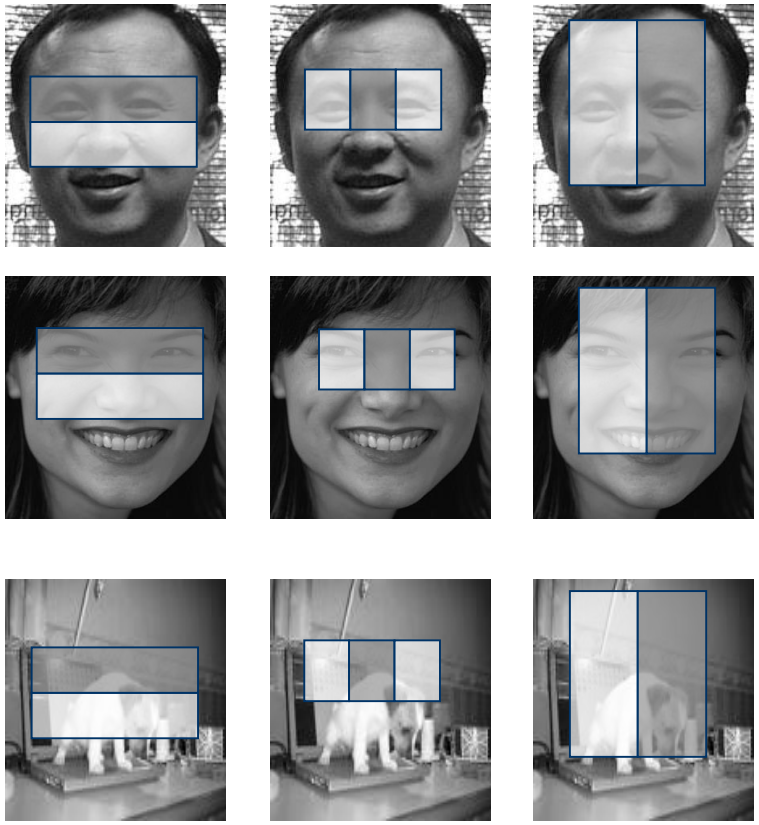
$$\text{Feature Value} = \sum(\text{Pixel in white area}) - \sum(\text{Pixel in black area})$$

$$\text{Feature Value} = \sum(\text{Pixel in white area}) - \sum(\text{Pixel in black area})$$

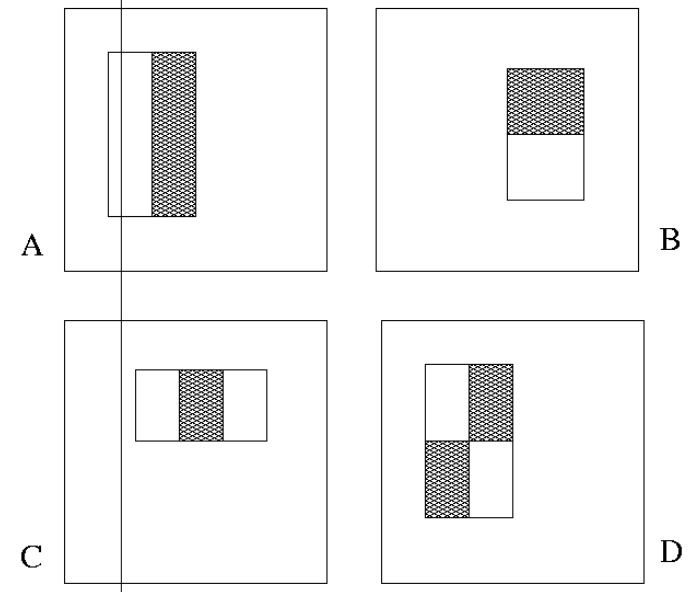
Image Features



Rectangle filters



Size of Feature Space



Rectangle filters

- How many number of possible rectangle features for a 24x24 detection region?

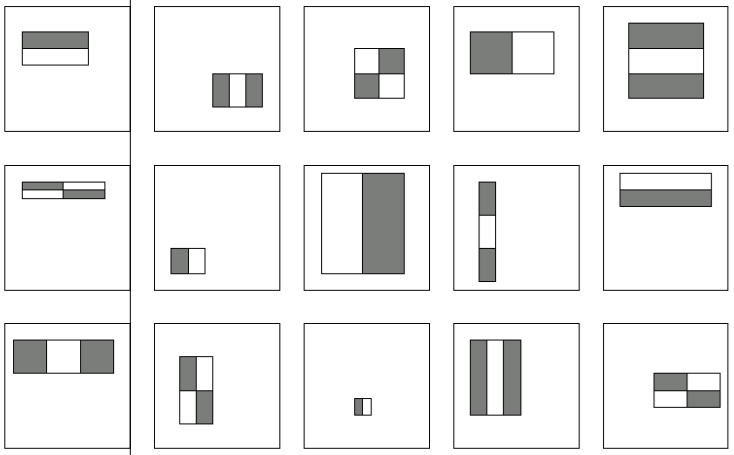
A+B $2\sum_{w=1}^{12}\sum_{h=1}^{24}(24-2w+1)(24-h+1)+$

C $2\sum_{w=1}^{8}\sum_{h=1}^{24}(24-3w+1)(24-h+1)+$

D $\sum_{w=1}^{12}\sum_{h=1}^{12}(24-2w+1)(24-2h+1)$

$\approx 160,000$

Feature Selection



What features are good for face detection?

- How many number of possible rectangle features for a 24x24 detection region?

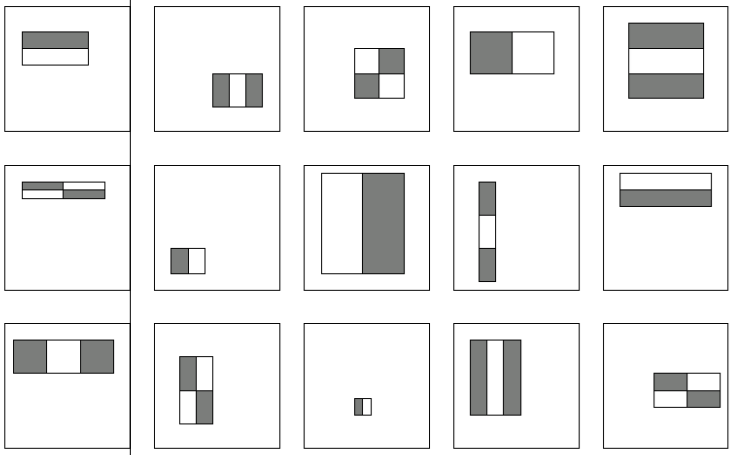
A+B $2\sum_{w=1}^{12}\sum_{h=1}^{24}(24-2w+1)(24-h+1)+$

C $2\sum_{w=1}^{8}\sum_{h=1}^{24}(24-3w+1)(24-h+1)+$

D $\sum_{w=1}^{12}\sum_{h=1}^{12}(24-2w+1)(24-2h+1)$

$\approx 160,000$

Feature Selection



- How many number of possible rectangle features for a 24x24 detection region?

