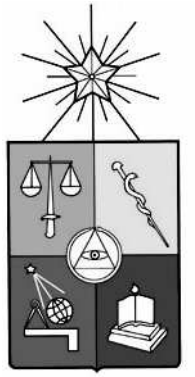


Universidad de Chile
Department of Electrical Engineering

Object detection using cascades of boosted classifiers

Javier Ruiz-del-Solar and Rodrigo Verschae

EVIC 2006
December 15th, 2006
Chile

General Outline

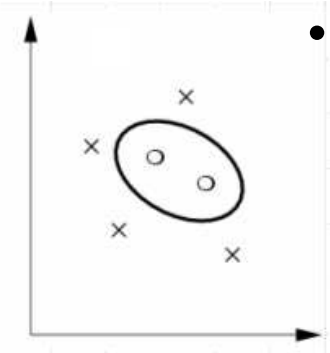
- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Applications to face analysis problems

General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Applications to face analysis problems

The 2-class Classification Problem

- Definition:
  - Classification of patterns or samples in 2 a priori known class. One class can be defined as the negation of the other class (detection problem).
- Examples:
  - Face detection, tumor detection, hand detection, biometric identity verification (hand, face, iris, fingerprint, …), fault detection, skin detection, person detection, car detection, eye detection, object recognition, …
- Face Detection as an exemplar difficult case
  - High dimensionality (20x20 pixels → $256^{400} = 2^{3200}$ possible combinations)
  - Many possible different faces (6408 10^6 habitants ≈ $1.5*2^{32}$)
  - Differences in race, pose, rotation, illumination, …

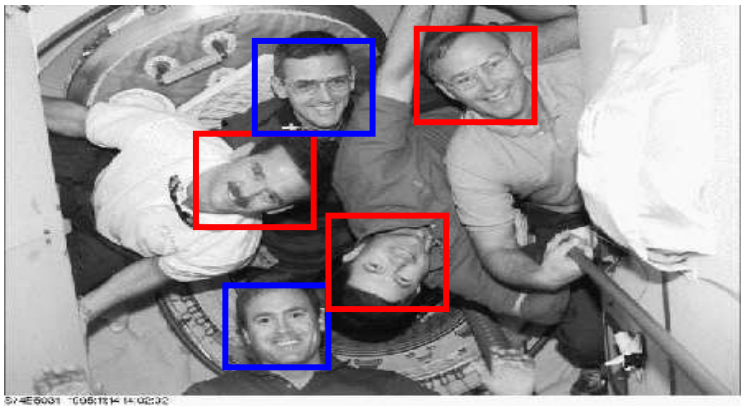


What is object detection?

- **Definition**:
  - Given a arbitrary image, to find out the position and scale of all objects (of a given class) in the images, if there is any.
- **Examples**:

Views (Poses)

In some cases objects observed under different views are considered as different objects.



