Face Detection

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Roberto Capobianco

Special thanks to:
Prof. Domenico Bloisi
Dr. Jacopo Serafin

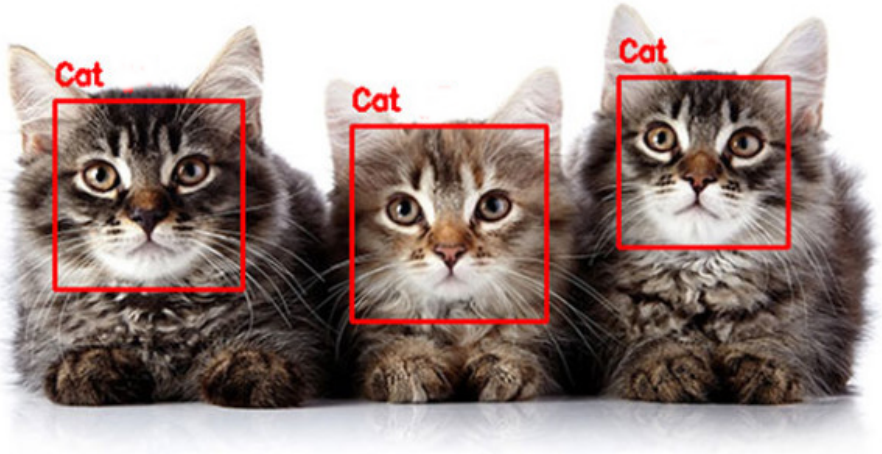
References and Credits

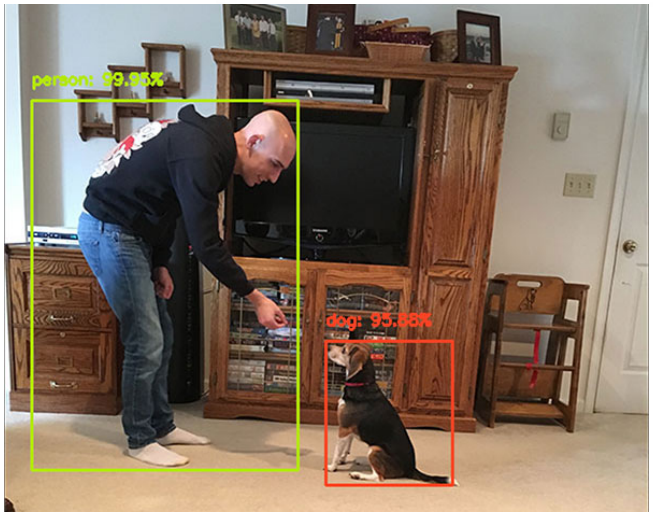
Some slides of this presentation adapted from

- KH Wong. “Ch. 6: Face detection”
- P. Viola and T.-W. Yue. “Adaboost for Face Detection”
- D. Miller. “Face Detection & Synthesis using 3D Models & OpenCV”
- S. Lazebnik. “Face detection”
- C. Schmid. “Category-level localization”
- C. Huang and F. Vahid. “Scalable Object Detection Accelerators on FPGAs
- Using Custom Design Space Exploration”
- P. Smyth. “Face Detection using the Viola-Jones Method”
- K. Palla and A. Kalaitzis. “Robust Real-time Face Detection”
- P. Viola and M. Jones. “Fast and Robust Classification using Asymmetric
- AdaBoost and a Detector Cascade”
- D. Hoiem. “Adaboost”
- H. Wen-Chung. “Introduction to AdaBoost”