A+B $$2\sum_{w=1}^{12}\sum_{h=1}^{24}(24-2w+1)(24-h+1)+$$

C $$2\sum_{w=1}^{8}\sum_{h=1}^{24}(24-3w+1)(24-h+1)+$$

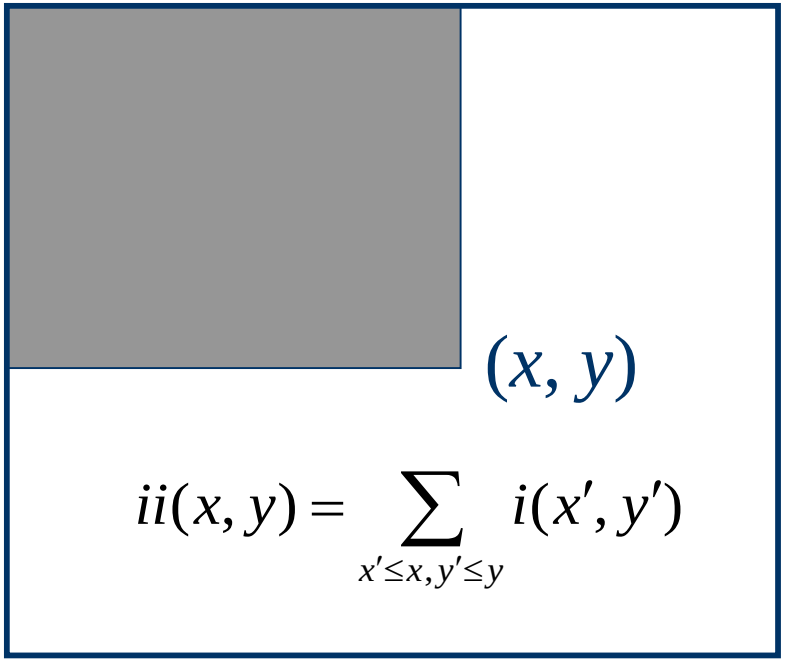
D $$\sum_{w=1}^{12}\sum_{h=1}^{12}(24-2w+1)(24-2h+1)$$

$\approx 160,000$

- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

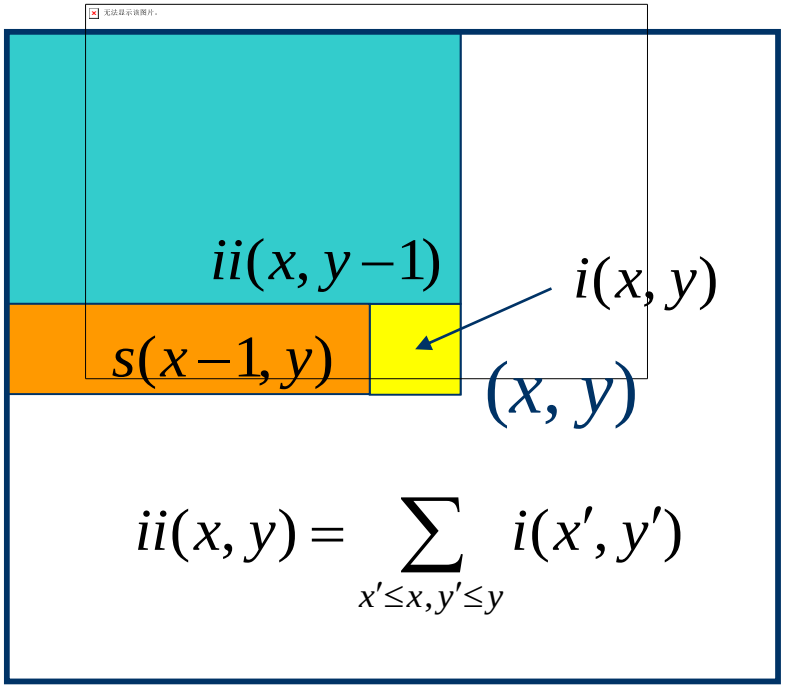
Integral images

- The integral image computes a value at each pixel (x, y) that is the sum of the pixel values above and to the left of (x, y), inclusive.



Computing the Integral Image

- The integral image computes a value at each pixel (x, y) that is the sum of the pixel values above and to the left of (x, y), inclusive.
- This can quickly be computed in one pass through the image.

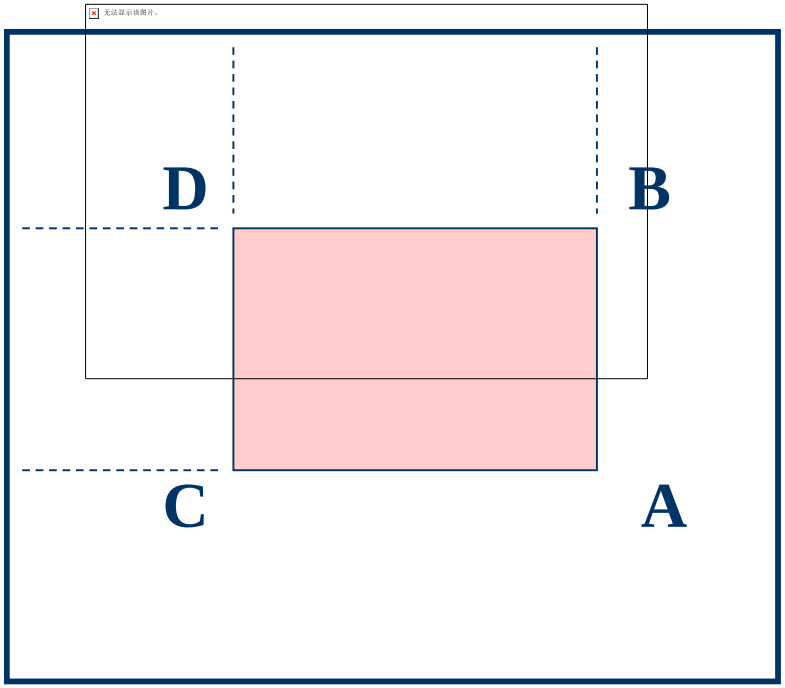


$$s(x, y) = s(x-1, y) + i(x, y)$$

$$ii(x, y) = ii(x, y-1) + s(x, y)$$

Computing Sum within a Rectangle

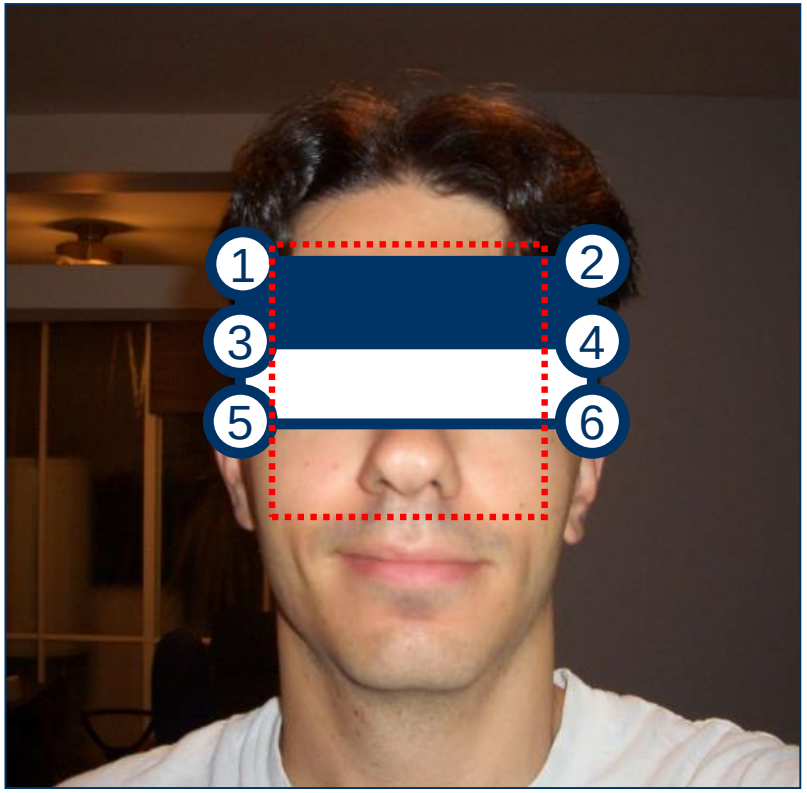
$$sum = ii_A - ii_B - ii_C + ii_D$$



Only 3 additions are required for any size of rectangle!

Scaling

- Integral image enables us to evaluate *all* rectangle sizes in constant time.
- Therefore, *no* image scaling is necessary.
- Scale the rectangular features instead!



Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
  - A weak learner need only do better than chance
- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - "Hardness" is captured by weights attached to training examples

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

$D_t(i)$:probability distribution of x_i's at time t

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

minimize weighted error

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

for minimize exponential loss

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Give error classified patterns more chance for learning.