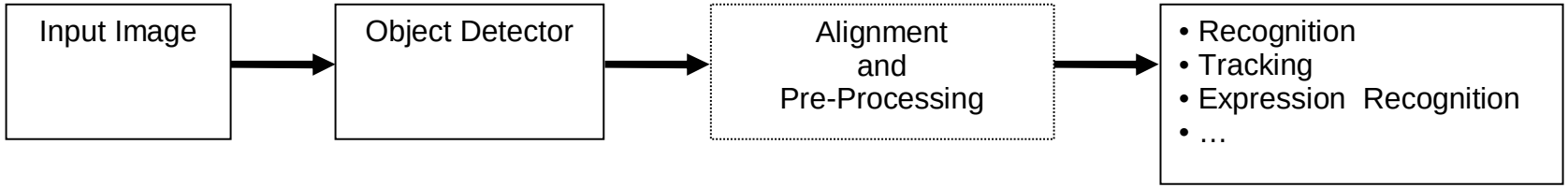
Frontal

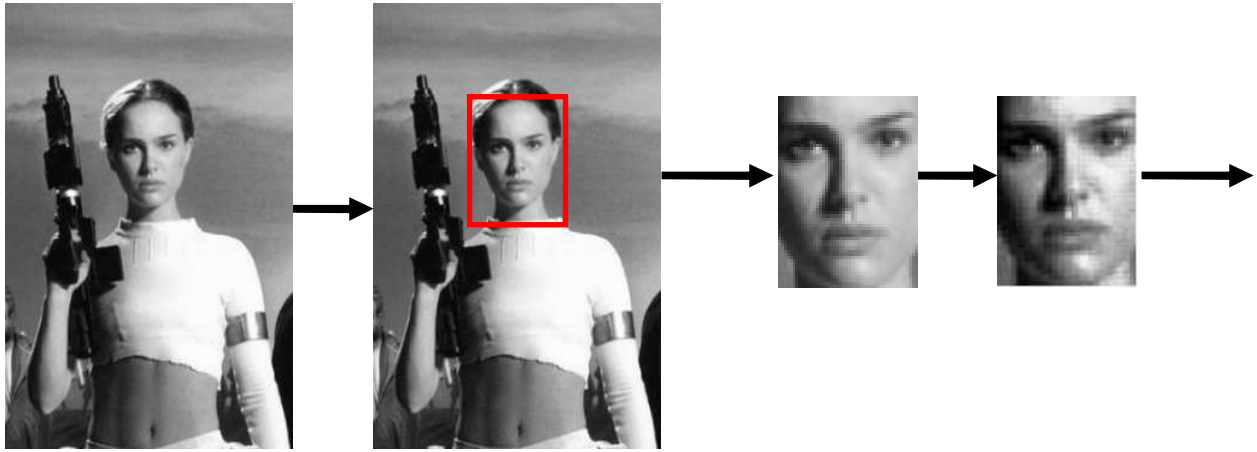
Semi-Frontal

Profile

Applications

- A object detector is the first module needed for any application that uses information about that kind of object.





Challenges (1)

Why it is difficult to detect objects?

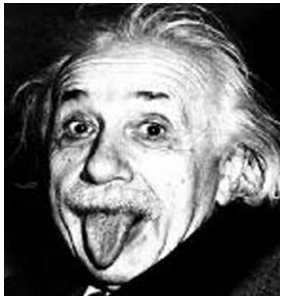
- Reliable operation in real-time, real-world.
- Problems:
  - intrinsic variability in the objects
  - extrinsic variability in images.
- Some faces which are difficult to detect are shown in red

Challenges (2)

- Intrinsic variability:

Presence or absence of structural components

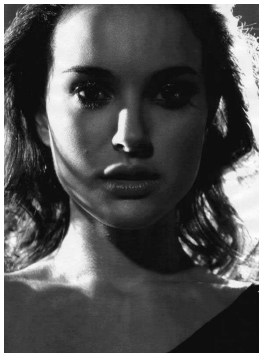
Variability among objects



Variability of the particular object

Challenges (3)

- Extrinsic variability in images:



Illumination

Out-of-Plane Rotation (Pose)

Occlusion

In-plane Rotation

Scale

- Capturing Device / Compression / Image Quality/ Resolution

Challenges (4)

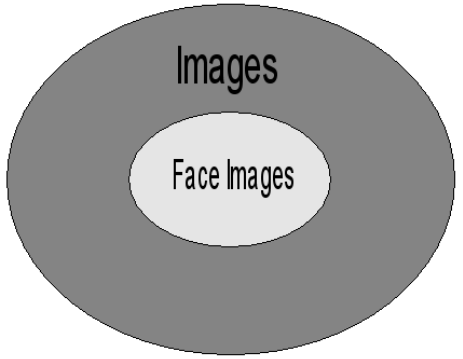
- Why gray scale images?
  - Some images are just in grey scale and in others the colors were modified.
  - The color changes according to the illumination conditions, capturing device, etc
  - The background can have similar colors
  - Using the state of the art segmentation algorithms, it is only possible to obtain very good results when the working environment is controlled.
  - However, color is very useful to reduce the search space, though some objects may be lost.
  - In summary, under uncontrolled environments, it is even more difficult to detect objets if color is used.

