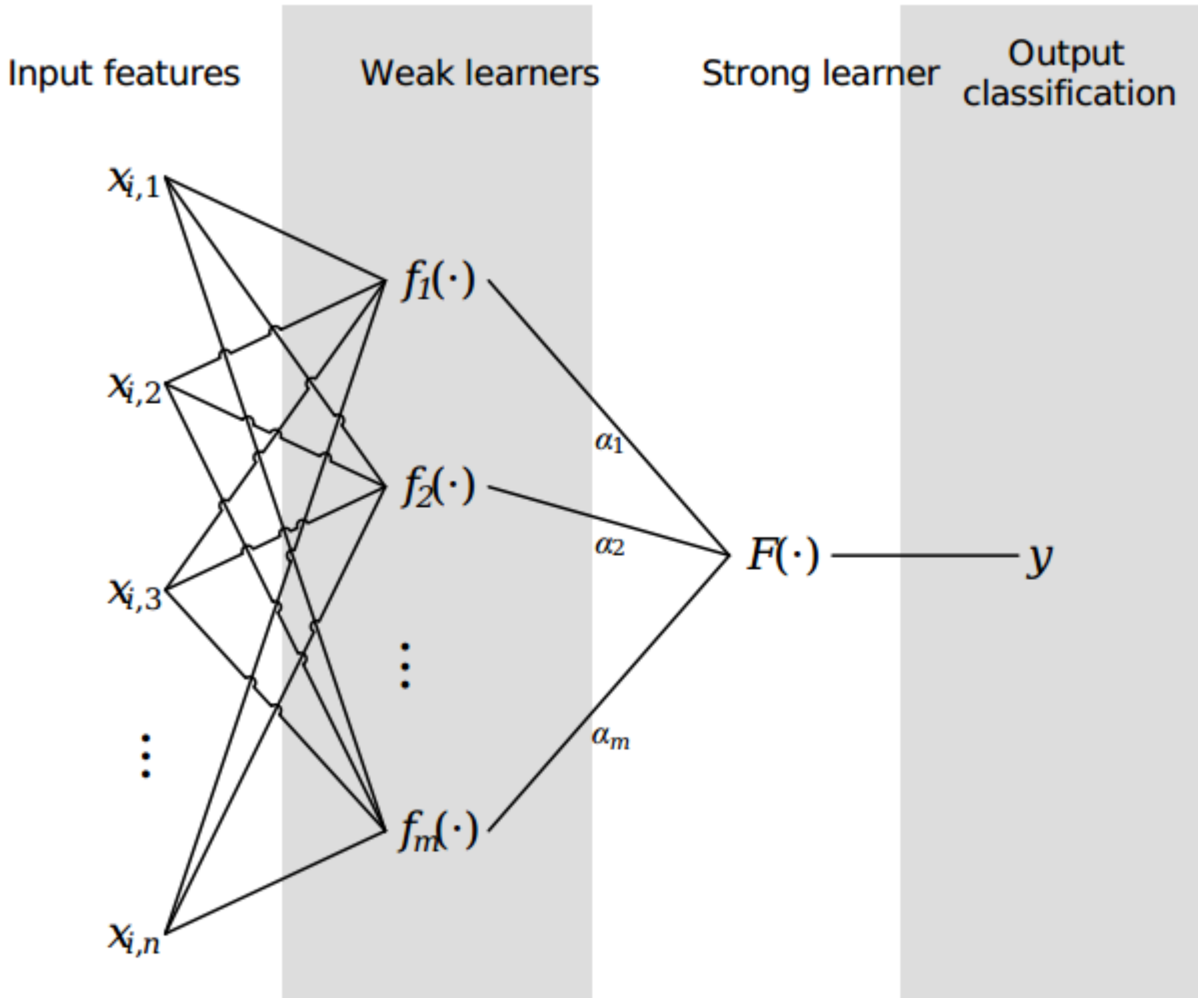
Tutorial 10: Boosting

Rui Zhao
rzhao@ee.cuhk.edu.hk

As a neural network



Pseudo code

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

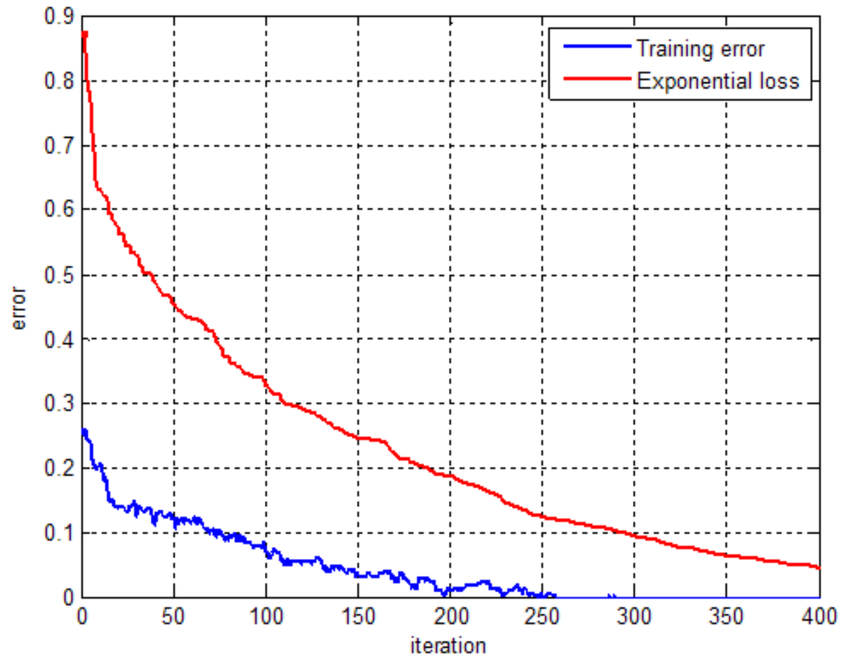
Output the final classifier:

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

Matlab code for adaboost

Matlab code and toy data is provided in the supplementary materials:

'adaboost_demo.m'
'test_data.mat'



Other tools for boosting

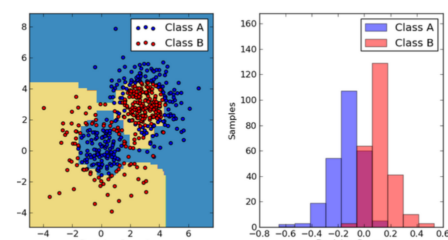
Classic adaboost classifier:

Matlab

http://www.mathworks.com/matlabcentral/fileexchange/27813-classic-adaboost-classifier