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

Weak Learners for Face Detection

What base learner is proper for face detection?

Given: $(x_1, y_1), \ldots, (x_m, y_m)$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

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

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

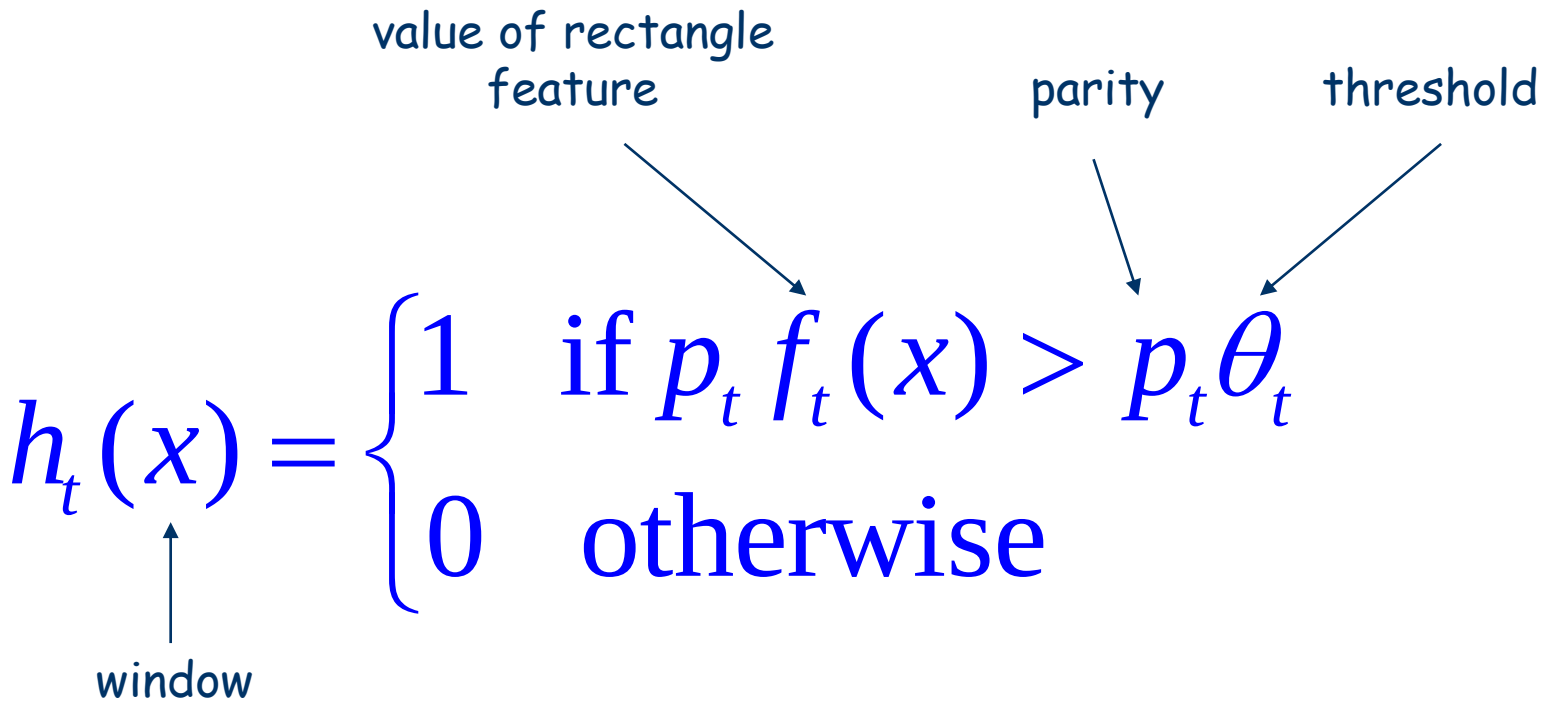
$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

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

Weak Learners for Face Detection



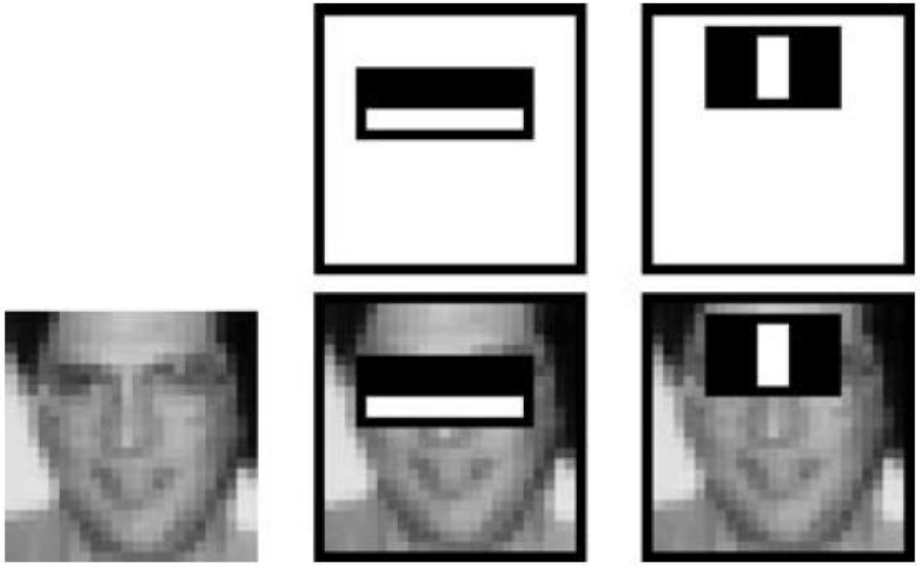
Boosting

- Training set contains face and nonface examples
  - Initially, with equal weight
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples

- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

Features Selected by Boosting

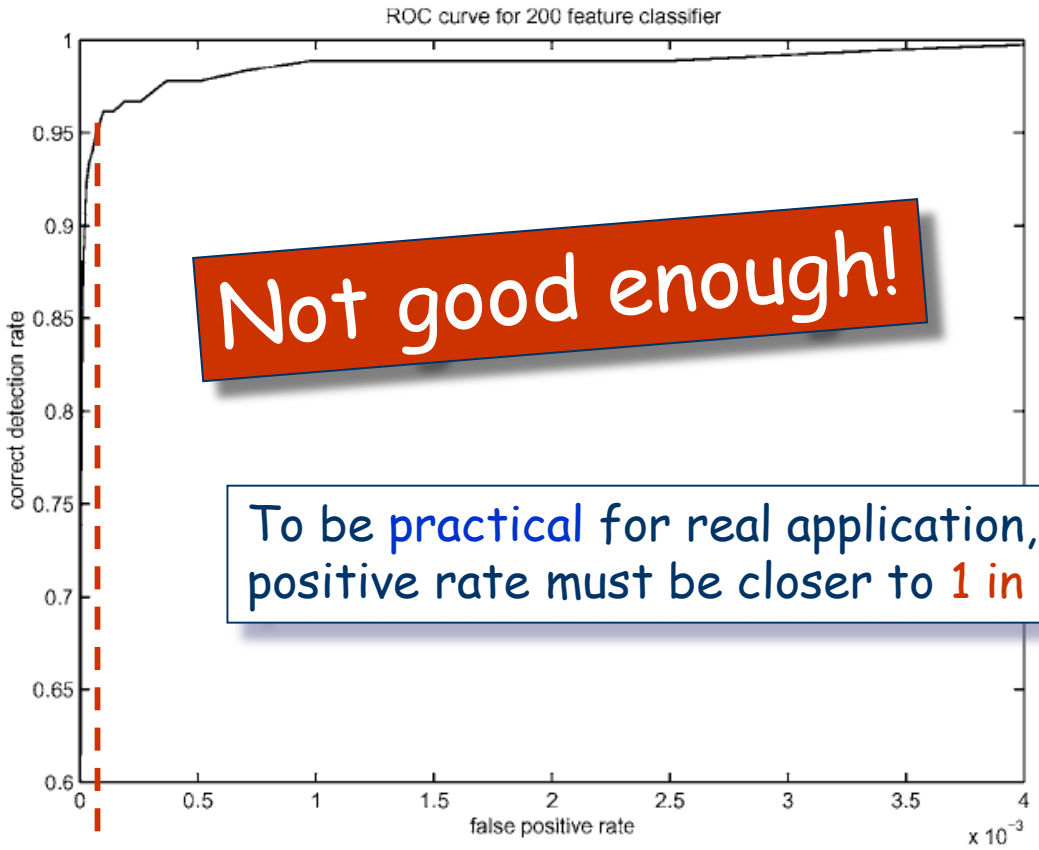
First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084.