General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Applications to face analysis problems

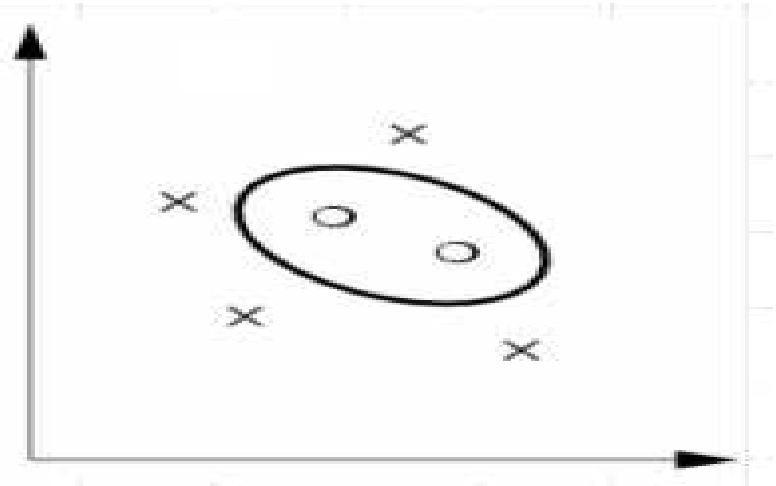
State of the art

- Statistical learning based methods:
  - SVM (Support Vector Machines, Osuna et al. 1997) *
  - NN (Neural Networks)
    - Rowley et al. 1996; Rowley et al. 1998 (Rotation invariant)
  - Wavelet-Bayesian (Schneiderman & Kanade 1998, 2000) *
  - SNoW (Sparse Network of Winnows, Roth et al.1998) *
  - FLD (Fisher lineal Discriminant, Yang et al. 2000)
  - MFA (Mixture of Factor Analyzers, Yang et al. 2000)
  - Adaboost/Nested-Cascade*
    - Viola & Jones 2001 (Original work), 2002 (Asymmetrical), 2003 (Multiview); Bo WU et al. 2004 (Rotation invariant, multiview); Fröba et al. 2004 (Robust to extreme illumination conditions); Yen-Yu Lin et al. 2004 (occlusions)
  - Kullback-Leibler boosting (Liu & Shum, 2003)
  - CFF (Convolutional Face Finder, neural based, Garcia & Delakis, 2004)
  - Many Others…
- Best Reported Performance:
  - Adaboost/Nested-Cascade *
  - Wavelet-Bayesian *
  - CFF
  - Kullback-Leibler boosting



Statistical Classification Paradigm

- Set of training examples $S = \{x_i, y_i\}_{i=1...m}$
- We estimate *f()* using $S = \{x_i, y_i\}_{i=1...m}$
  - The set *S*,the training set, is used to learn a function *f(x)* that predicts the value of *y* from *x.*
- *S* is supposed to be sampled i.i.d from an unknown probability distribution *P*.
- The goal is to find a function *f()*, a classifier, such that $\text{Prob}_{(x,y)\sim P}[f(x)!=y]$ is small.