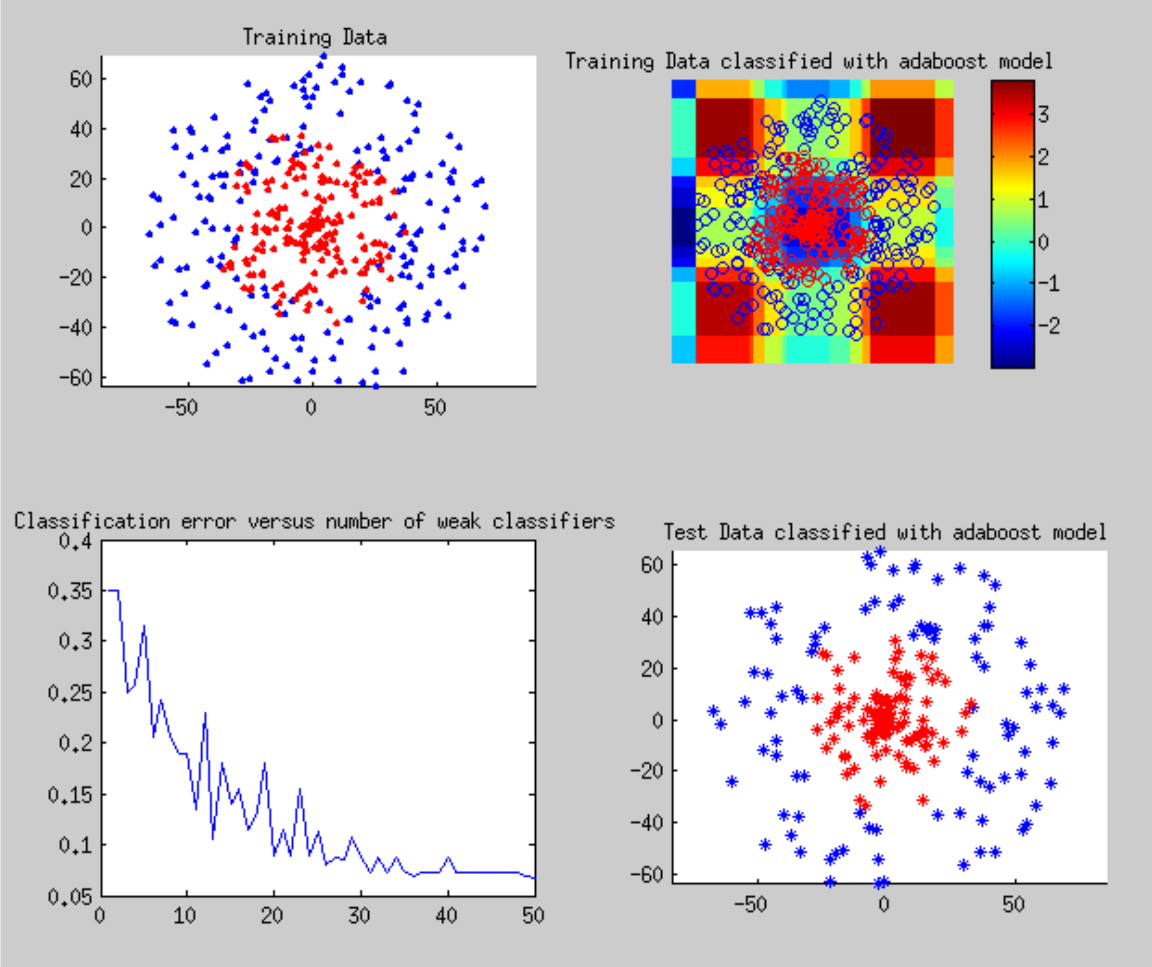
Python

http://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html



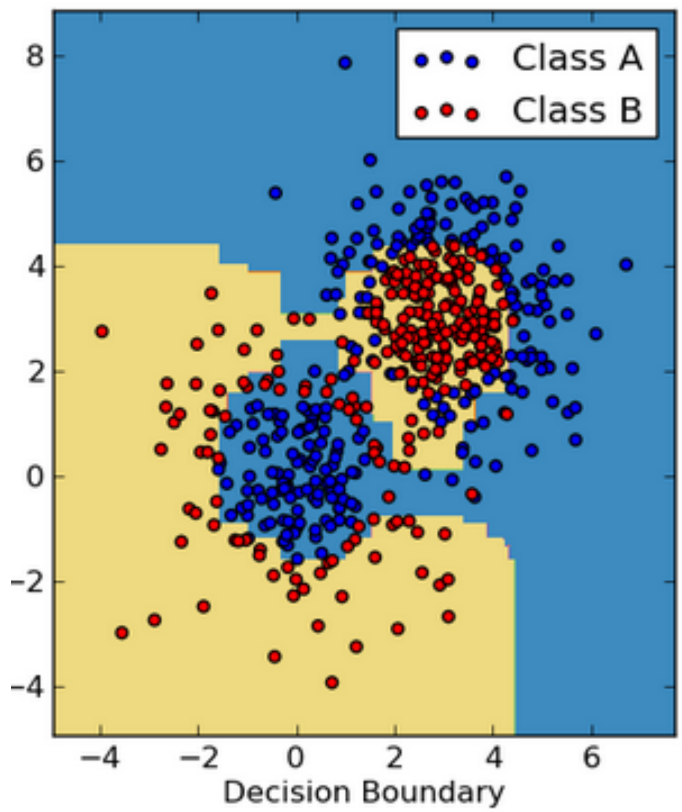
Matlab code for adaboost

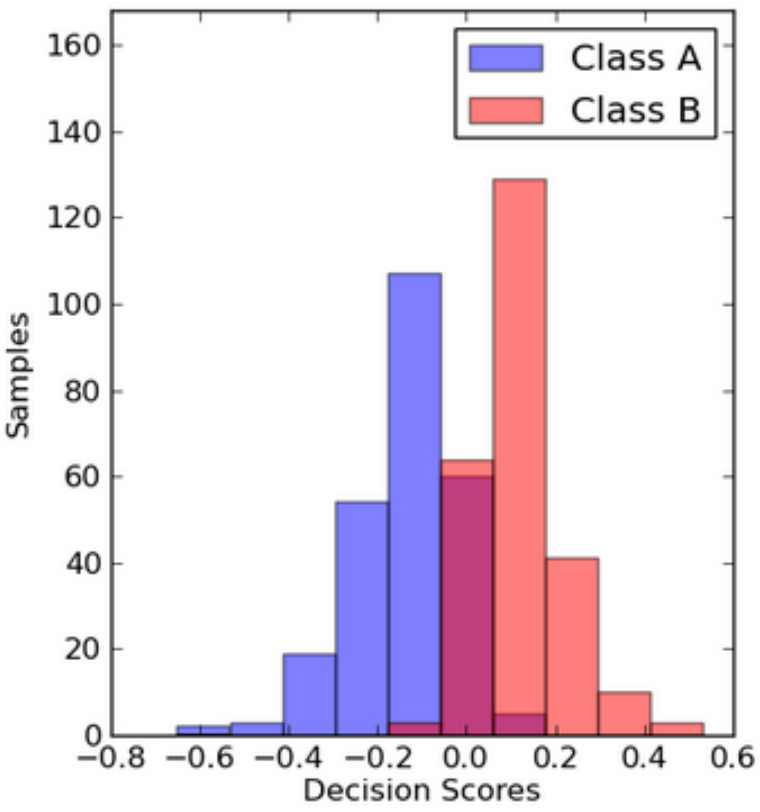
http://www.mathworks.com/matlabcentral/fileexchange/27813-classic-adaboost-classifier



Python code for adaboost

http://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html





Famous Applications

Introduction to Jones and Viola's work on face detection using adaboost.

Following slides are borrowed.

Cos 429: Face Detection (Part 2)
Viola-Jones and AdaBoost

Guest Instructor: Andras Ferencz
(Your Regular Instructor: Fei-Fei Li)

Thanks to Fei-Fei Li, Antonio Torralba, Paul Viola, David Lowe, Gabor Melli (by way of the Internet) for slides

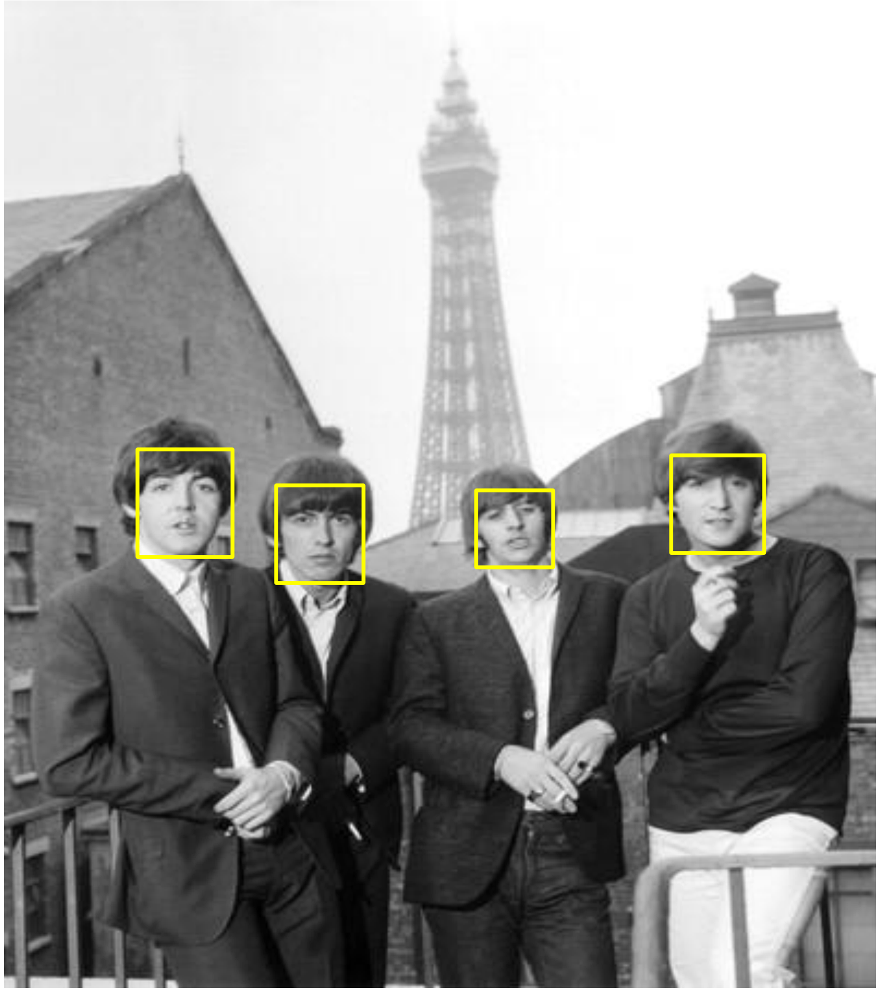
Cos 429: Face Detection (Part 2) Viola-Jones and AdaBoost

Guest Instructor: Andras Ferencz
(Your Regular Instructor: Fei-Fei Li)