ROC Curve for 200-Feature Classifier

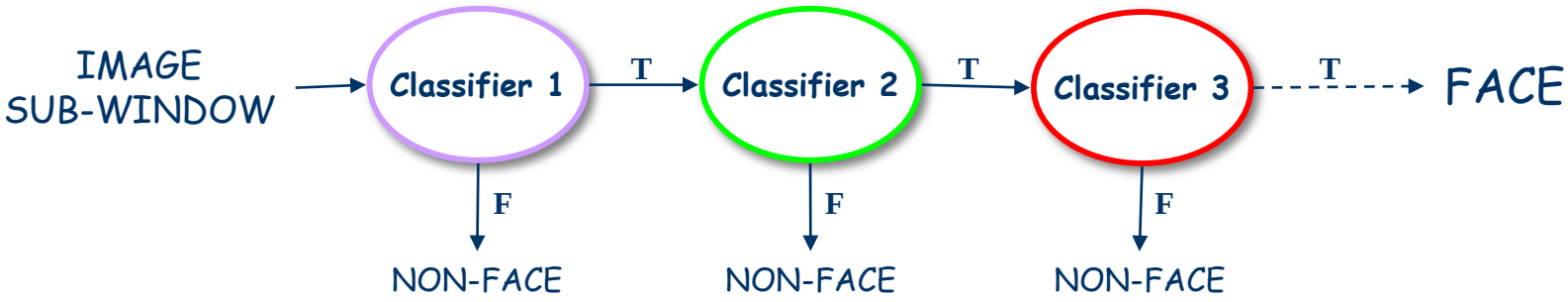


Not good enough!

To be practical for real application, the false positive rate must be closer to 1 in 1,000,000.

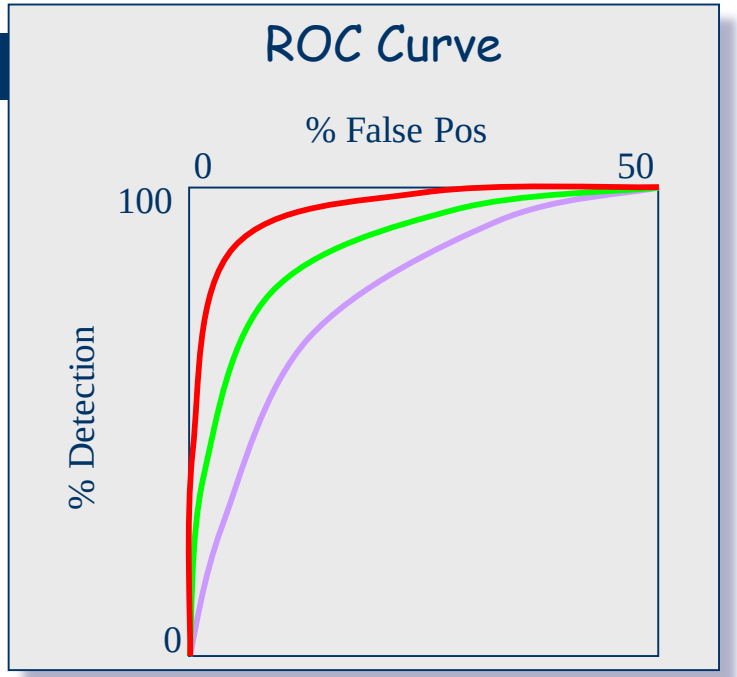
Attentional Cascade

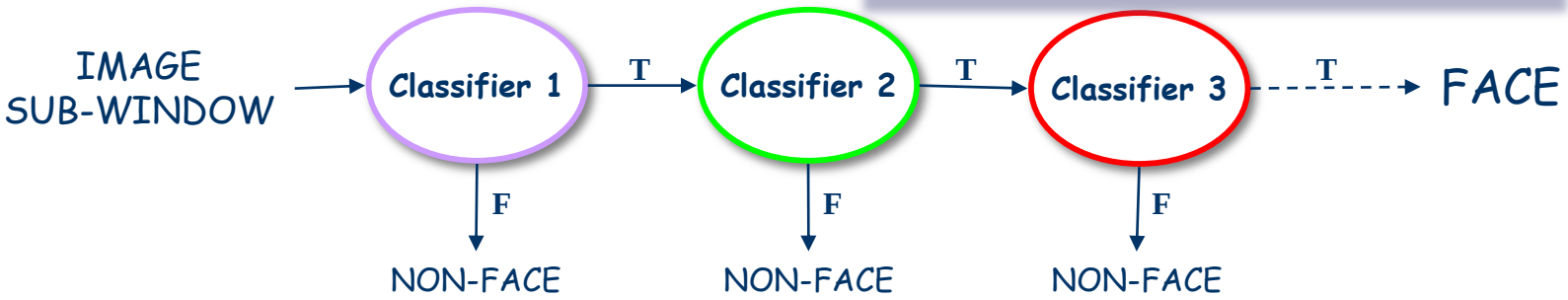
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



Attentional Cascade

- Chain classifiers that are progressively more complex and have lower false positive rates

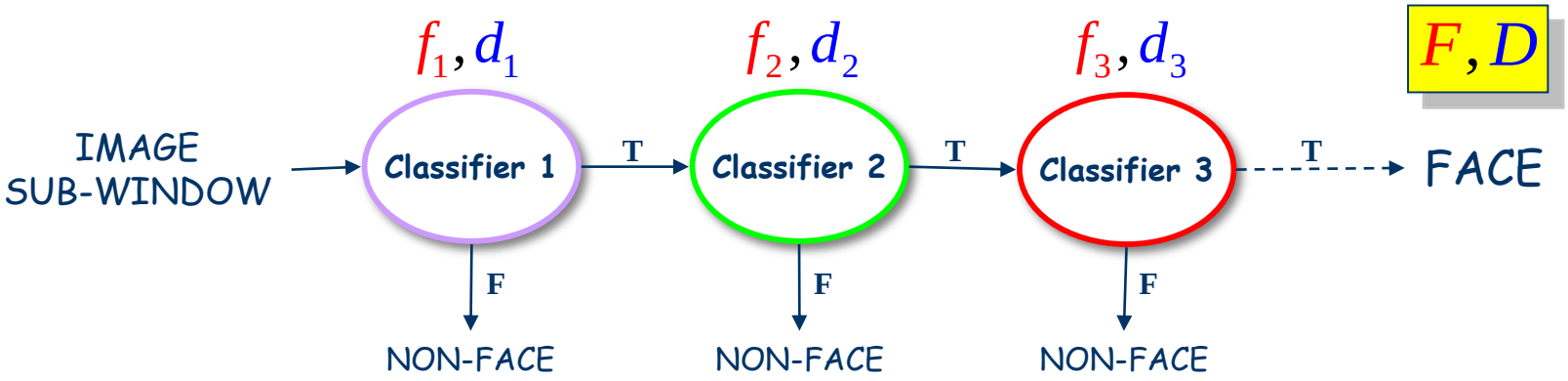




Detection Rate and False Positive Rate for Chained Classifiers

$$F = \prod_{i=1}^{K} f_i, \quad D = \prod_{i=1}^{K} d_i$$

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of **0.9** and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of **0.99** ($0.99^{10} \approx 0.9$) and a false positive rate of about **0.30** ($0.3^{10} \approx 6\times10^{-6}$)