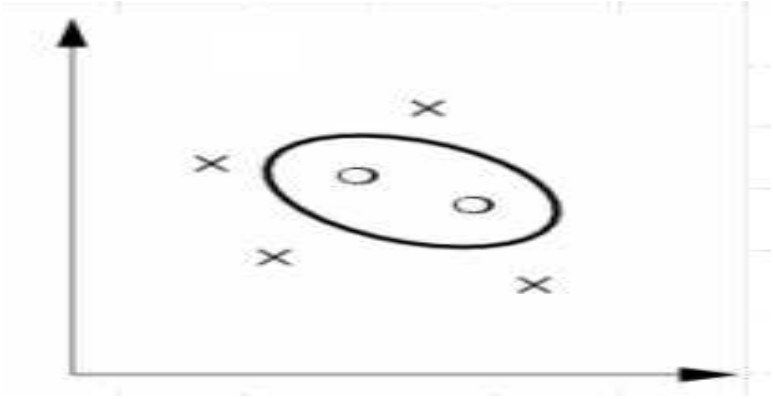


Statistical Classification Paradigm

- *Training Error(f)* = $\text{Prob}_{(x,y)\sim S}[f(x)!=y]$

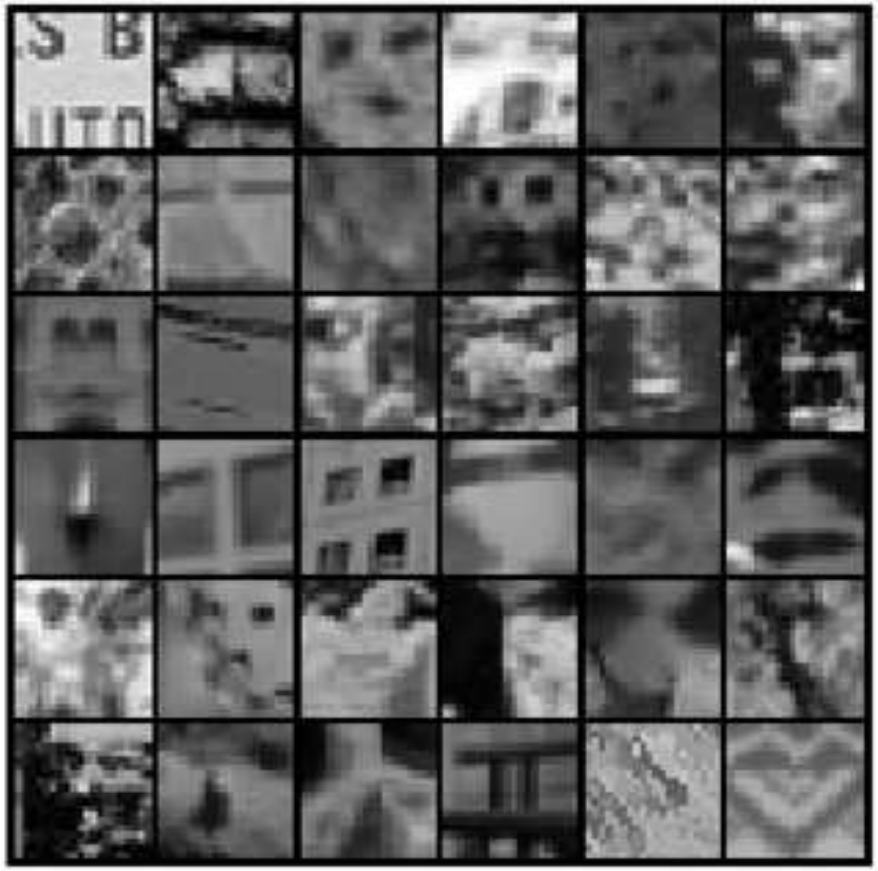
  = probability of incorrectly classifying an *x* coming from the training set
- *Test Error(f)* = $\text{Prob}_{(x,y)\sim P}[f(x)!=y]$

  = Generalization error
- We are interested on minimizing the Test Error, i.e., minimizing the probability of wrongly classifying a new, unseen sample.