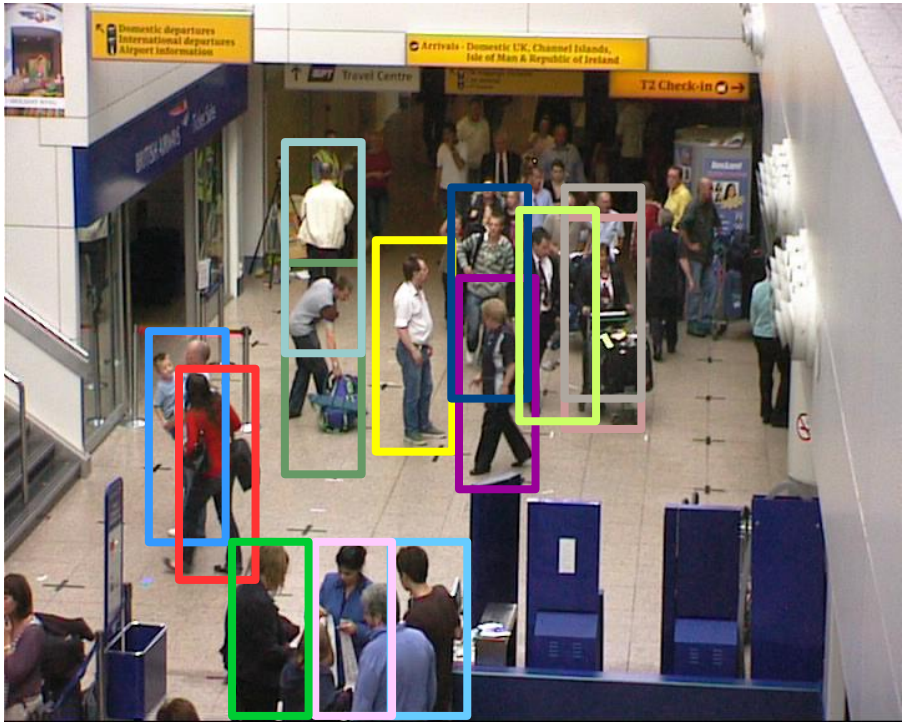
General Problem Description

Given an image, find regions that contain instances of specific objects



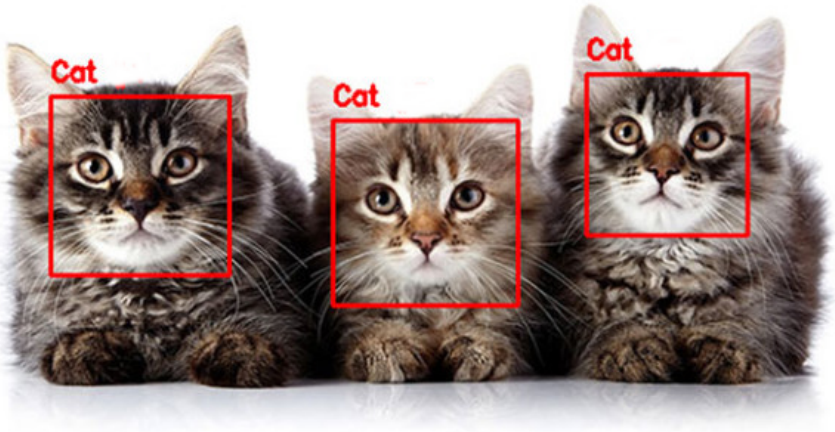


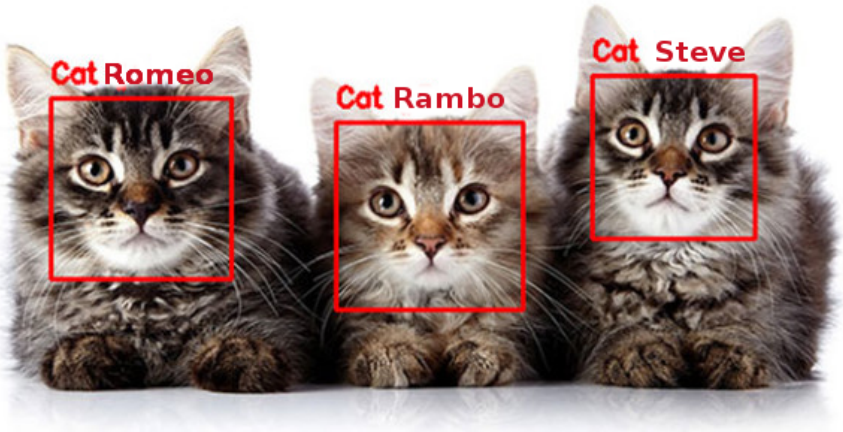
Detection VS Identification



Identification: you know the identifier of the object you detected over time

Detection VS Recognition

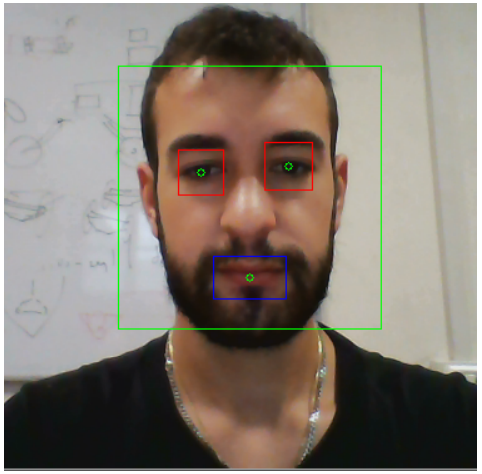


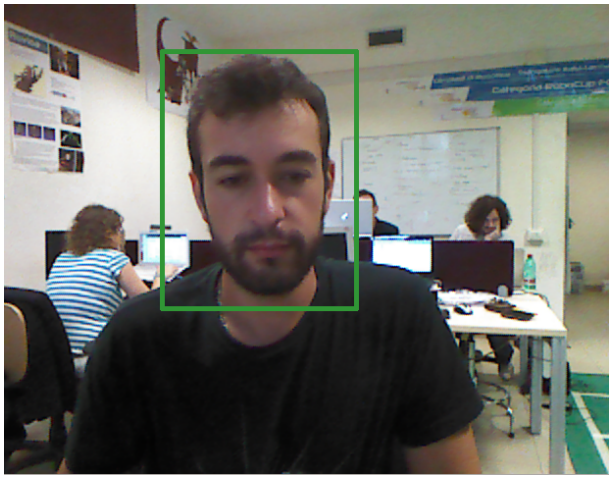


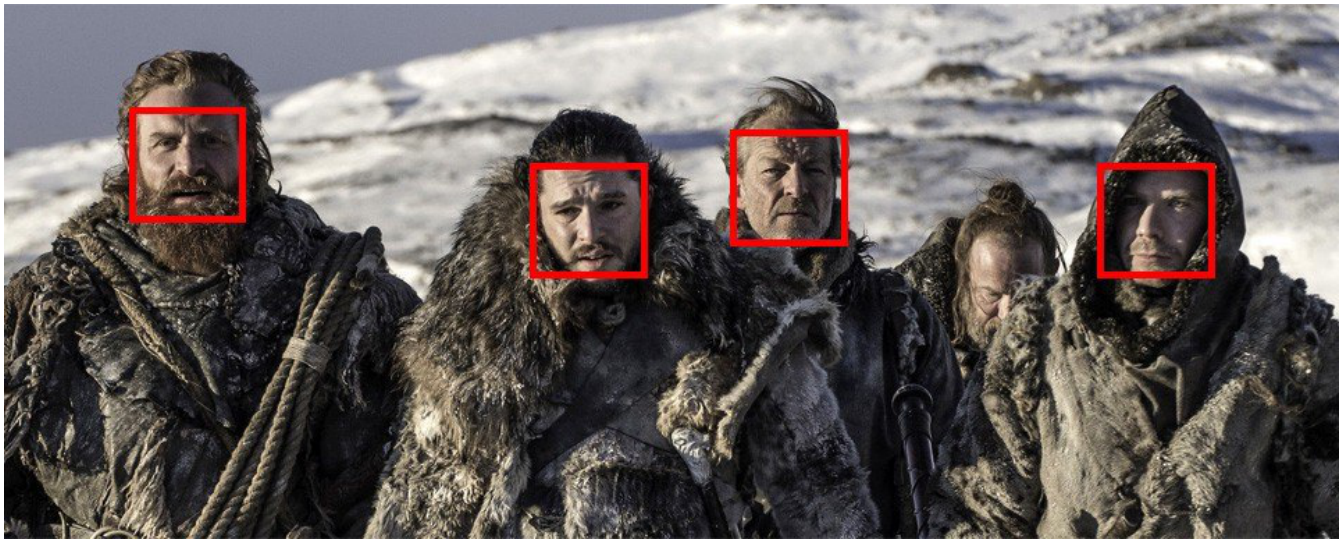
Recognition: you recognize who is who

Specific Problem Description

Given an image, find regions that contain instances of faces

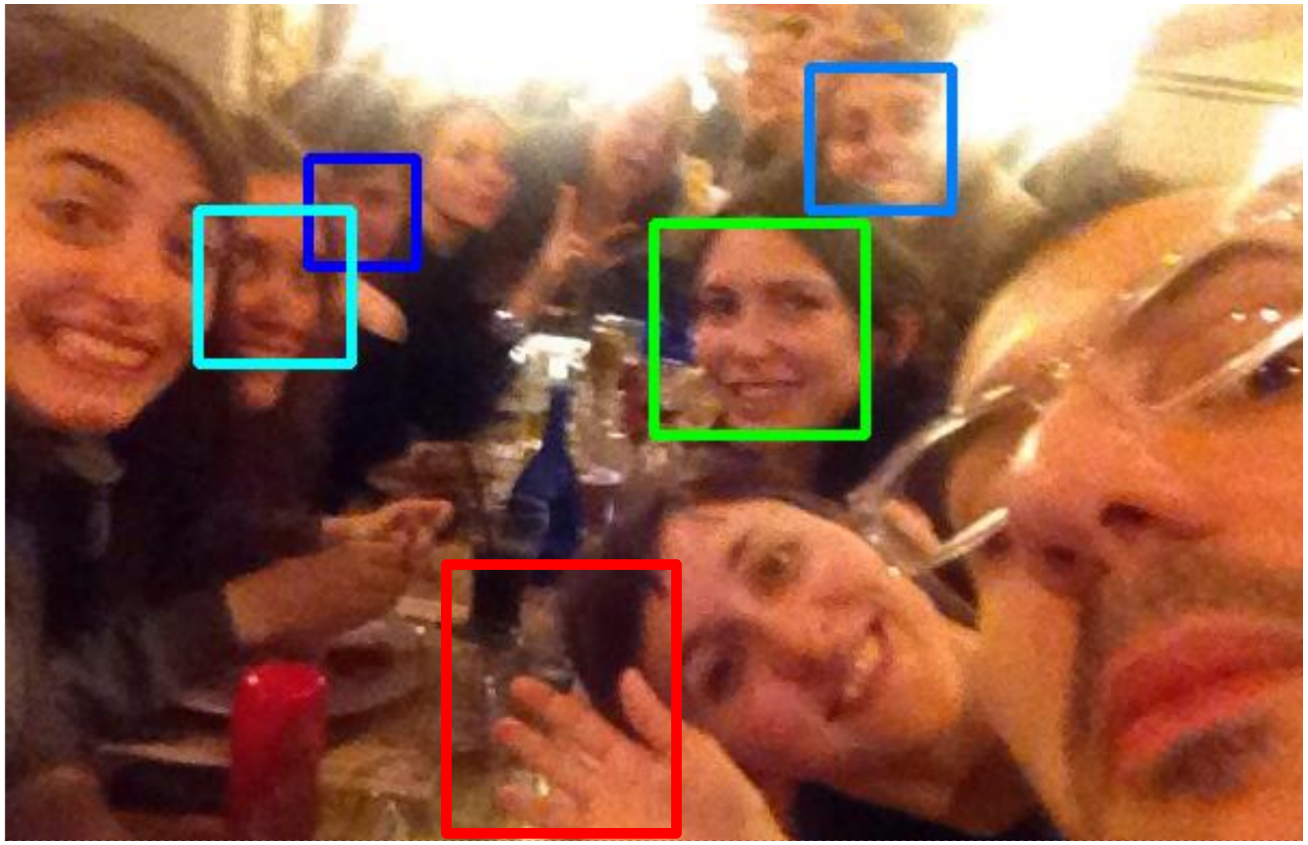




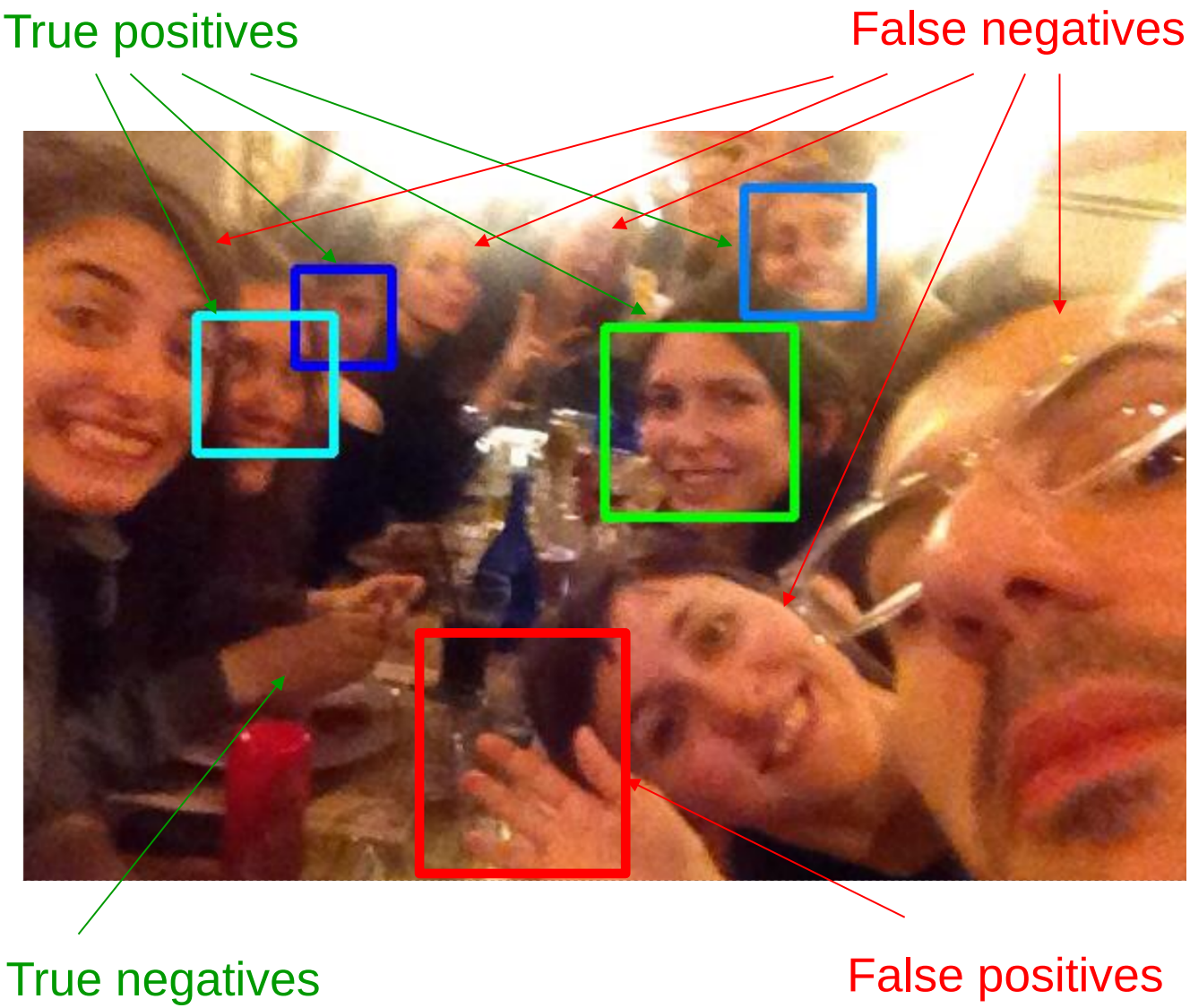


Challenges

- rotation
- blurring
- illumination
- occlusion
- glasses
- ...