Training the Cascade

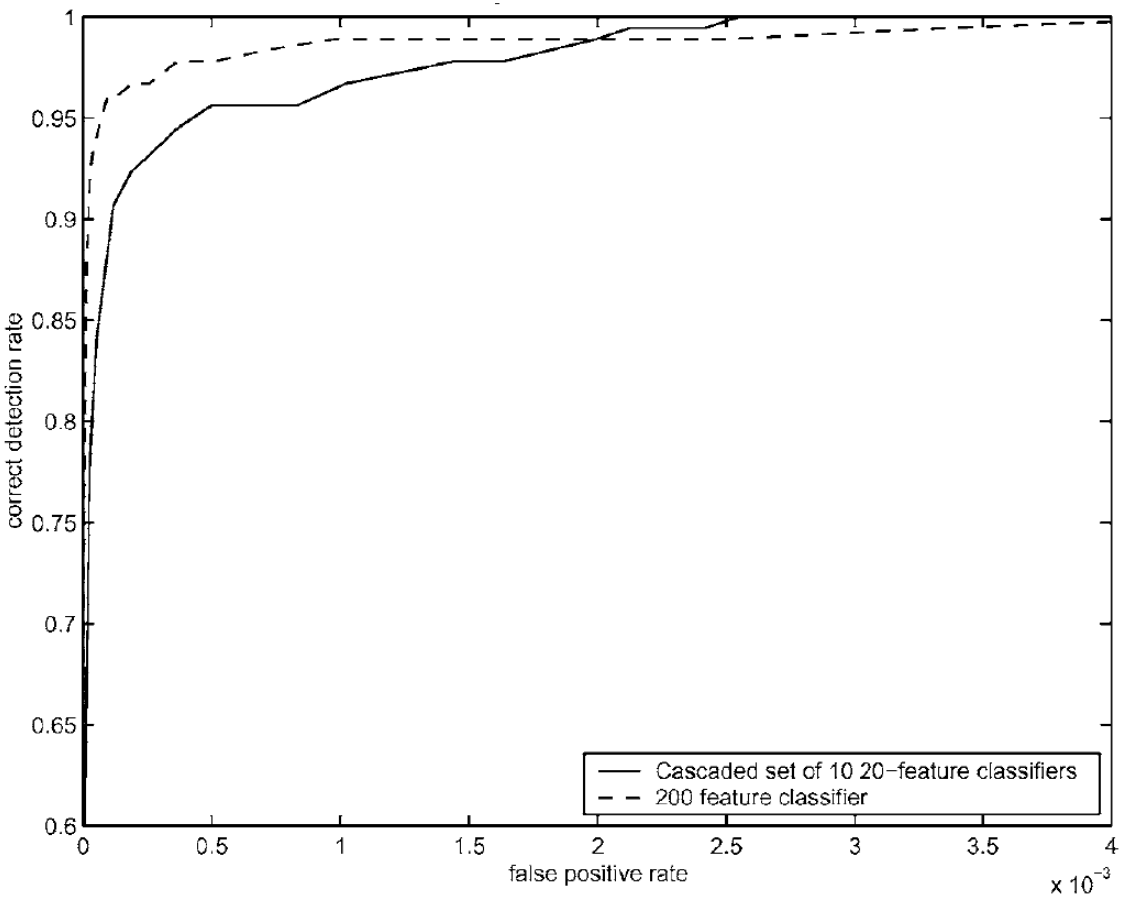
- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is *not* low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

Trainin

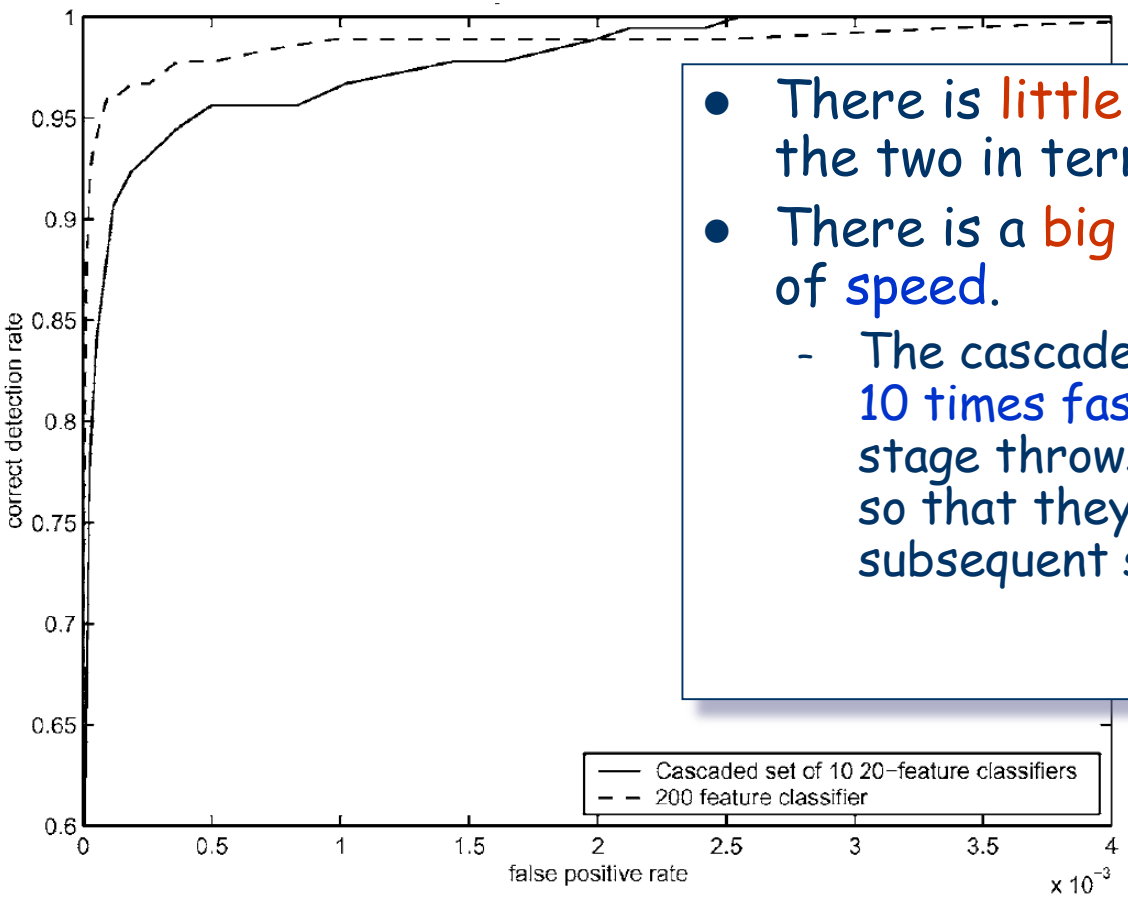
Table 2. The training algorithm for building a cascaded detector.