Training Sets

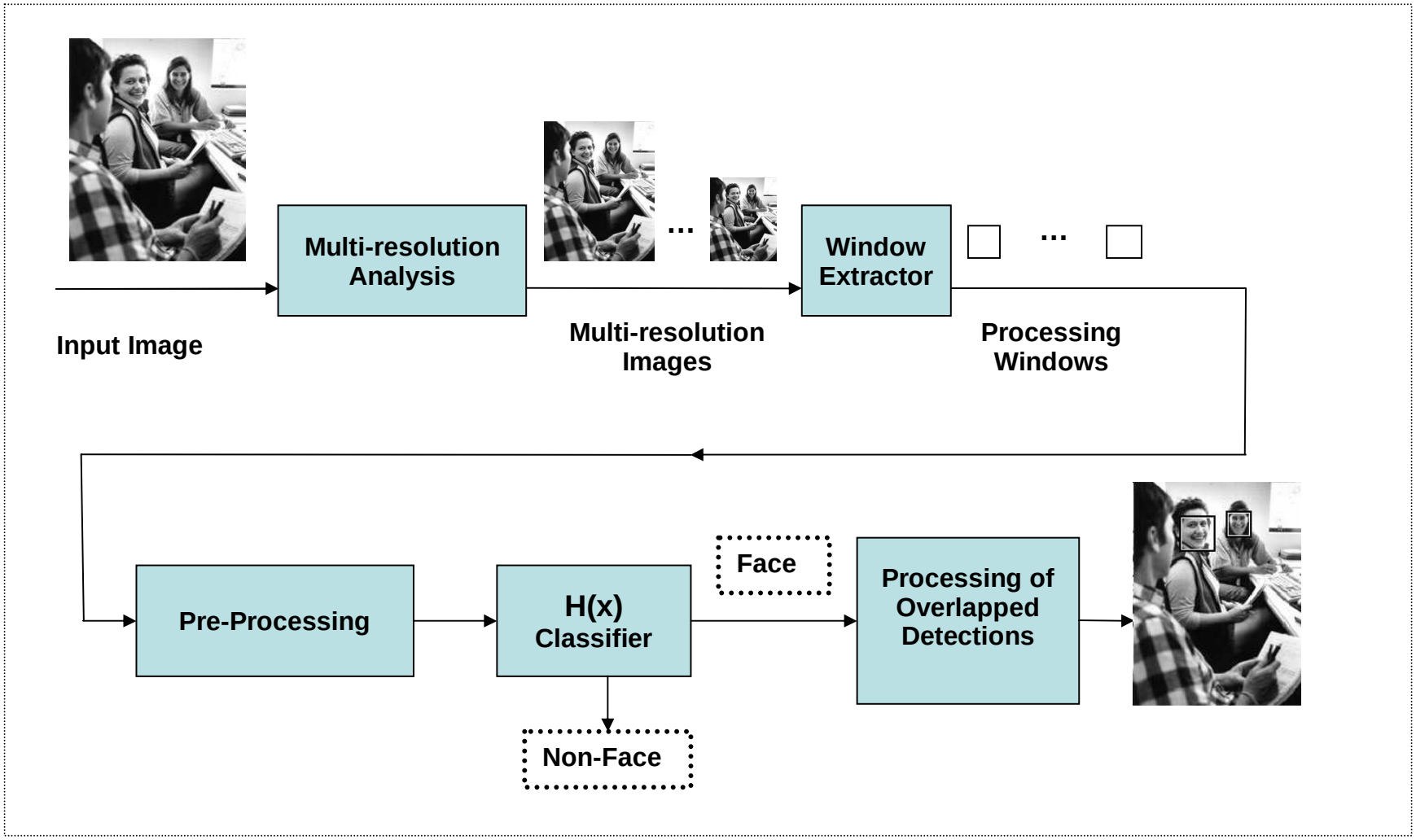
Faces