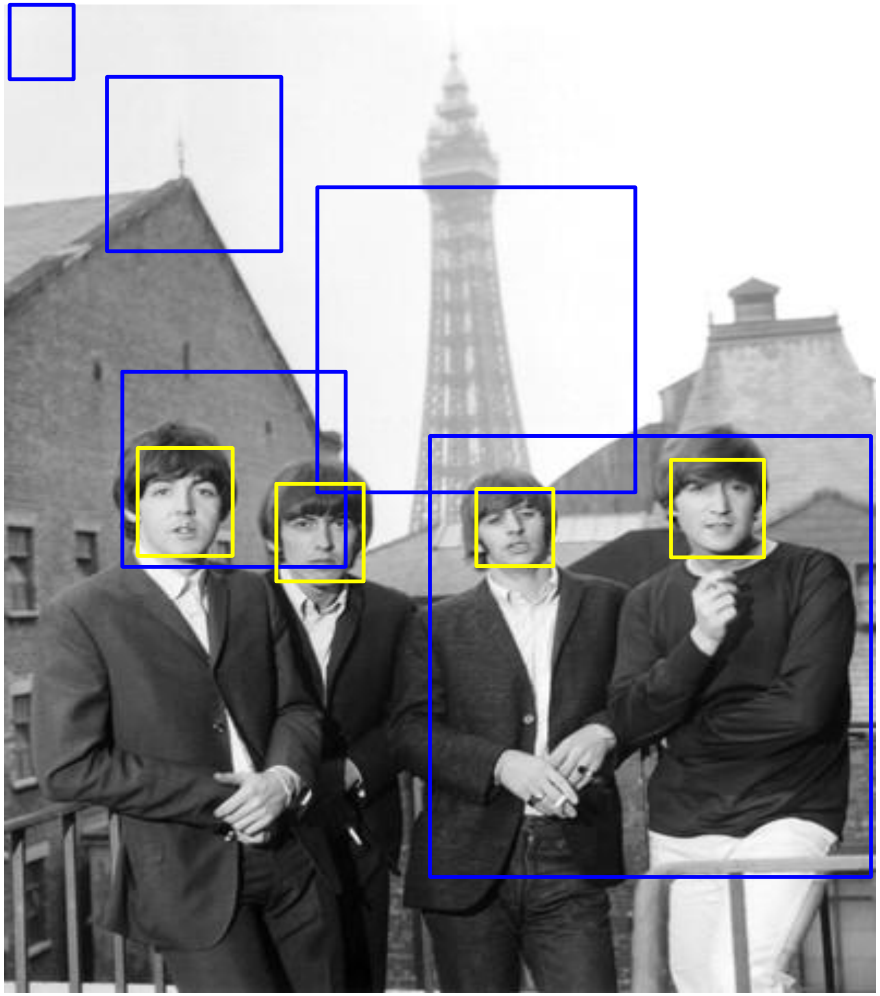
Thanks to Fei-Fei Li, Antonio Torralba, Paul Viola, David Lowe, Gabor Melli (by way of the Internet) for slides

Face Detection

Face Detection



Sliding Windows



1. hypothesize:

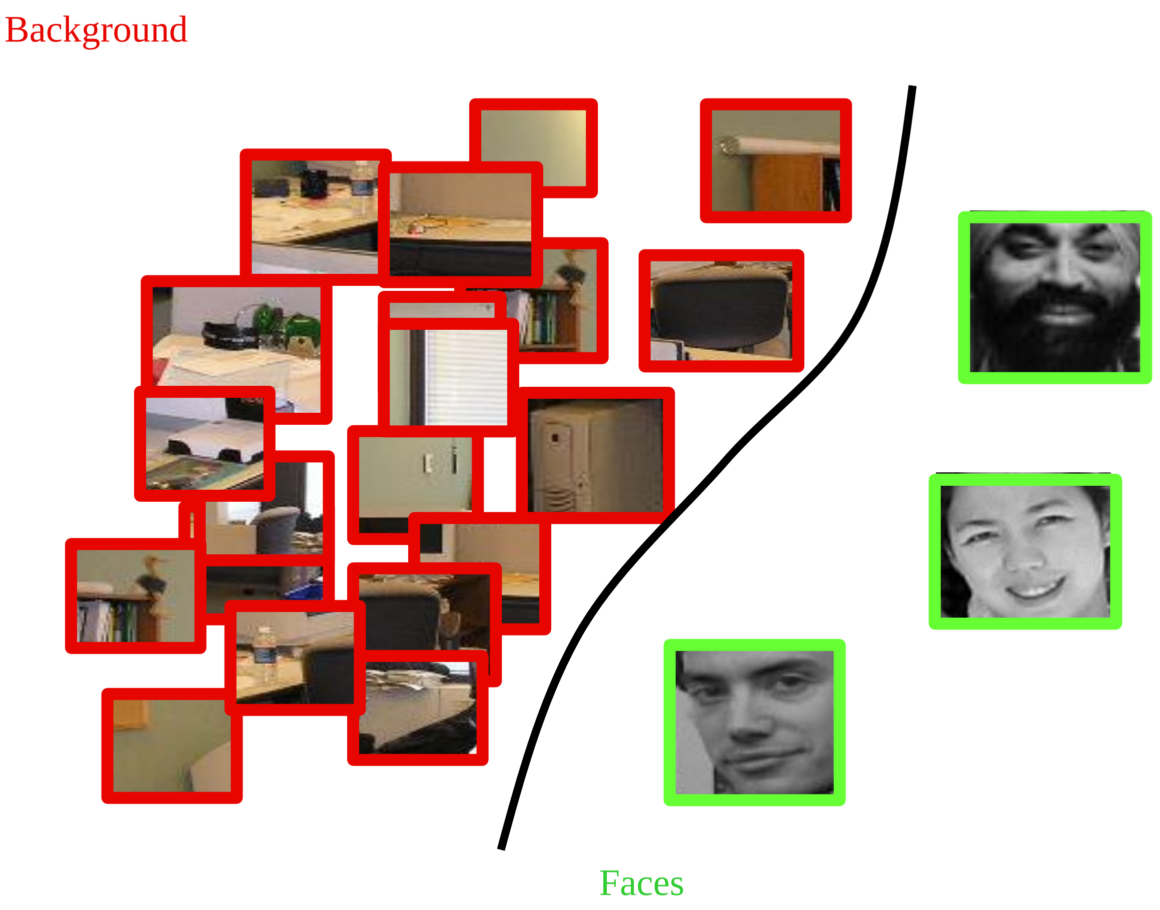
try all possible rectangle locations, sizes

2. test:

classify if rectangle contains a face (and only the face)

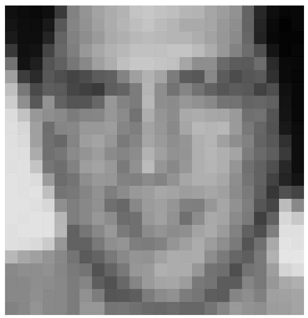
Note: 1000's more false windows then true ones.

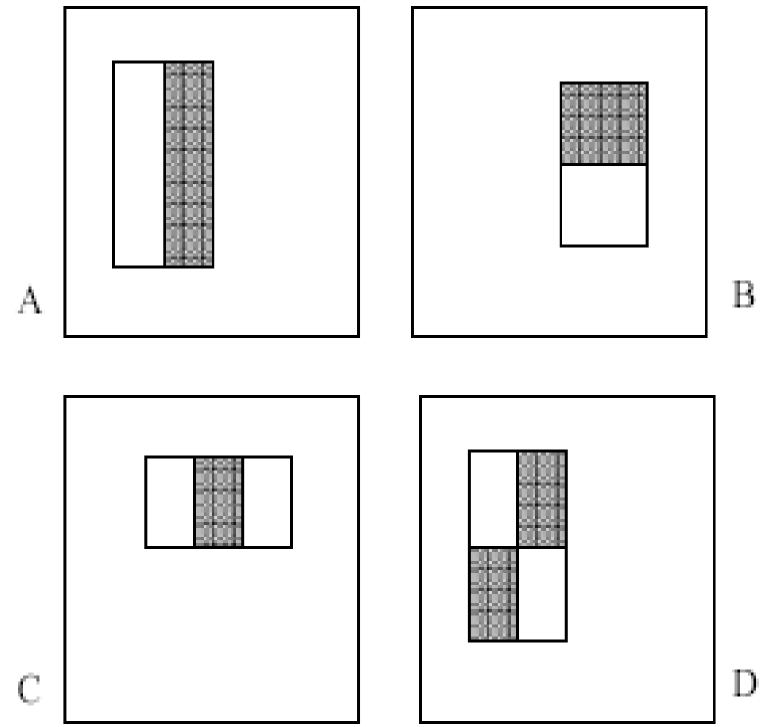
Classification (Discriminative)



In some feature space

Image Features



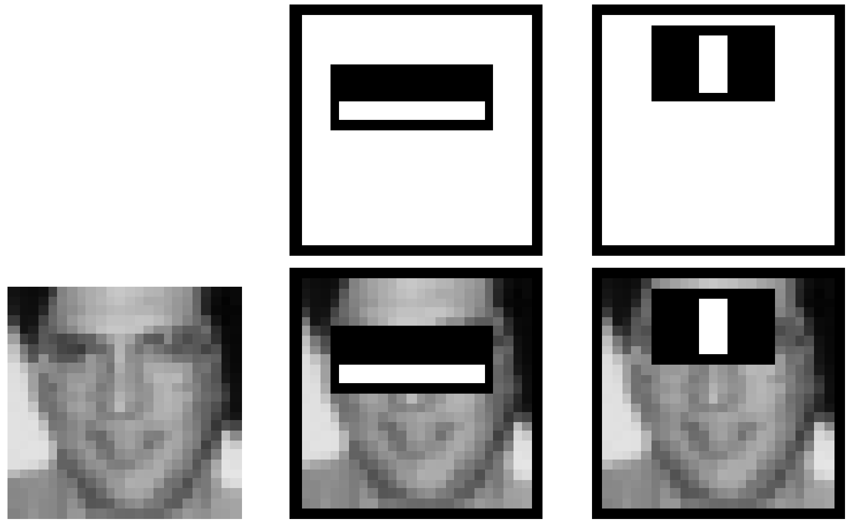


4 Types of “Rectangle filters”
(Similar to Haar wavelets
Papageorgiou, et al.)

Based on 24x24 grid:
160,000 features to choose from

g(x) =
sum(WhiteArea) - sum(BlackArea)