- User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, F_{target}.
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
  - $i \leftarrow i + 1$
  - $n_i = 0$; $F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use P and N to train a classifier with n_i features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine F_i and D_i.
    - Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)
  - $N \leftarrow \emptyset$
  - If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N

ROC Curves Cascaded Classifier to Monlithic Classifier



ROC Curves Cascaded Classifier to Monlithic Classifier



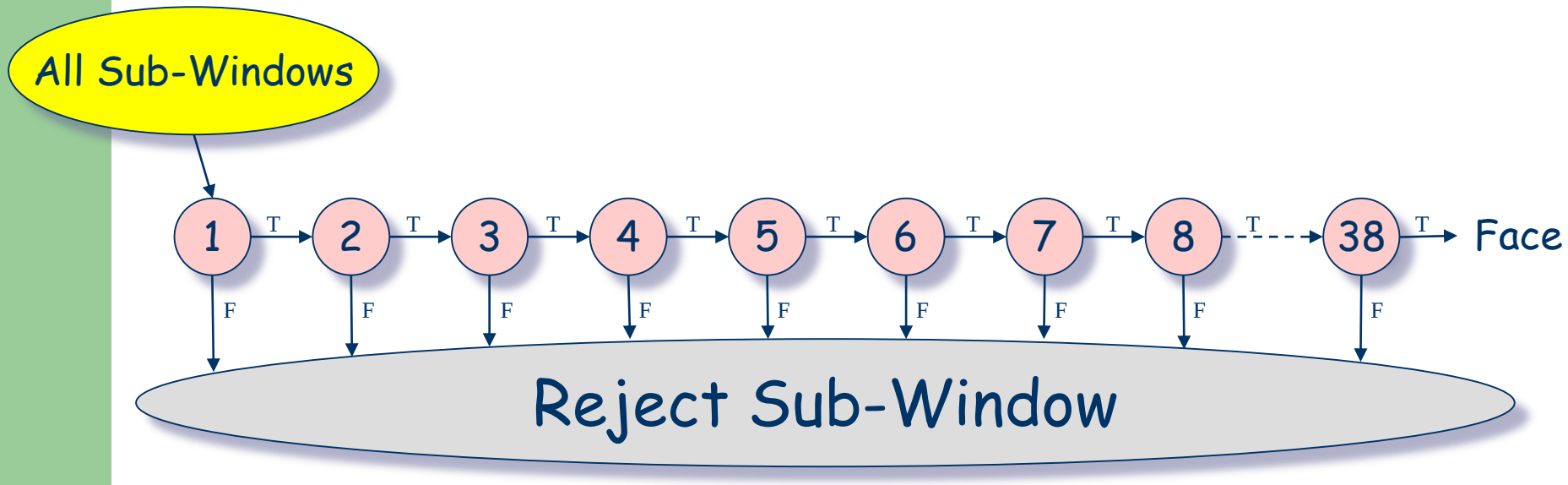
- There is little difference between the two in terms of accuracy.
- There is a big difference in terms of speed.
  - The cascaded classifier is nearly 10 times faster since its first stage throws out most non-faces so that they arenever evaluated by subsequent stages.

The Implemented System

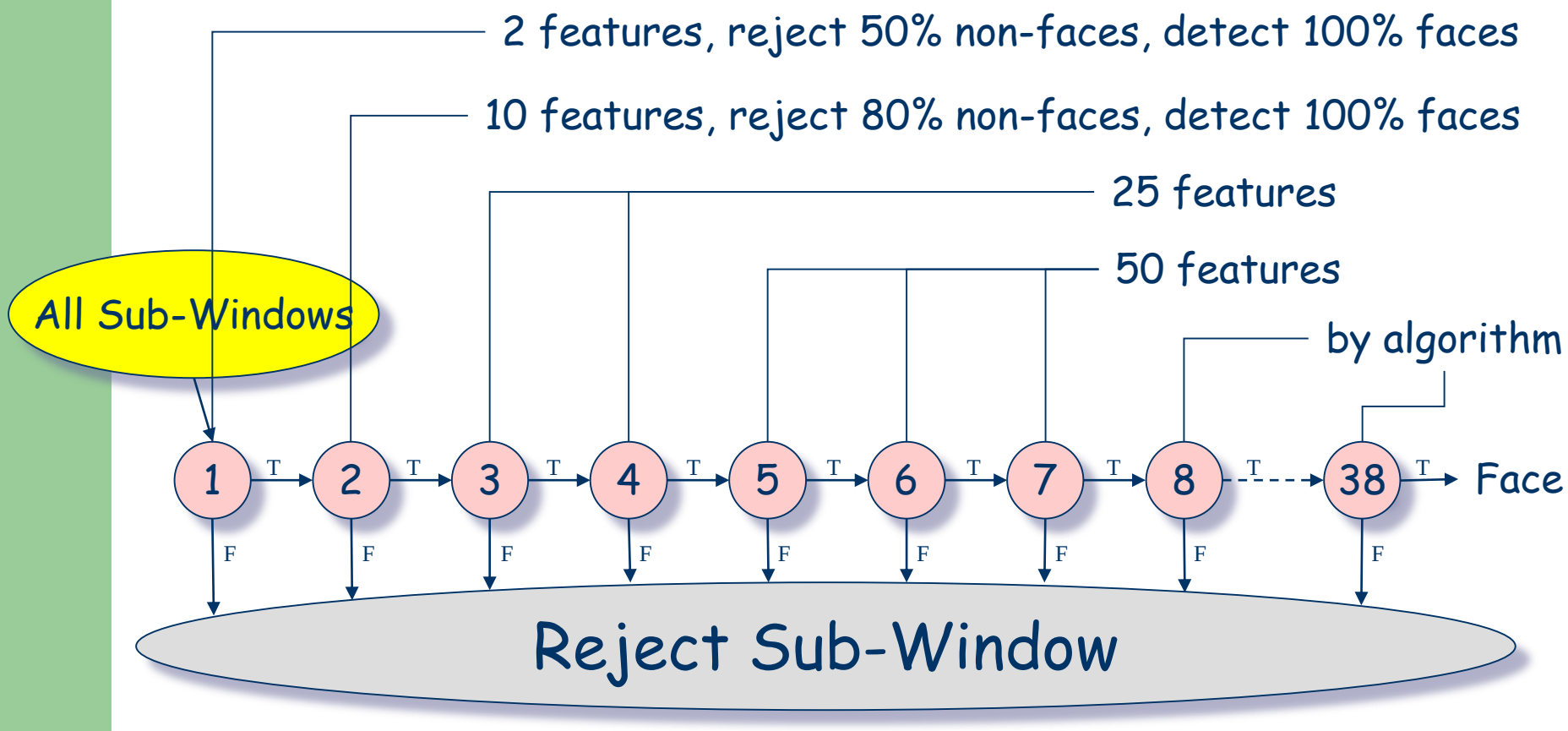
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

Structure of the Detector Cascade

- Combining successively more complex classifiers in cascade
  - 38 stages
  - included a total of 6060 features



Structure of the Detector Cascade



Speed of the Final Detector

- On a 700 Mhz Pentium III processor, the face detector can process a 384 ×288 pixel image in about .067 seconds"
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

- Average of 8 features evaluated per window on test set

Image Processing

- Training — all example sub-windows were variance normalized to minimize the effect of different lighting conditions
- Detection — variance normalized as well