Metrics


True positives
False negatives
True negatives
False positives

Basic Machine Learning Concepts

Idea: enable a machine to approximate (learn) a function given some data or experiences

- **supervised**
- unsupervised
- reinforcement learning

Supervised Learning

Dataset: set of data experienced and collected by somebody. Each datum in the dataset is composed by features and labels.

Example:

Features: days of rain	**average temperature**	**Label: season**
14	4°	Winter
12	6°	Winter
21	9°	Autumn
2	28°	Summer
5	17°	Spring

Classification VS Regression

Supervised learning typically deals with **regression** and **classification** problems

- regression: try to approximate a continuous function. You want to predict a continuous value.

  Example: given a person's job, age and years of experience predict salary

- classification: try to approximate a discrete function. You want to predict a discrete value (a category).

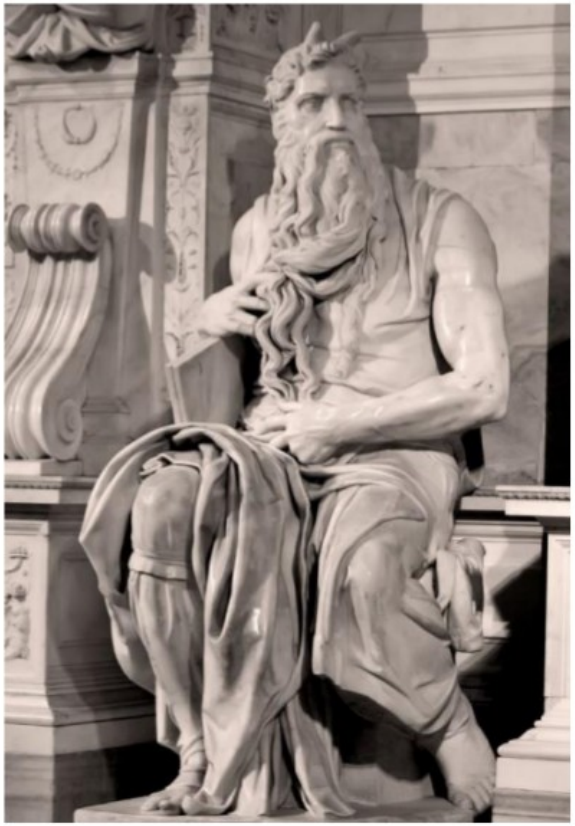
  Example: predict season given amount of rain and average temperature

Classification Based Detection

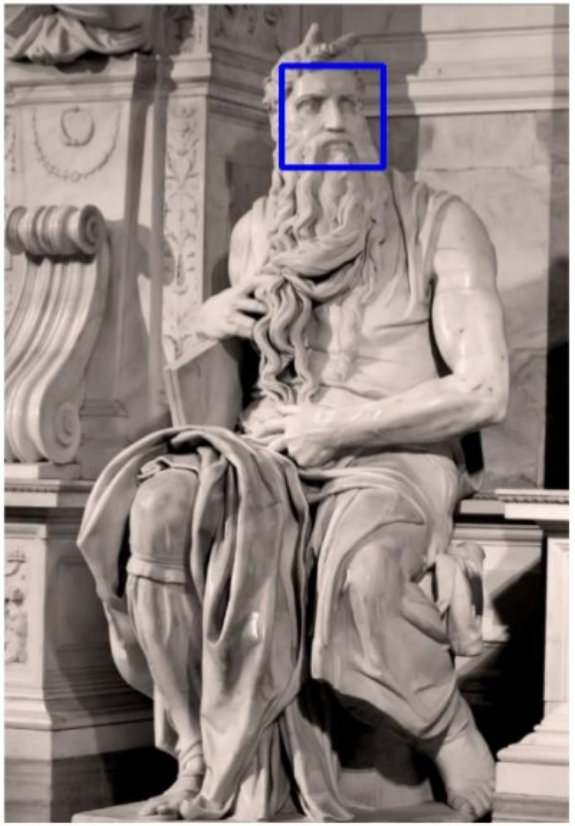
Input: image features (will talk more later)

Output: bounding box around instance(s) belonging to the class of objects for which the classifier has been trained

input image



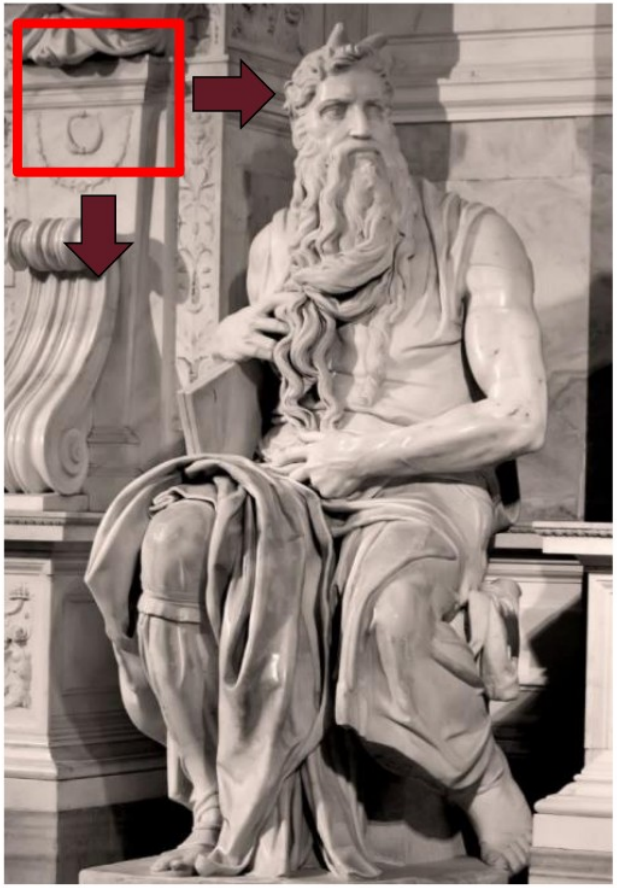
detection



Classification Based Detection: Idea

Slide a window of a specific size over the image and evaluate the current portion using a classifier at every location.

Note that most of the time the object is not there!