Image Features

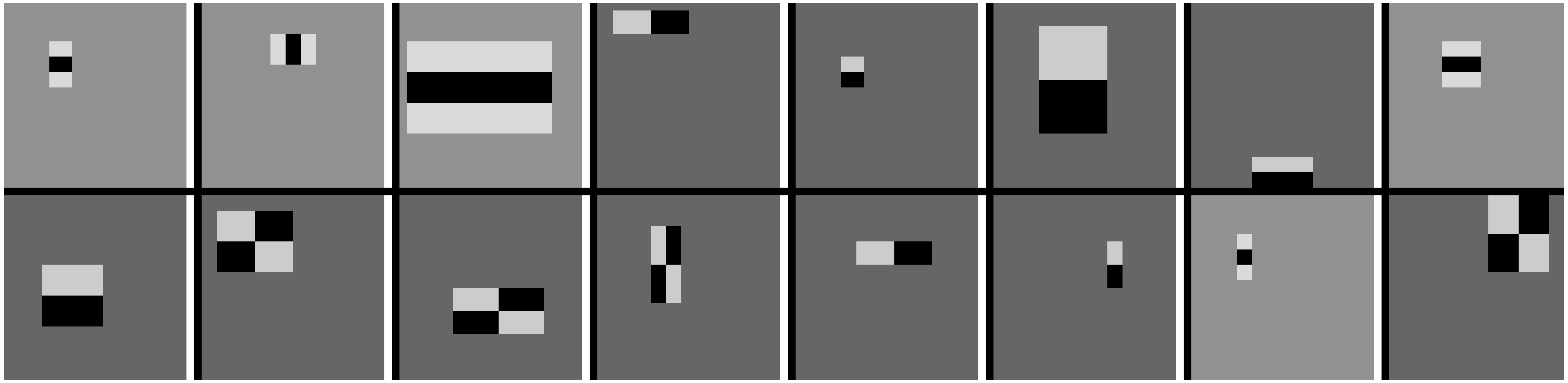


$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \ldots$$

$$f_i(x) = \begin{cases} 1 & \text{if } g_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases}$$

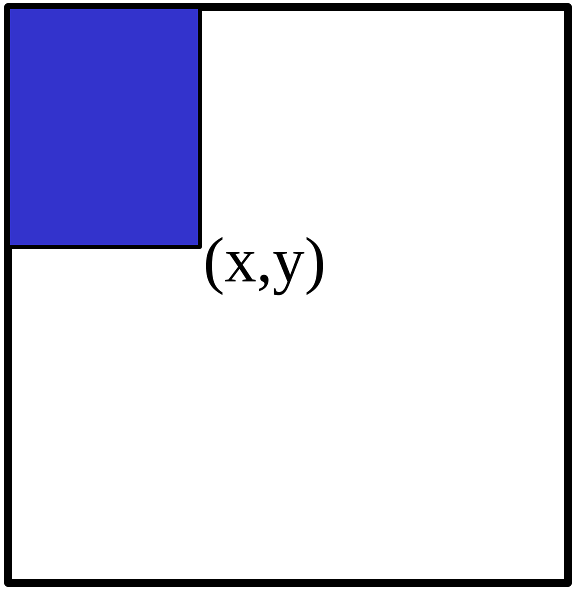
Need to: (1) Select Features i=1..n,
(2) Learn thresholds θ_i ,
(3) Learn weights α_i

A Peak Ahead: the learned features



Why rectangle features? (1)
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

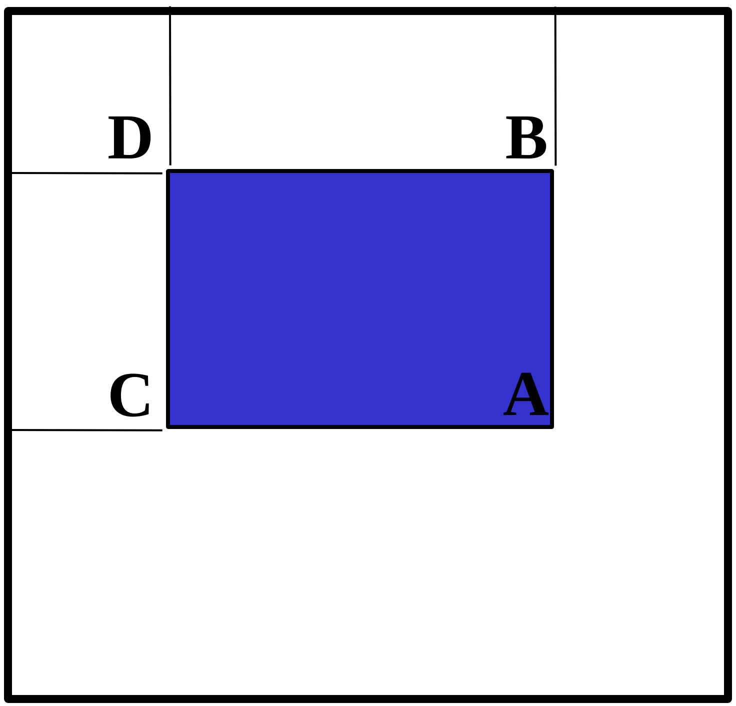


Why rectangle features? (2)
Computing Sum within a Rectangle

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed:

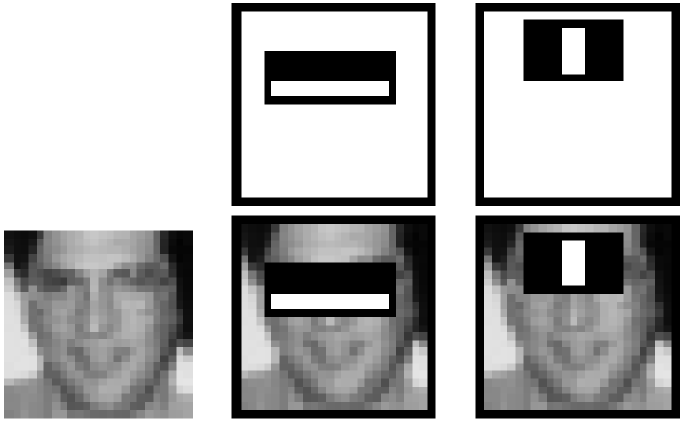
  sum = A – B – C + D

- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision



Boosting

How to select the best features?



How to learn the classification function?

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \ldots.$$

Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

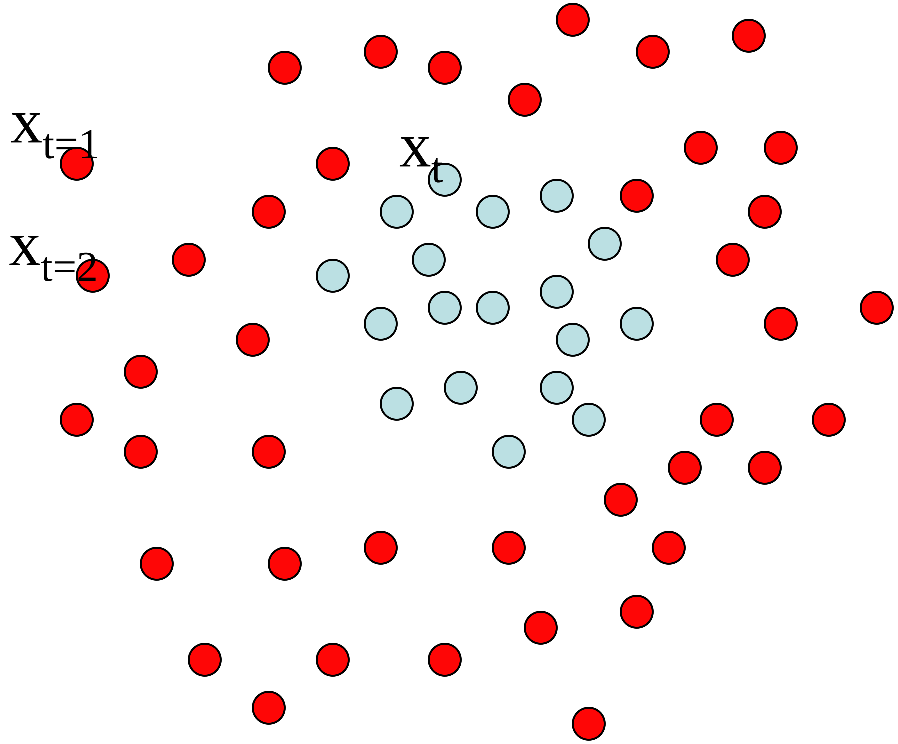
Features vector

Weight

Weak classifier

Boosting

- It is a sequential procedure:



Each data point has

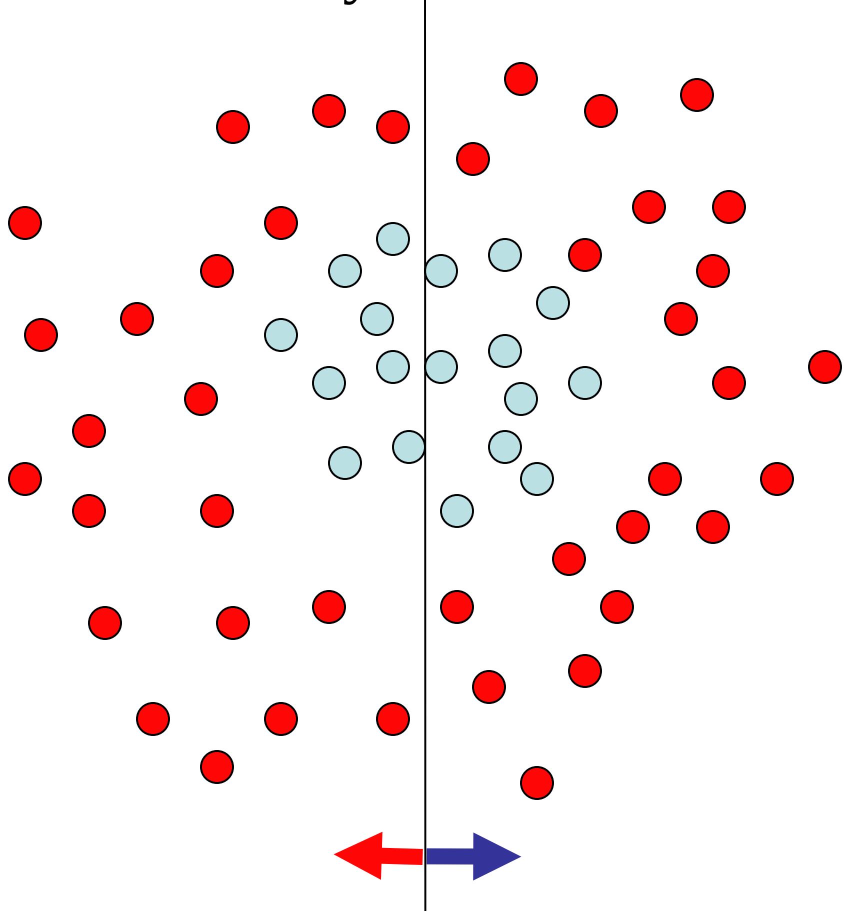
a class label: $y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$

and a weight:

$w_t = 1$

Toy example

Weak learners from the family of lines



Each data point has

a class label:

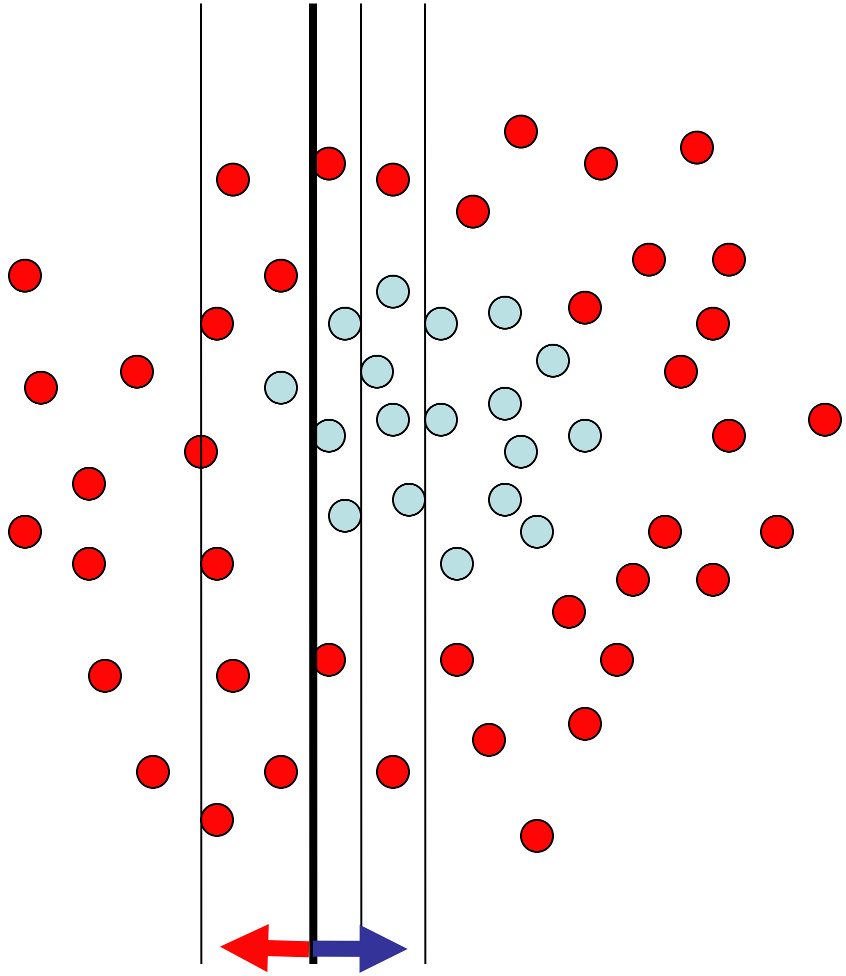
$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$$

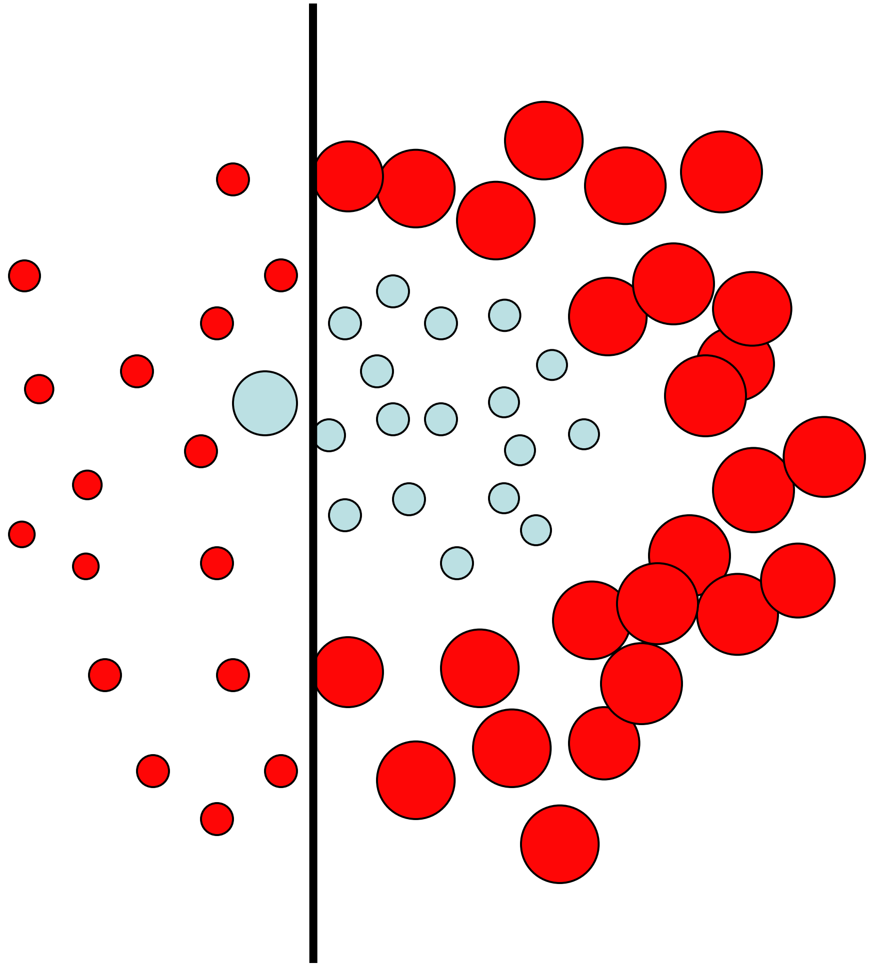
and a weight:

$w_t = 1$

This one seems to be the best

This is a '**weak classifier**': It performs slightly better than chanc

Toy example



Each data point has a class label:

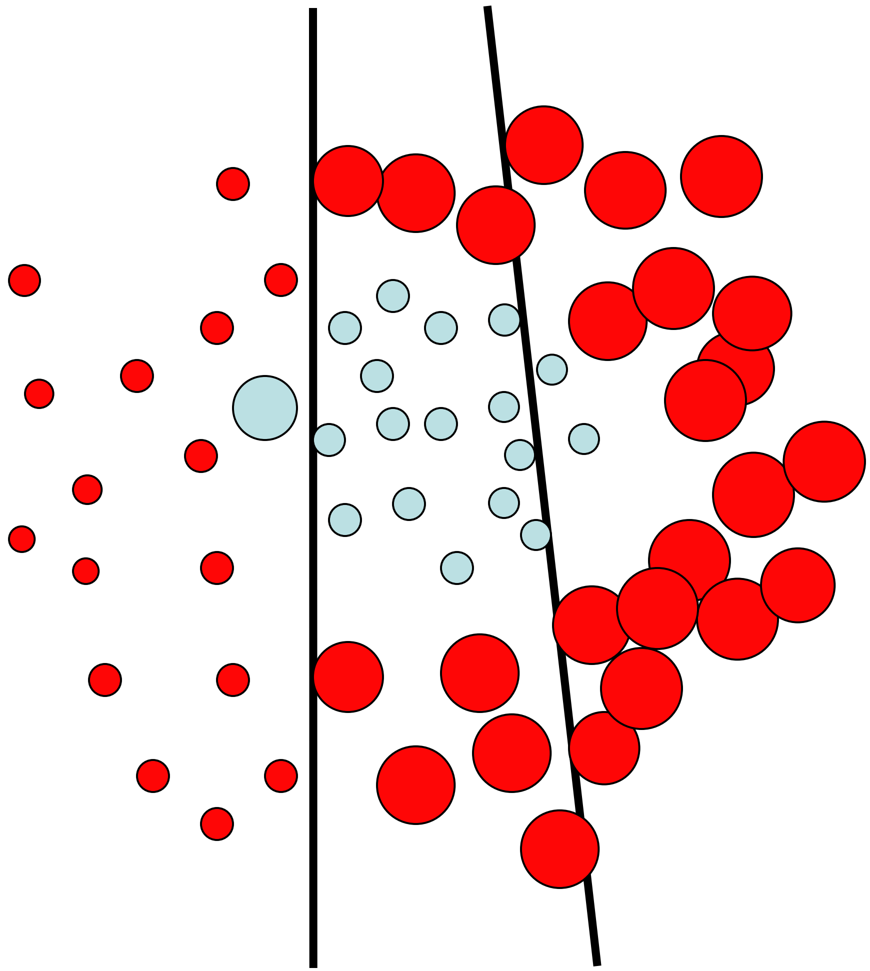
$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at