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

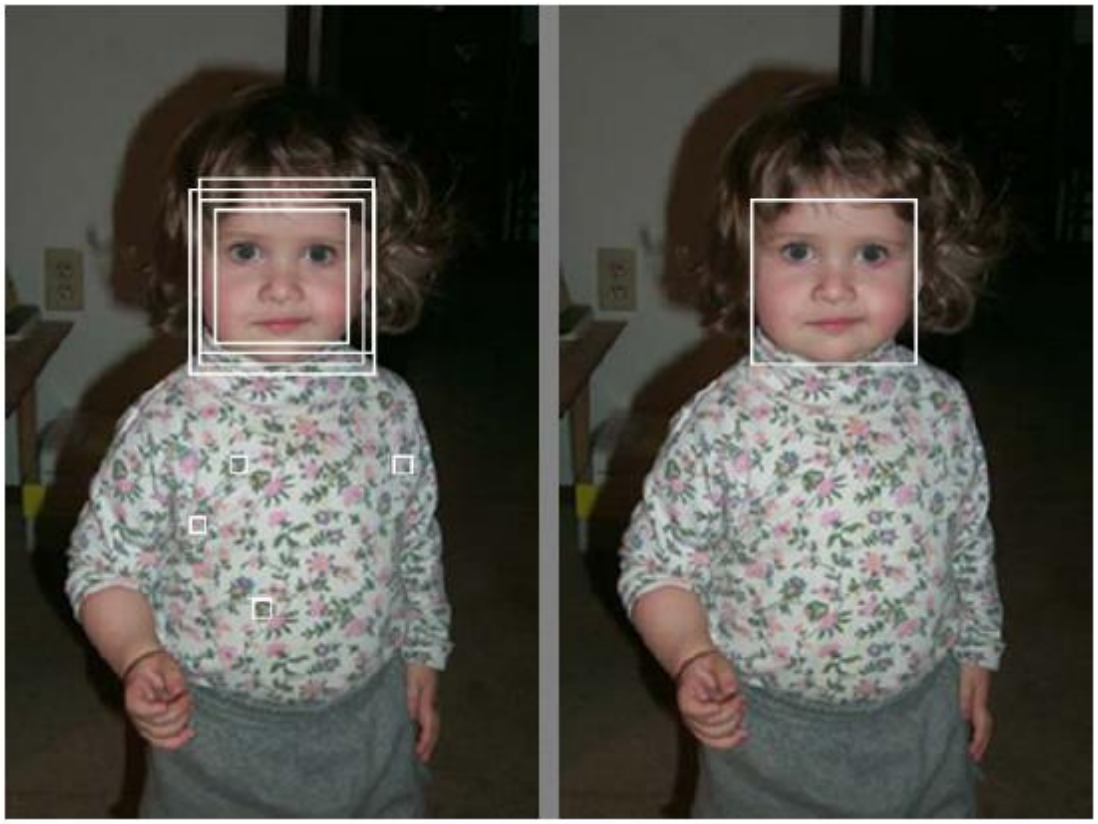
Can be computed using integral image

Can be computed using integral image of squared image

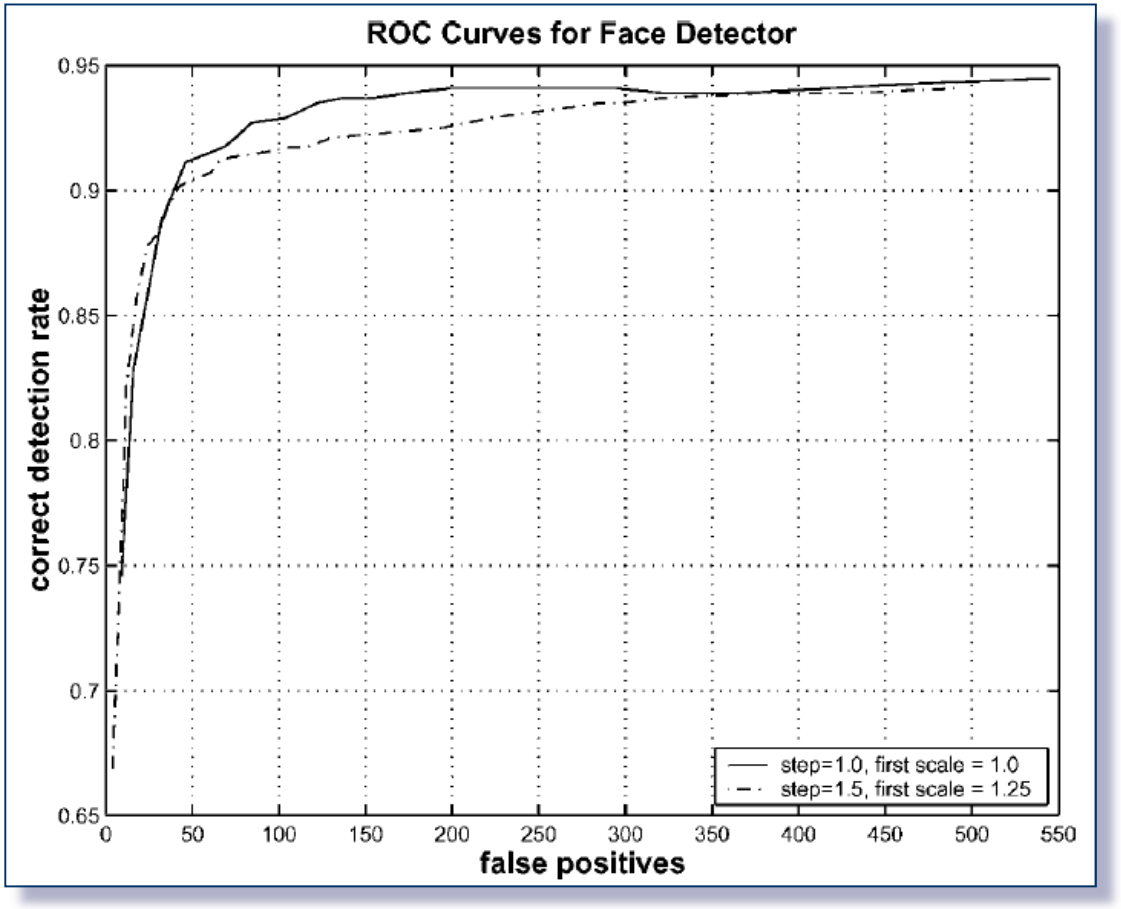
Scanning the Detector

- Scaling is achieved by scaling the detector itself, rather than scaling the image
- Good detection results for scaling factor of 1.25
- The detector is scanned across location
  - Subsequent locations are obtained by shifting the window $[s\Delta]$ pixels, where s is the current scale
  - The result for $\Delta = 1.0$ and $\Delta = 1.5$ were reported