Problem: Multiple Scales

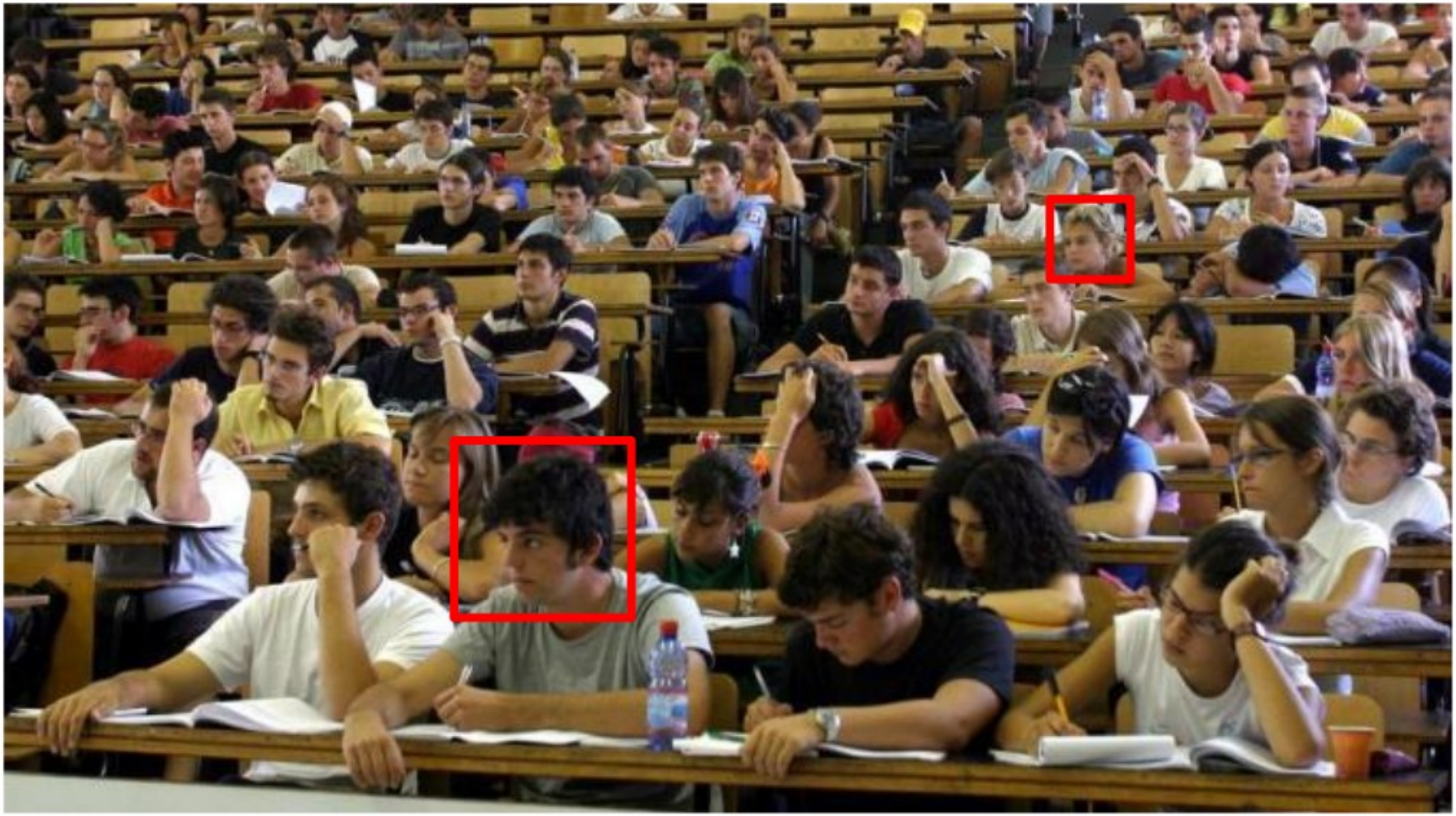


Image pyramids

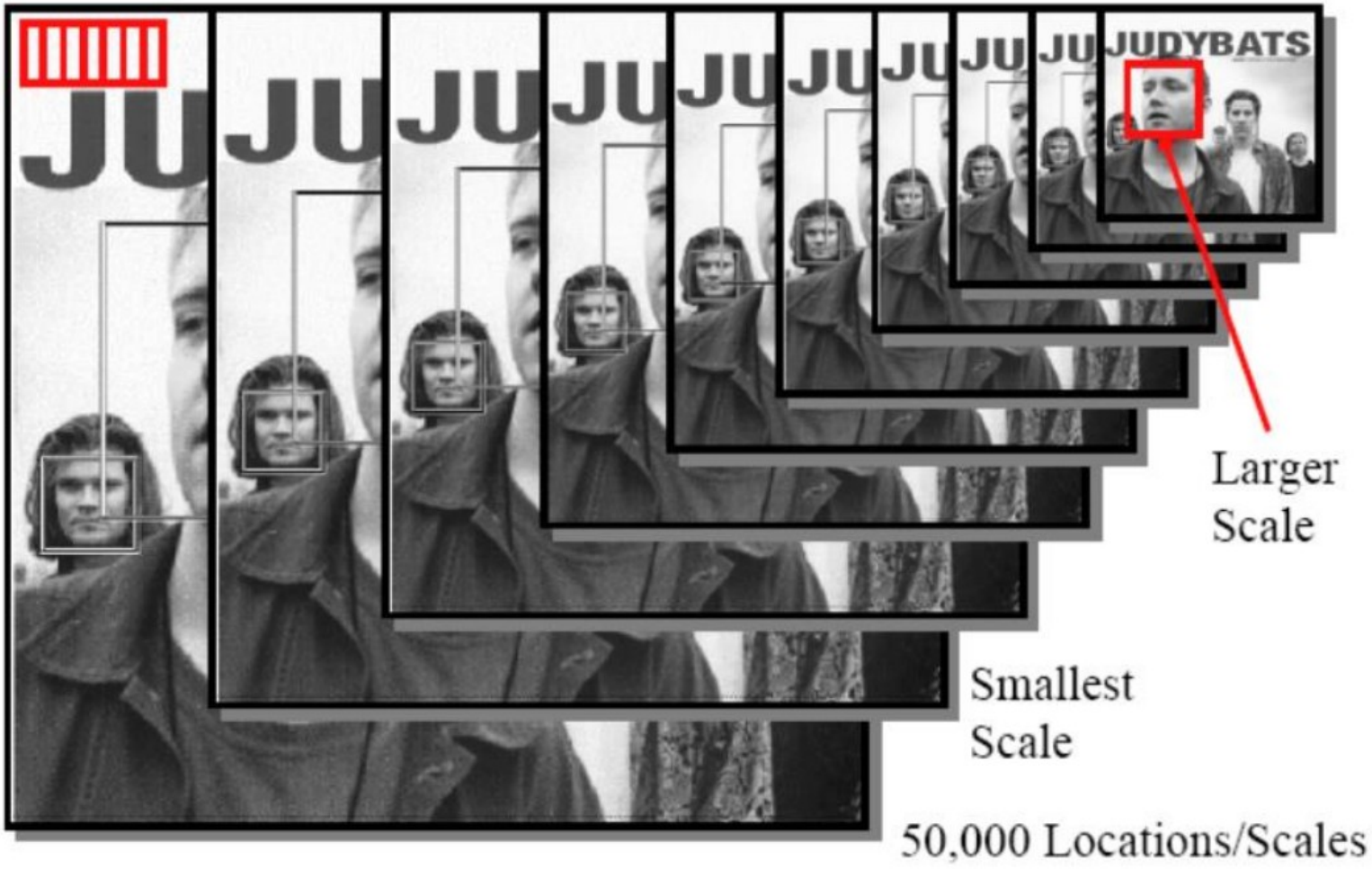
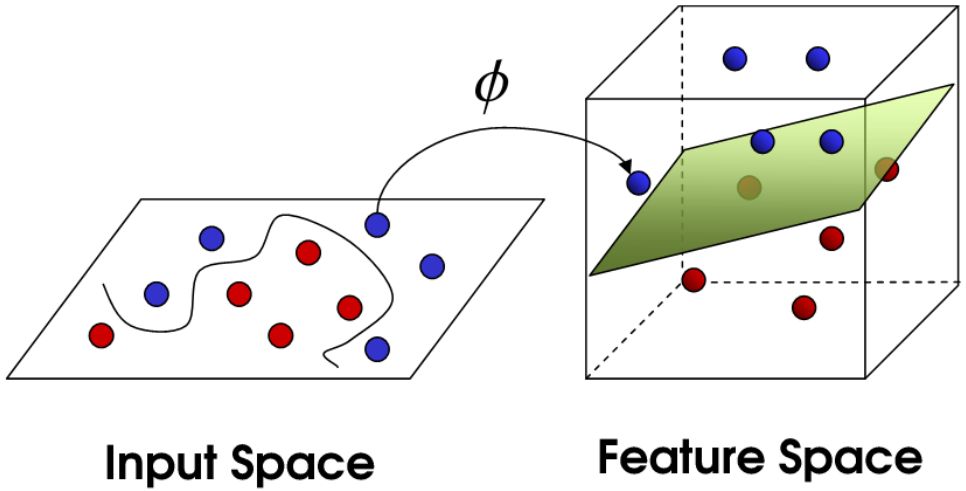


Image Features

- **computation:** features should be quick to compute
- **selection:** features should be highly discriminating
- **localization:** relevant features should focus only on potentially interesting areas of the image



Viola-Jones Method

One of the most famous approaches for face detection

- fast detection
- contributions:
  - use of integral images (computation)
  - boosting for face detection (selection)
  - attentional cascade for fast rejection of non-face areas (localization)

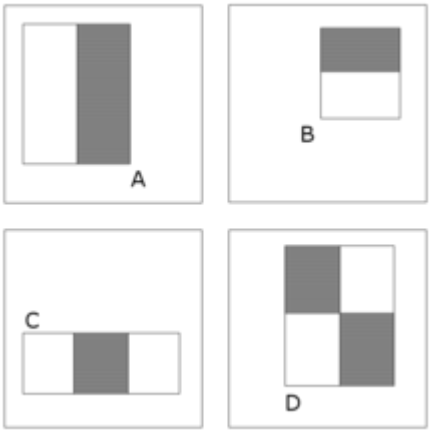
[1] P. A. Viola, M. J. Jones. Rapid object detection using a boosted cascade of simple features, CVPR, pp.511-518, 2001

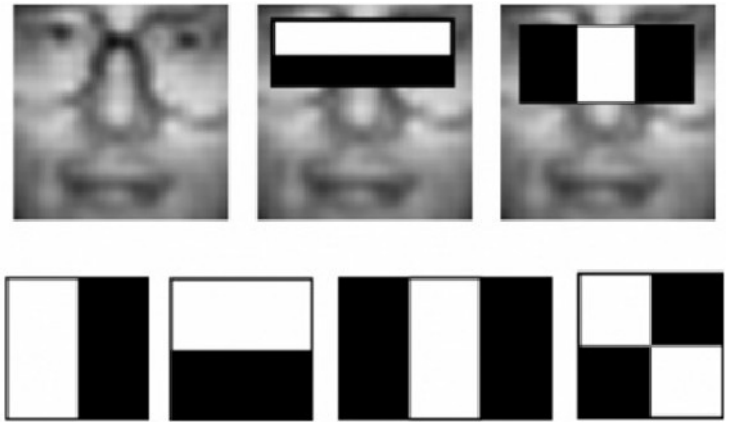
[2] P. A. Viola, M. J. Jones. Robust Real-Time Face Detection. IJCV 57(2), pp. 137-154, 2004