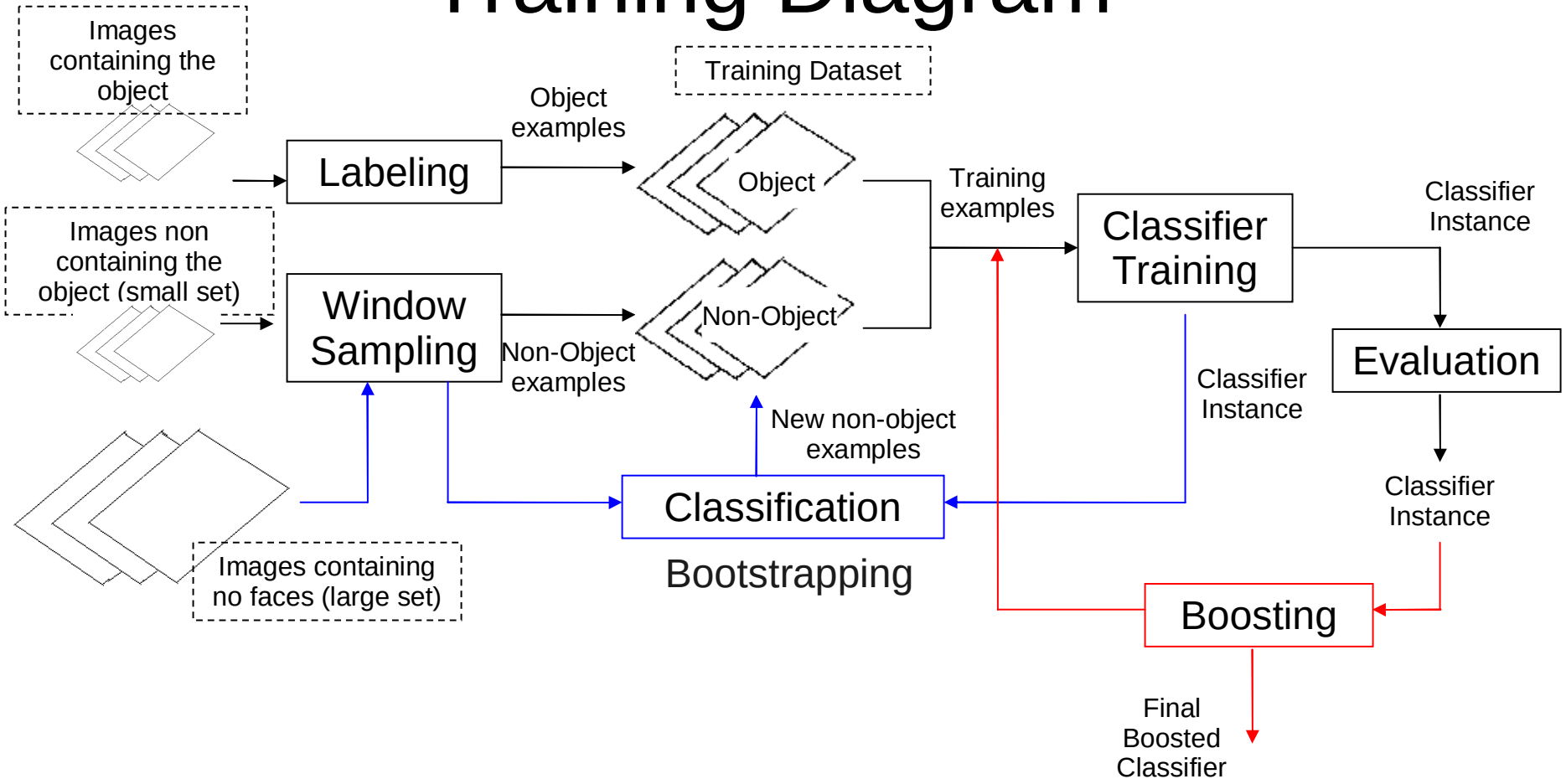
Non-Faces

[Images from: Ce Liu & Hueng-Yeung Shum, 2003]