Toy example



Each data point has a class label:

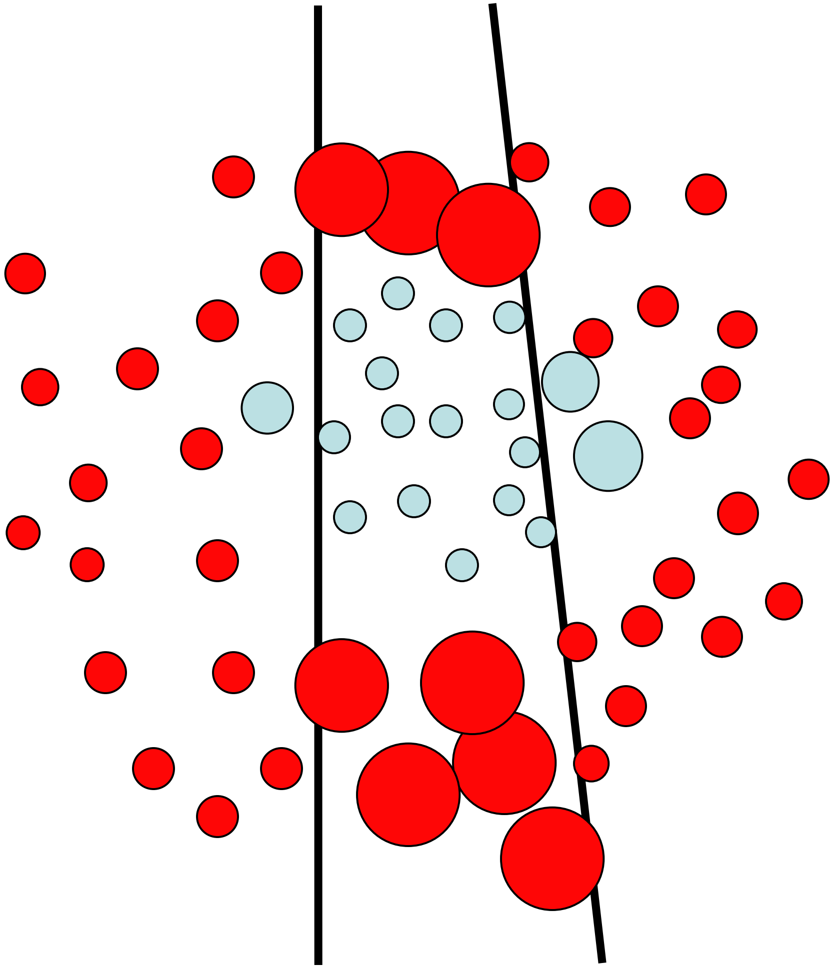
$$y_t = \begin{cases} +1 \; (\quad) \\ -1 \; (\quad) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at c

Toy example



Each data point has a class label:

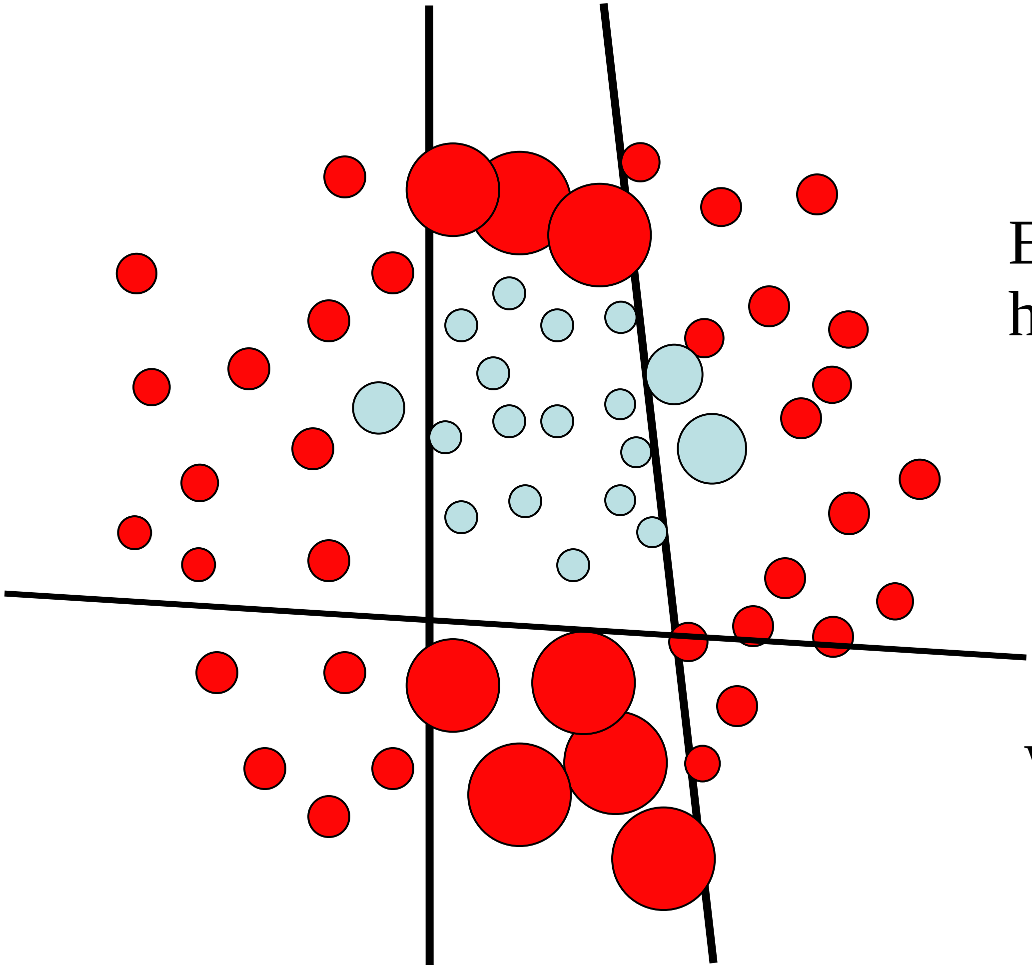
$$y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at c

Toy example



Each data point has a class label:

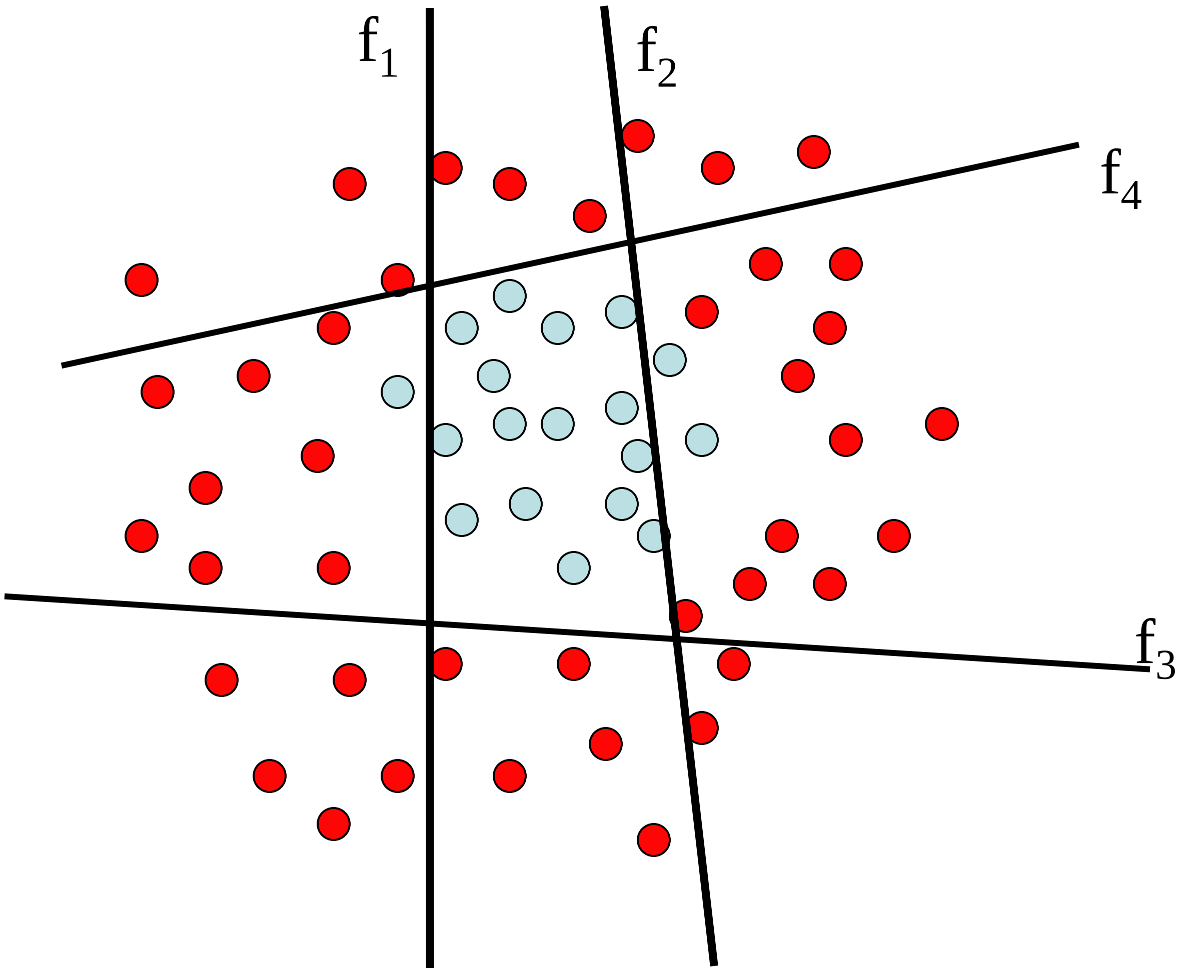
$$y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at c