Viola-Jones Features

Four type of rectangle features:

- easy to compute (faster with integral images)
- white areas are subtracted from black areas

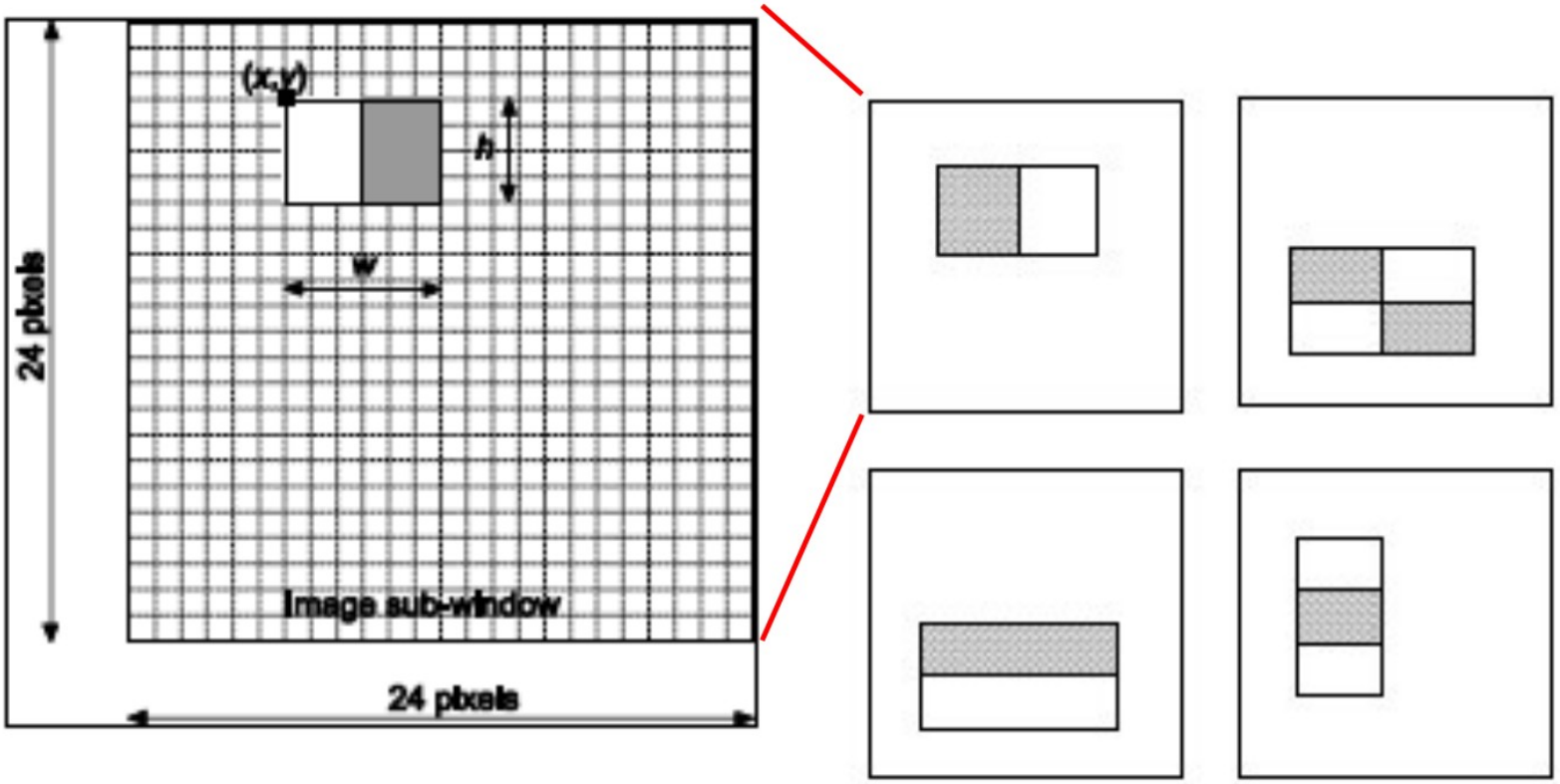




Viola-Jones Feature Computation

Slide rectangles over the sliding window:

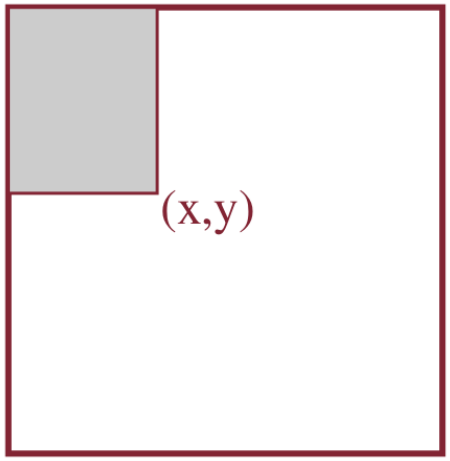
Value = Σ (pixels in black area) - Σ (pixels in white area)



Rectangle Feature Efficiency

The motivation behind using rectangular features, as opposed to more expressive steerable filters, is due to their extreme computational efficiency.

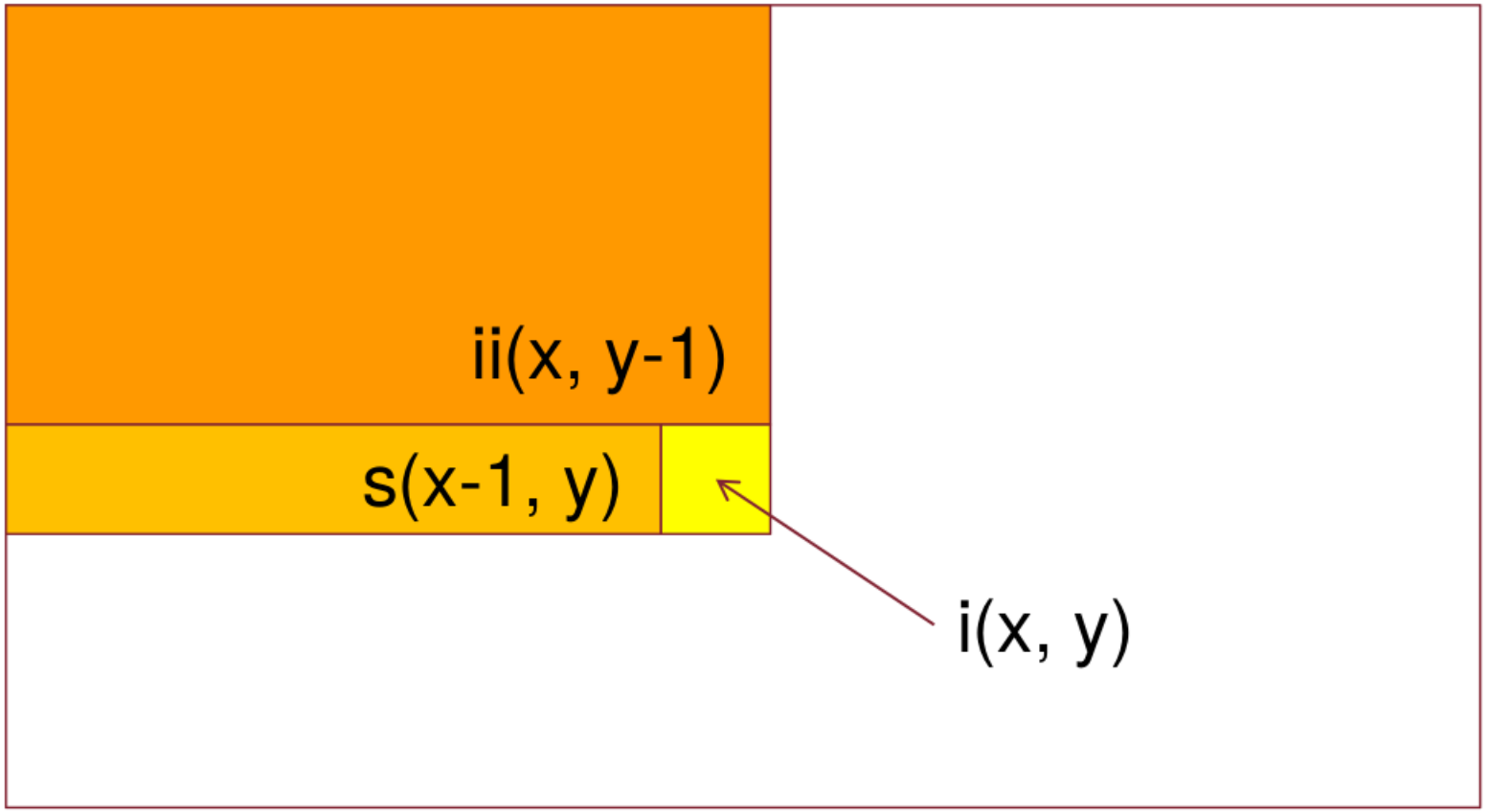
Efficiency can be boosted using integral images.