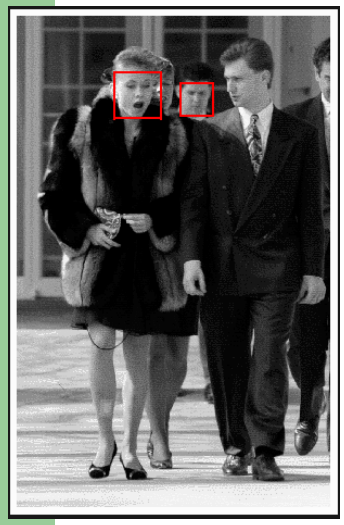
Merging Multiple Detections

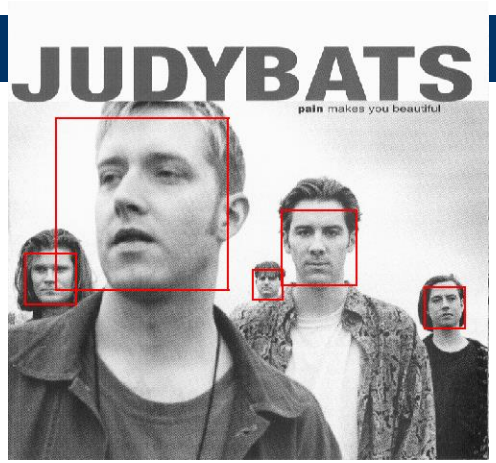


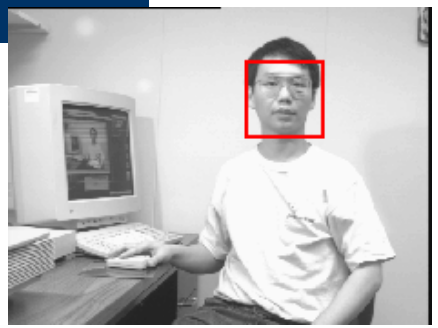
ROC Curves for Face Detection

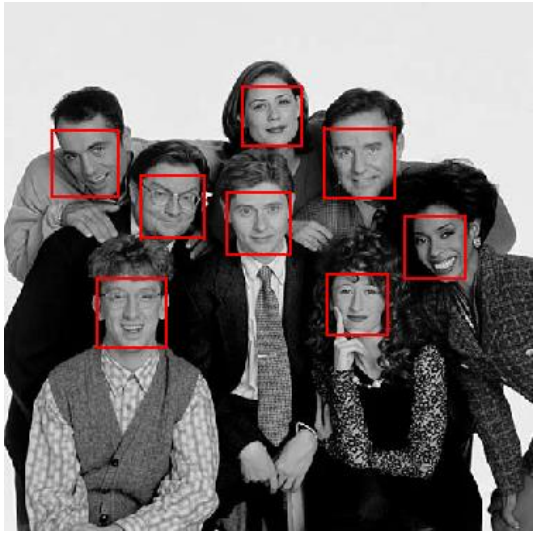


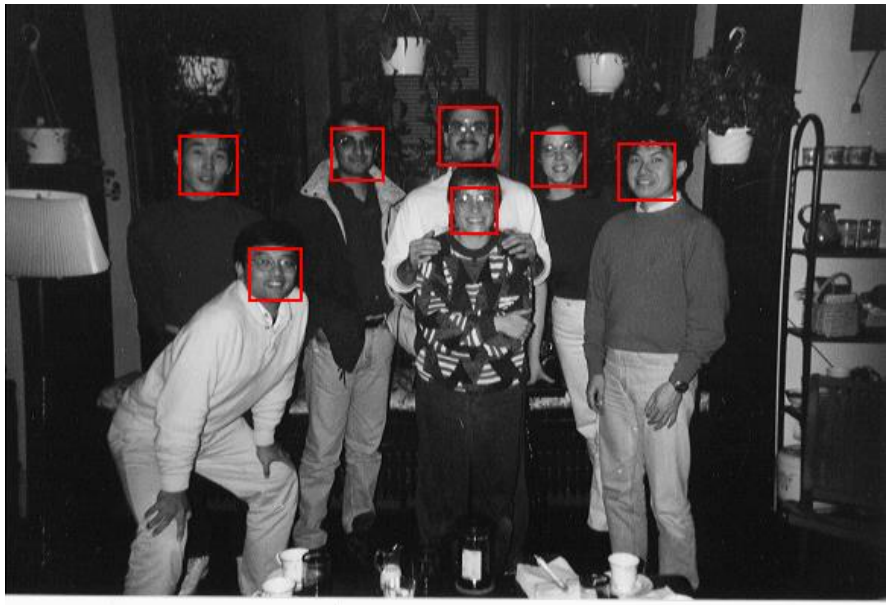
Output of Face Detector on Test Images





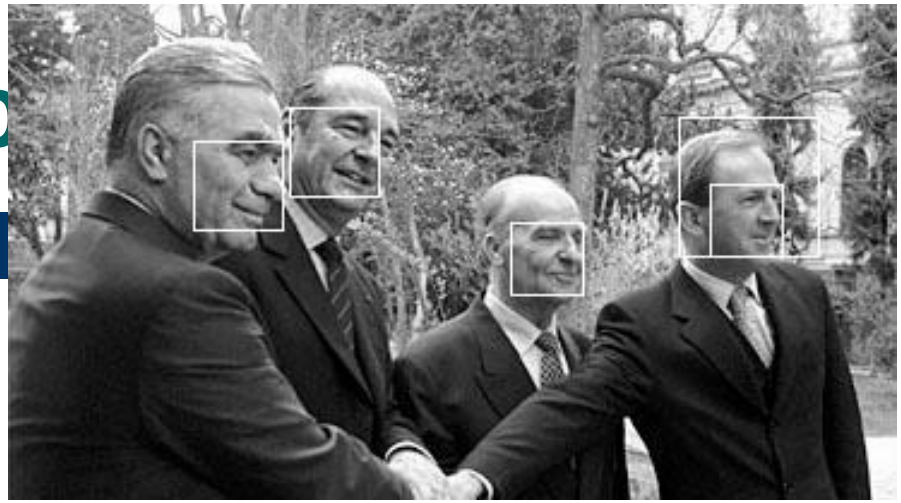






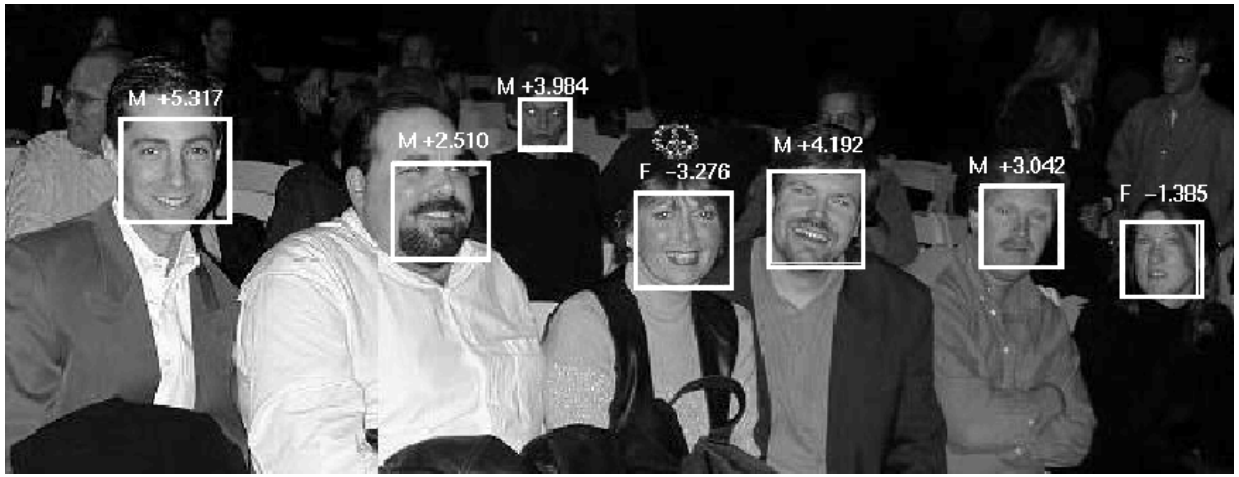
tic

Facial Feature Localization



Profile Detection

Male vs. female

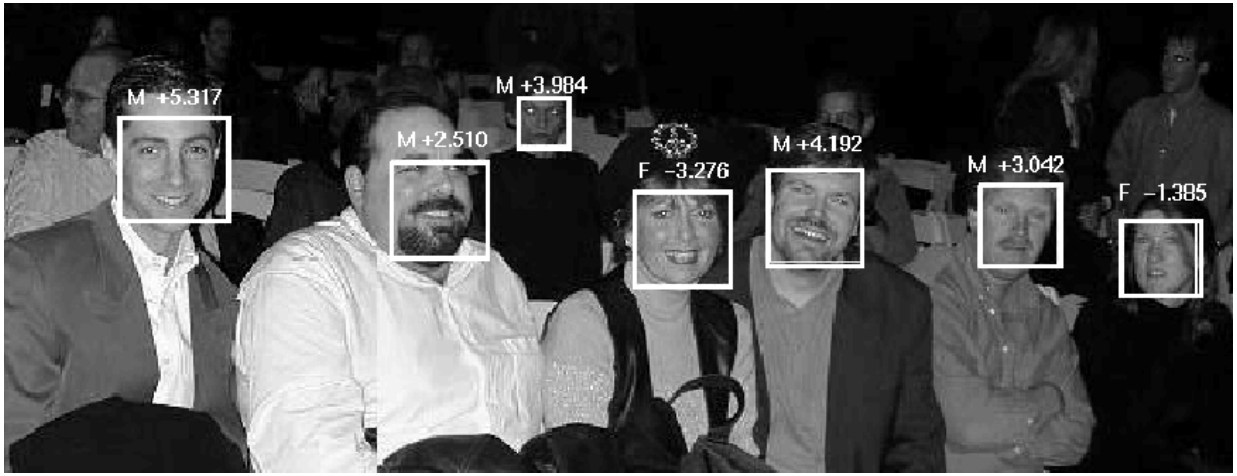


Other Detection Tasks

Facial Feature Localization

Profile Detection

Other Detection Tasks



Male vs. Female

Conclusions

- How adaboost works?
- Why adaboost works?
- Adaboost for face detection
  - Rectangle features
  - Integral images for fast computation
  - Boosting for feature selection
  - Attentional cascade for fast rejection of negative windows