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

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

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

The goodness of h_t is calculated over D_t and the bad guesses.

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.

Boosting with Rectangle Features

- For each round of boosting:
  - Evaluate each rectangle filter on each example (compute g(x))
  - Sort examples by filter values
  - Select best threshold (θ) for each filter (one with lowest error)
  - Select best filter/threshold combination from all candidate features (= Feature f(x))
  - Compute weight (α) and incorporate feature into strong classifier
    $F(x) \leftarrow F(x) + \alpha f(x)$
  - Reweight examples

Boosting

Boosting fits the additive model

$$F(x) = f_1(x) + f_2(x) + f_3(x) + \ldots$$

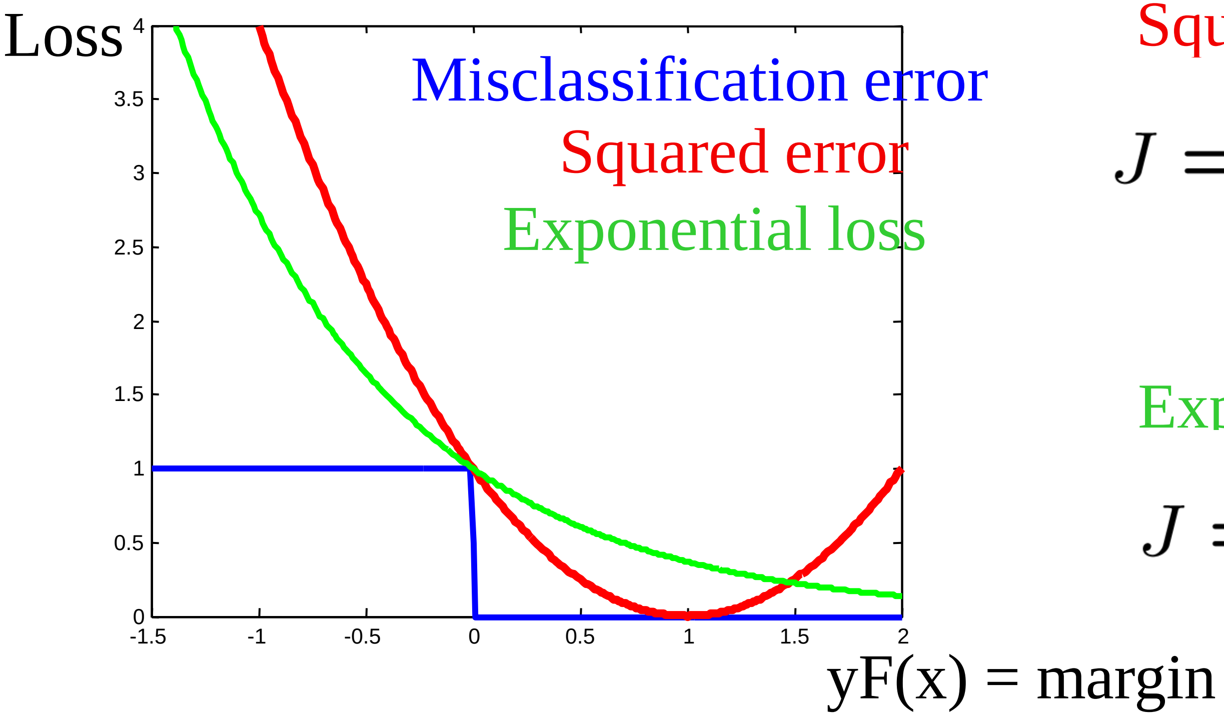
by minimizing the exponential loss

$$J(F) = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Training samples

The exponential loss is a differentiable upper bound to the misclassification error.

Exponential loss



Squared error

$$J = \sum_{t=1}^{N} [y_t - F(x_t)]^2$$

Exponential loss

$$J = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Boosting

Sequential procedure. At each step we add

$$F(x) \leftarrow F(x) + f_m(x)$$

to minimize the residual loss

$$(\phi_m) = \arg\min_{\phi} \sum_{t=1}^{N} J\left(y_i, F(x_t) + f(x_t; \phi)\right)$$

Parameters weak classifier

Desired output

input

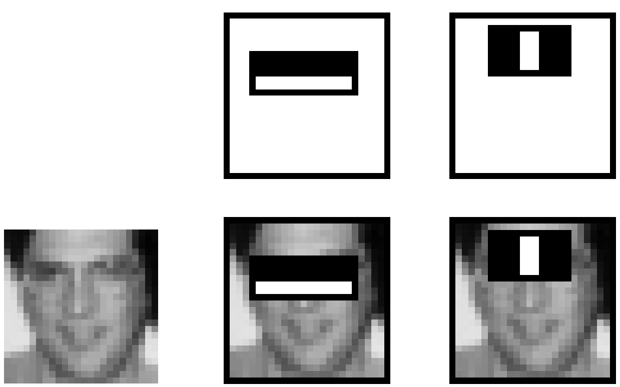
For more details: Friedman, Hastie, Tibshirani. "Additive Logistic Regression: a Statistical View of Boosting" (1998)

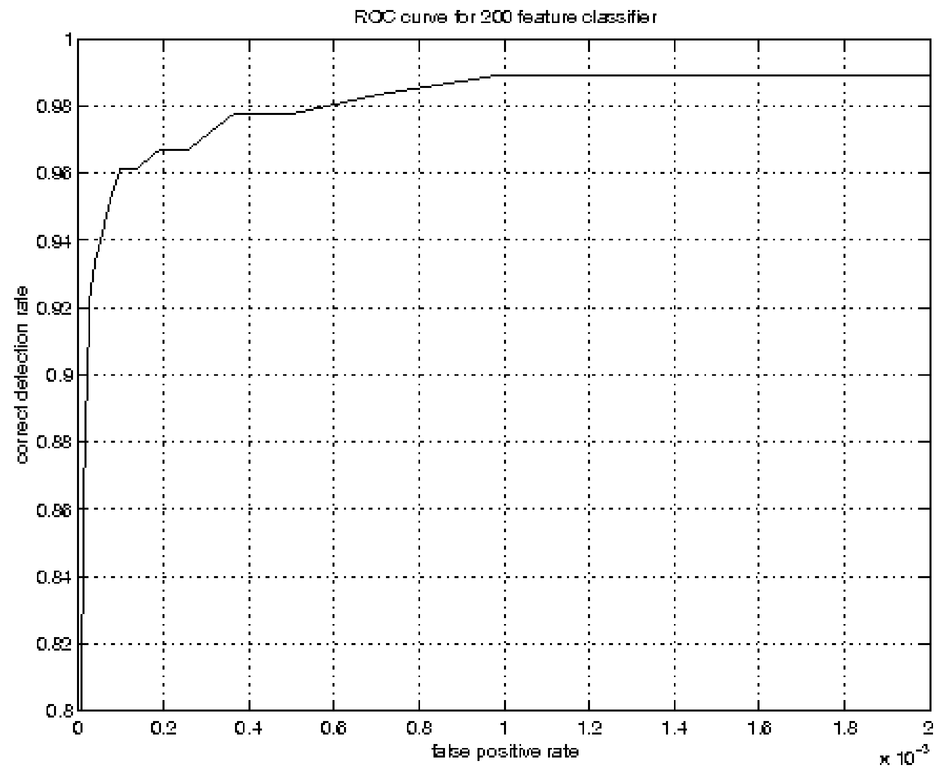
Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084
false positives.

Not quite competitive...