Integral image: an intermediate representation of the image for rapid calculation of rectangle features.

Integral Image

- Cumulative row sum: s(x, y) = s(x–1, y) + i(x, y)
- Integral image: ii(x, y) = ii(x, y−1) + s(x, y)



Integral Image: Example

IMAGE

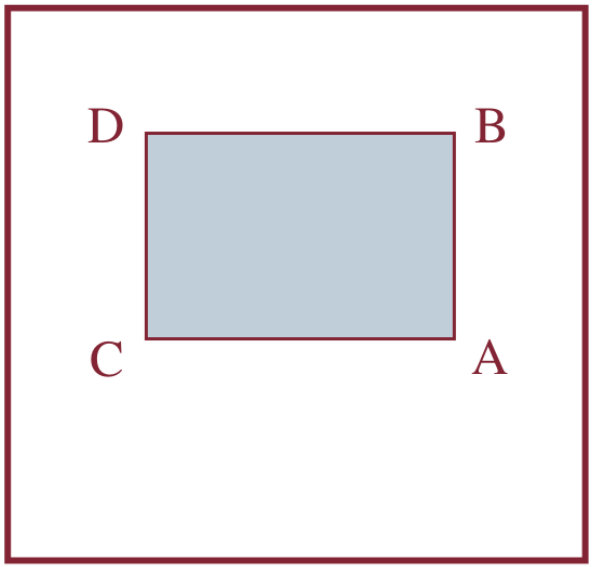
0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

INTEGRAL IMAGE

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Compute Rectangle Sum on Integral Images

- A, B, C, D: values of the integral image at the corners of a rectangle
- sum of original image values within the rectangle can be computed as: A – B – C + D
- only 3 additions are required for any size of rectangle



Feature Selection

- for a 24x24 detection region, the number of possible rectangle features is ~160,000!
- at test time, it is impractical to evaluate the entire feature set

Relevant feature

Irrelevant feature

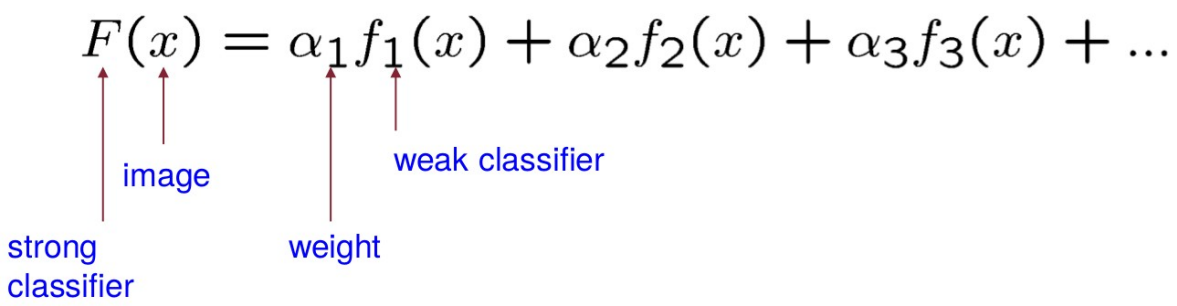
- how do we select only relevant features?

Boosting (Schapire 1989)

- randomly select $n_1 < n$ samples from D (dataset) to obtain D_1
    - train weak learner C_1
- select $n_2 < n$ samples from D with half of the samples misclassified by C_1 to obtain D_2
    - train weak learner C_2
- select all samples from D that C_1 and C_2 disagree on
    - train weak learner C_3
- final classifier is vote of weak learners

AdaBoost (Adaptive Boosting)

- instead of sampling from dataset, re-weight classifiers
- final classification is based on weighted vote
- strong classifiers are built as a linear combination of simple weak classifiers



AdaBoost (Adaptive Boosting)

- AdaBoost is an algorithm for constructing a ”strong” classifier as linear combination

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

of “simple” “weak” classifiers $h_t(x)$.

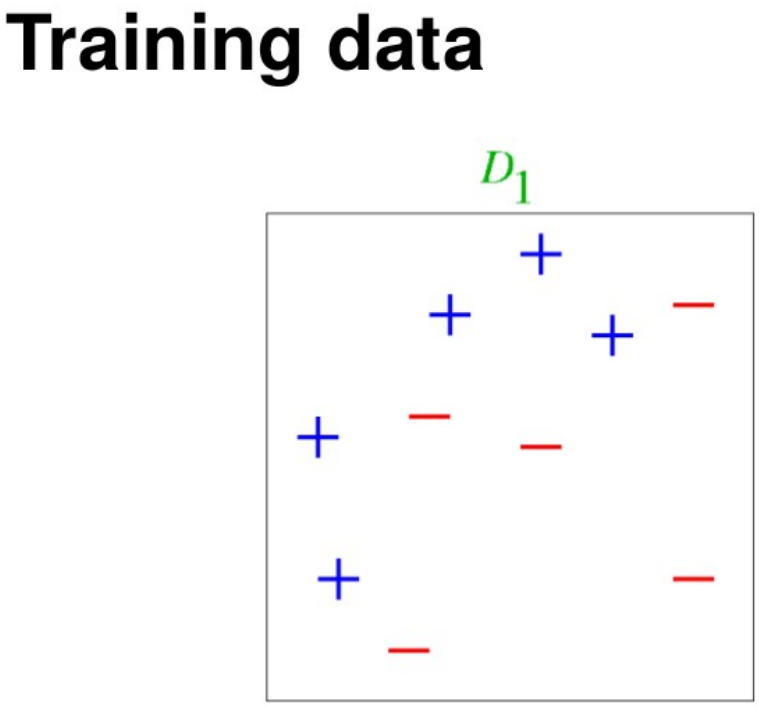
Terminology

- $h_t(x)$ ... “weak” or basis classifier, hypothesis, ”feature”
- $H(x) = \mathit{sign}(f(x))$ ... ‘’strong” or final classifier/hypothesis

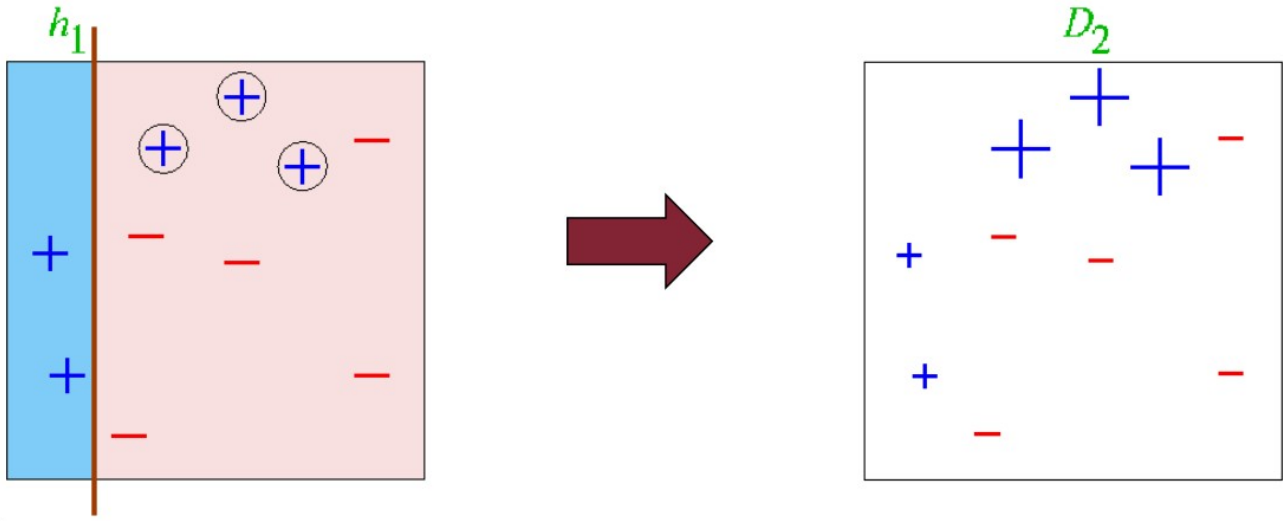
Comments

- The $h_t(x)$’s can be thought of as features.
- Often (typically) the set $\mathcal{H} = \{h(x)\}$ is infinite.