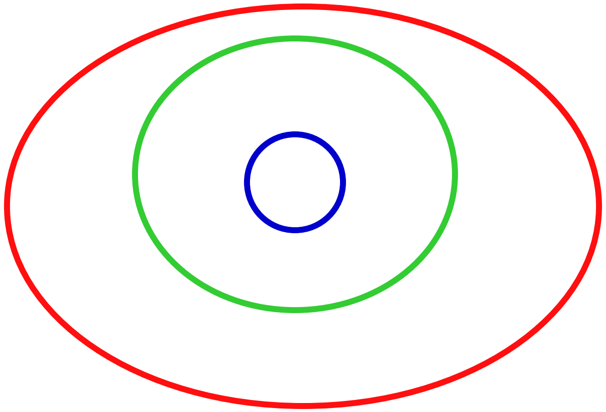


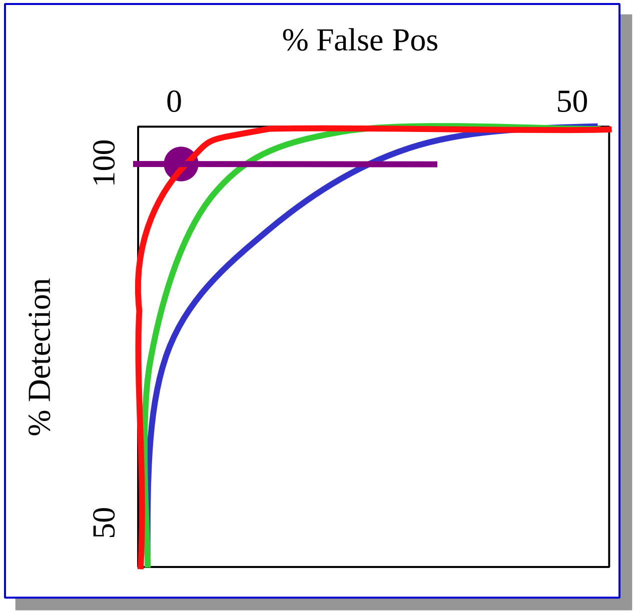


ROC curve for 200 feature classifier

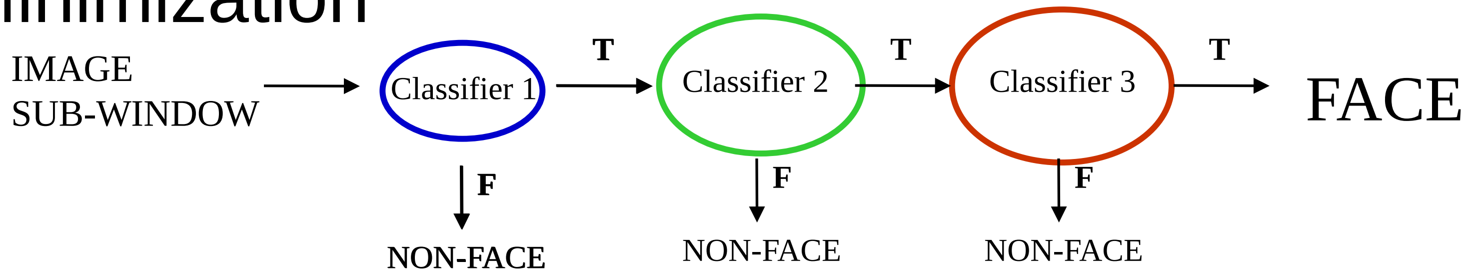
Building Fast Classifiers

- Given a nested set of classifier hypothesis classes

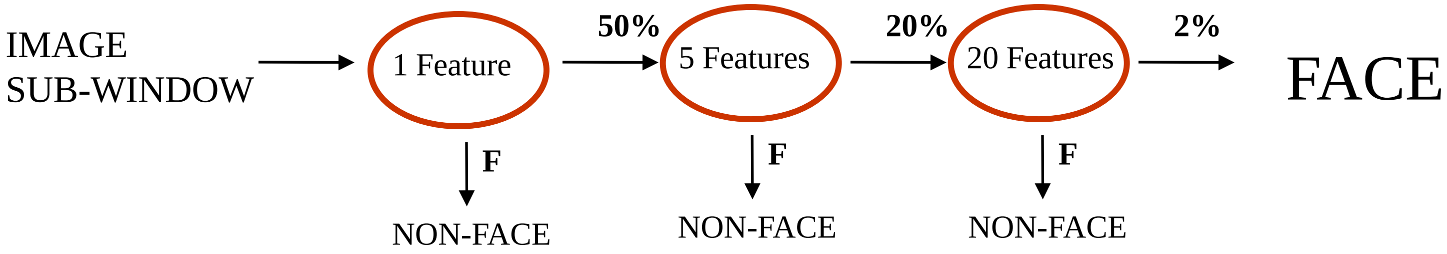




- Computational Risk Minimization

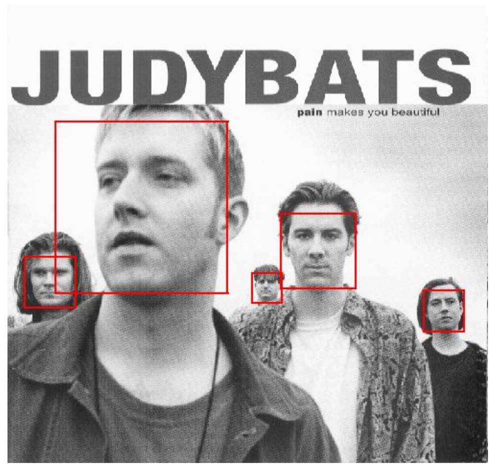


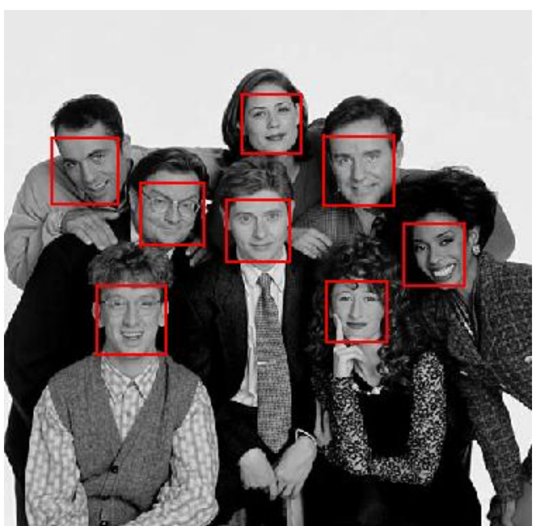
Cascaded Classifier

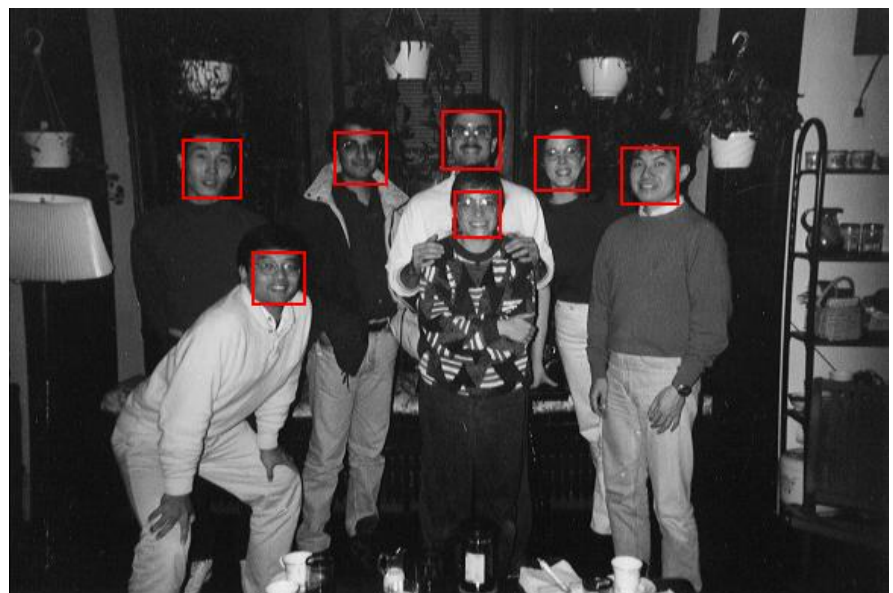


- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Output of Face Detector on Test Images

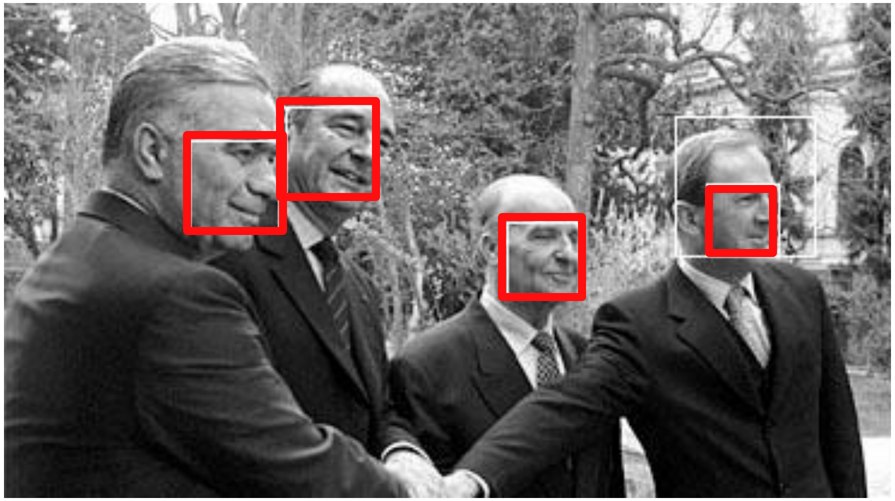






Solving other “Face” Tasks

Facial Feature Localization



Profile Detection

Demographic
Analysis