Boosting Example: Dataset



Boosting Example: 1st Weak Classifier



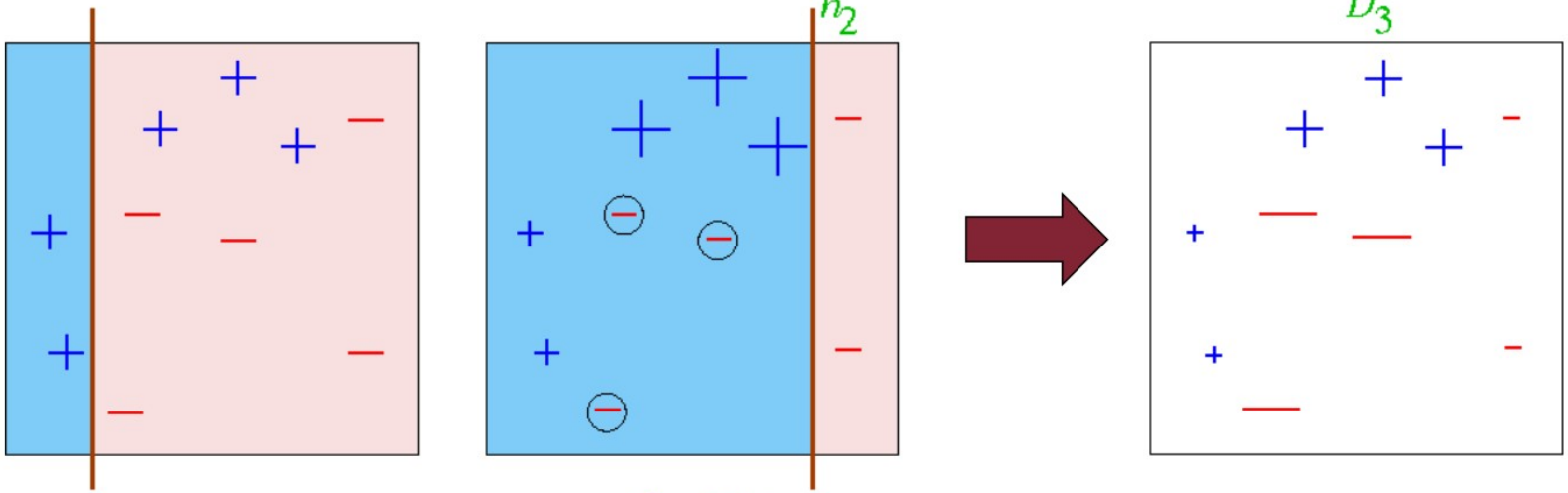
1st weak classifier

$\varepsilon_1 = 0.30$

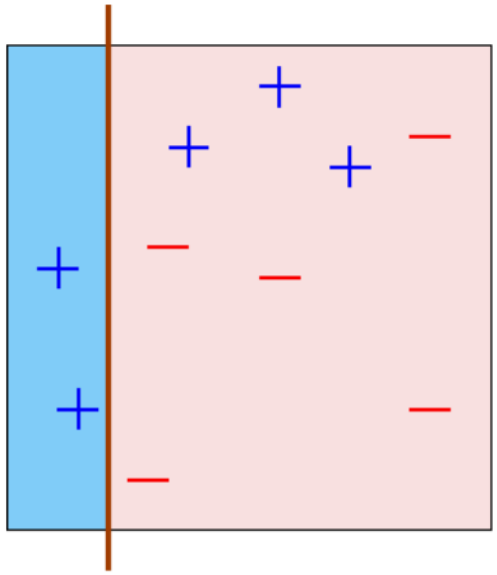
$\alpha_1 = 0.42$

Error

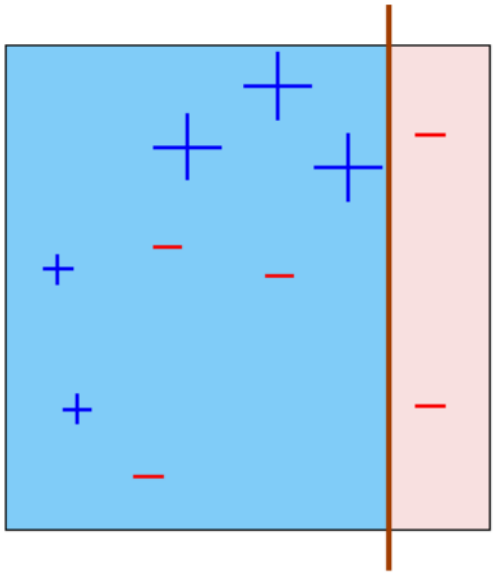
Boosting Example: 2nd Weak Classifier



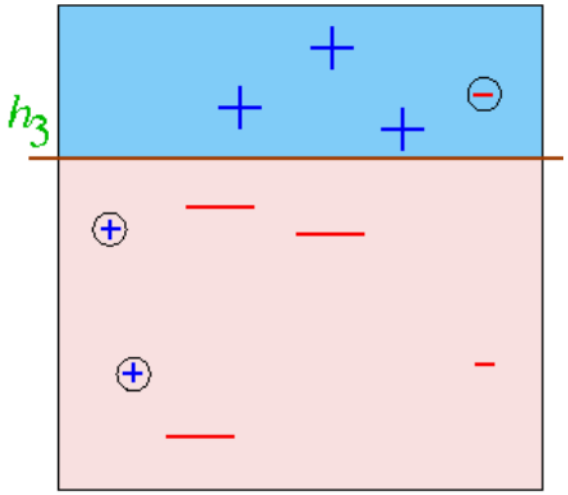
Boosting Example: 3rd Weak Classifier



1st weak classifier



2nd weak classifier

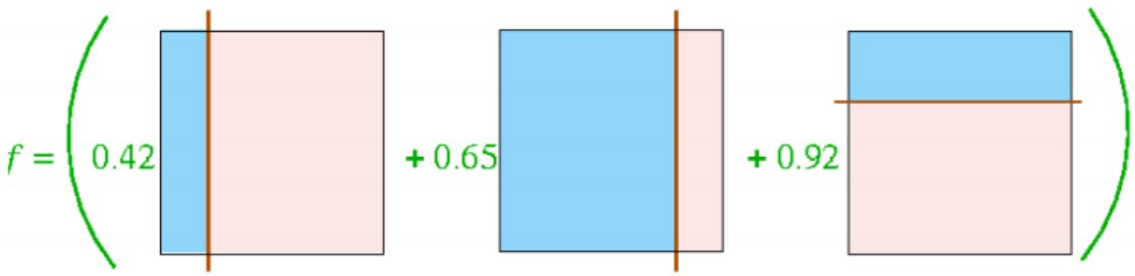


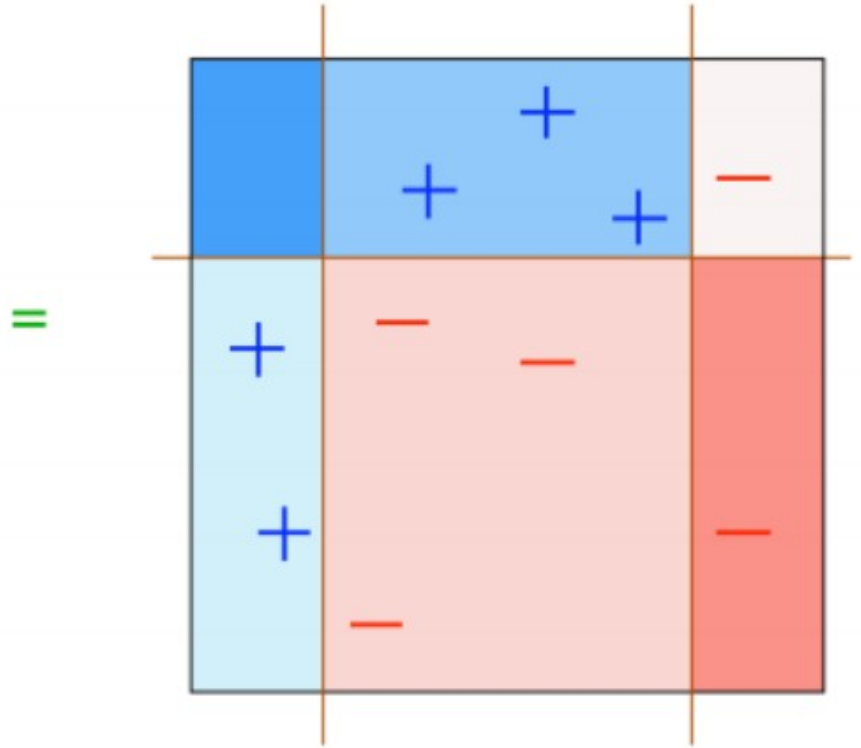
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

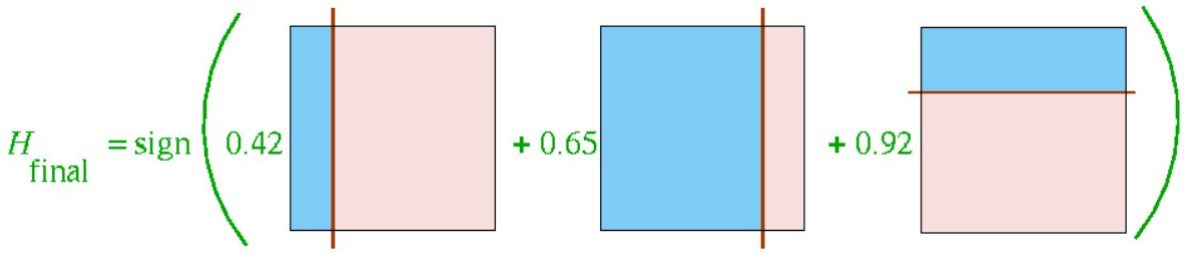
3rd weak classifier

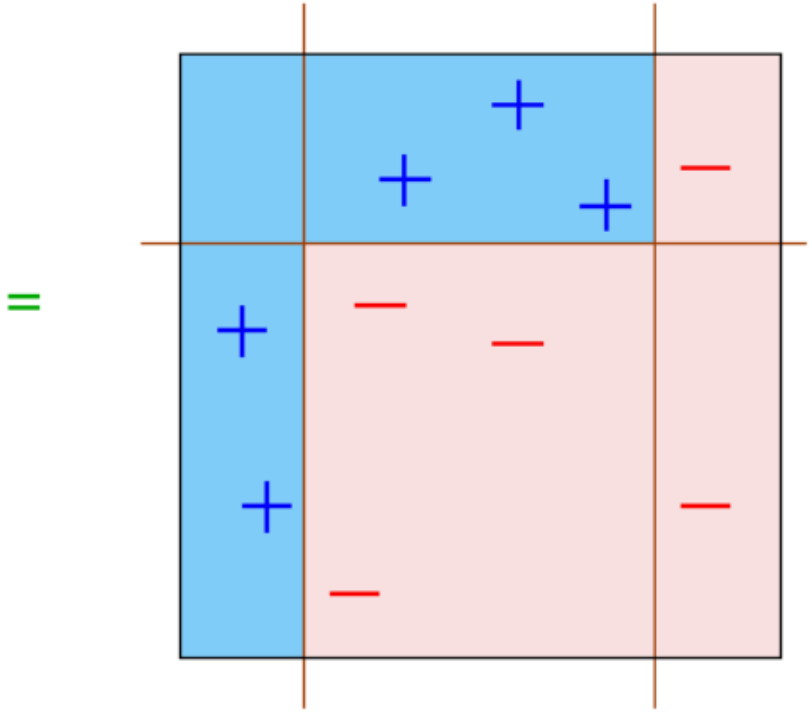
Boosting Example: Weighted Combination





Boosting Example: Final Classifier





AdaBoost Algorithm

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$$Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

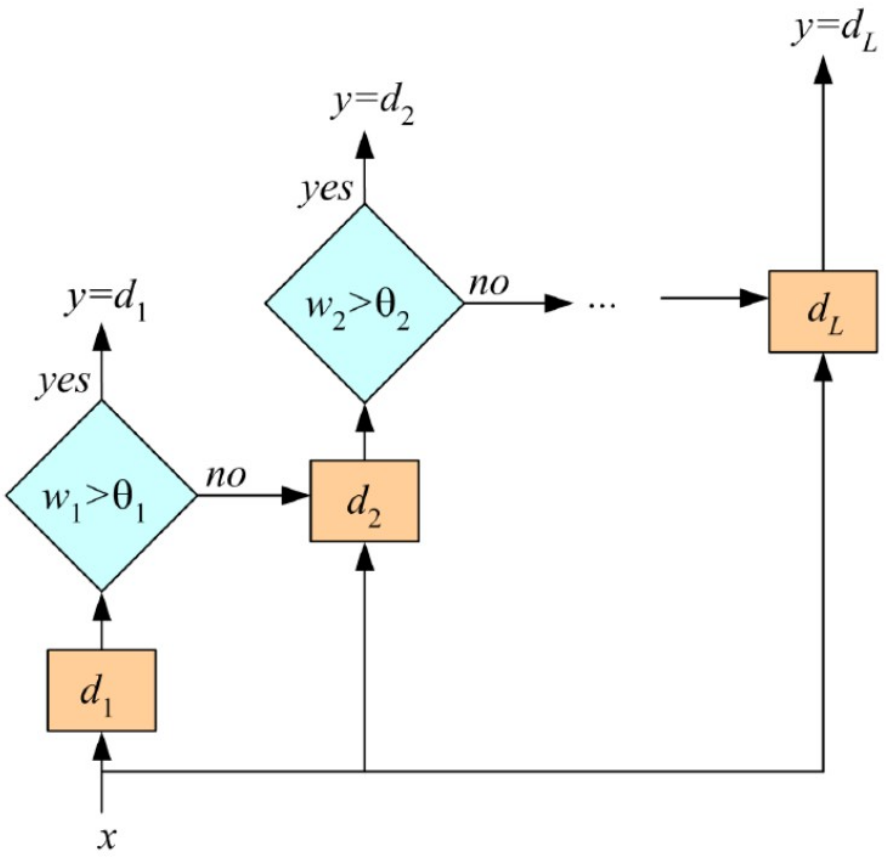
Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

Cascading Classifiers

Use d_j only if preceding ones are not confident

Cascade learners in order of complexity



AdaBoost Computational Complexity

- define weak learners based on rectangle features
- for each round of boosting:
  - evaluate each rectangle filter on each example
  - select best threshold for each filter
  - select best filter/threshold combination
  - re-weight examples
- computational complexity of learning: O(MNK)
  - M rounds, N examples, K features

Training

- 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 Ada-Boost 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

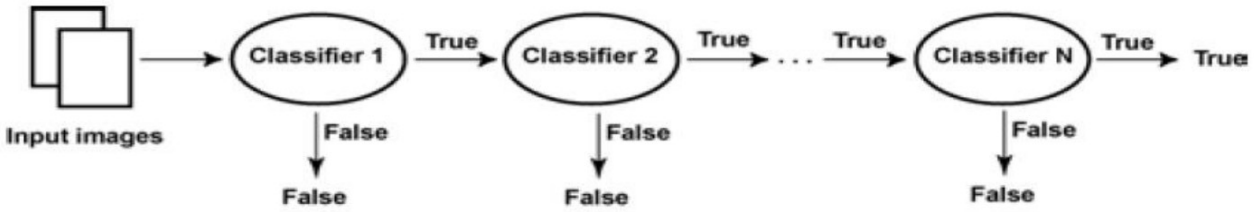
Training Data

- 5000 faces
  - all frontal
  - rescaled to 24x24 pixels
- 300 million non-faces
- faces are normalized wrt:
  - scale
  - translation
- variations wrt:
  - individuals
  - illumination
  - pose

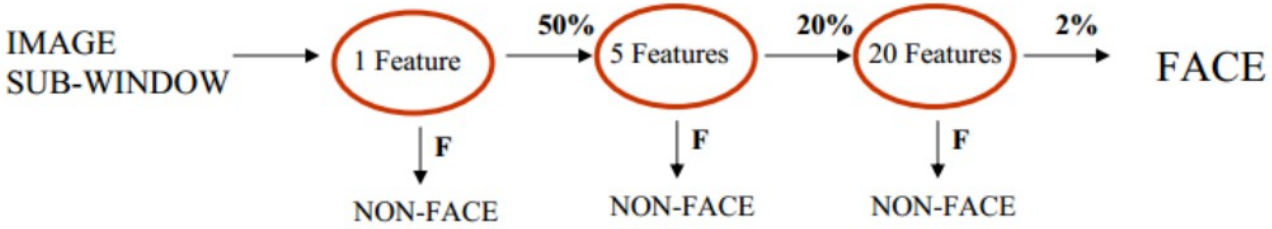
AdaBoost cascade

A chain of classifiers that each reject some fraction of the negative training samples while keeping almost all positive ones.

Each classifier is an AdaBoost ensemble of rectangular features sampled from a large pool.



Cascaded Classifier



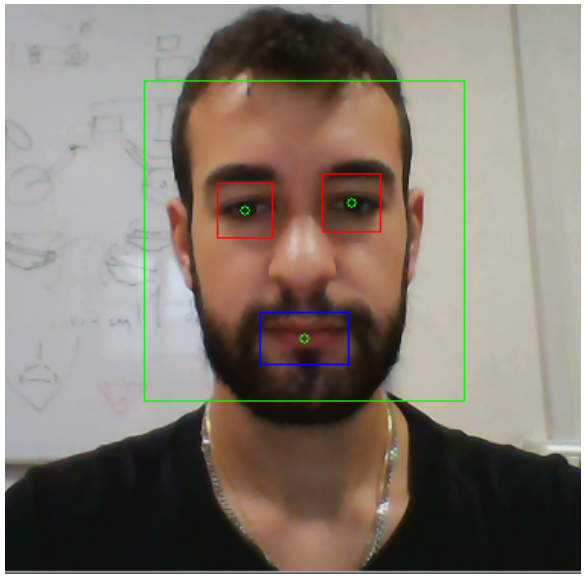
- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive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

Limitation of AdaBoost

- AdaBoost minimizes a quantity related to classification error and it does not minimize the number of false negatives.
- feature selection proceeds as if classification error was the only goal, and the features selected are not optimal for the task of rejecting negative examples

Face Detection in OpenCV

- OpenCV comes with a trainer as well as a detector
- OpenCV already contains many pre-trained classifiers for face, eyes, smile etc.



https://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html

OpenCV: Loading HAAR Features

```
#include <opencv2/opencv.hpp>
#include <opencv2/imgproc/imgproc.hpp>

cv::CascadeClassifier frontal_face_cascade;
cv::CascadeClassifier profile_face_cascade;

if(!frontal_face_cascade.load(frontalFaceCascadeFilename) ||
   !profile_face_cascade.load(profileFaceCascadeFilename)) {
    std::cerr << "Error while loading HAAR cascades." << std::endl;
    return -1;
}
```

Search feature in image:

```
frontal_face_cascade.detectMultiScale(grayImage, frontal_face_vector, 1.4, 4,
                                      0|CV_HAAR_SCALE_IMAGE, cv::Size(50, 50));
profile_face_cascade.detectMultiScale(grayImage, profile_face_vector, 1.4, 4,
                                      0|CV_HAAR_SCALE_IMAGE, cv::Size(50, 50));
```

Homework

- write a ROS node that reads an image from your webcam. The node should:
  - implement face detection using HAAR-like features on the whole image both for frontal and profile faces
  - select and show/store the most visible face
    - i.e., the face with the biggest area

(use http://wiki.ros.org/image_transport/Tutorials/PublishingImages#Adding_video_stream_from_a_webcam as a starting point)