Standard Multiscale Detection Architecture



Training Diagram



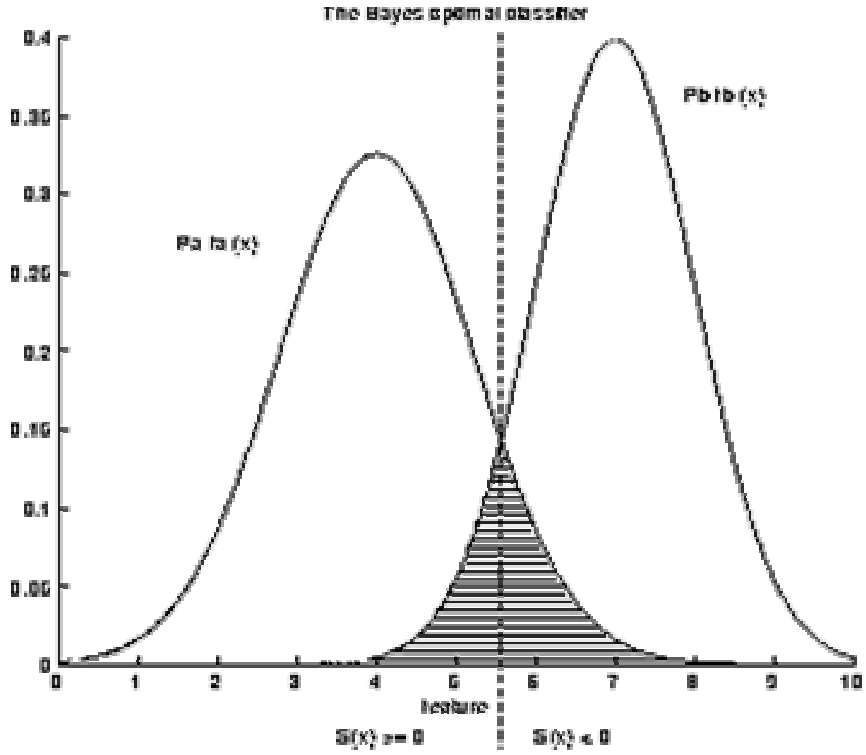
Bayes Classifiers

- Bayes Classifier

$$\frac{P(x/Object)}{P(x/nonObject)}$$

- Naive

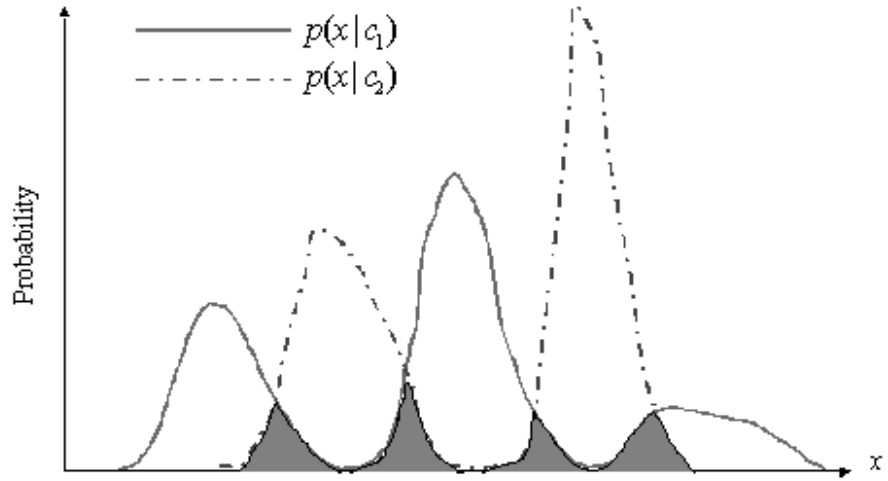
$$\prod_{i=1}^{K} \frac{P(F_i(x)/Object)}{P(F_i(x)/nonObject)} \geq \lambda$$



- The best any classifier can do in this case is labeling an object with the label for which the probability density function (multiplied by the a priori probability) is highest.

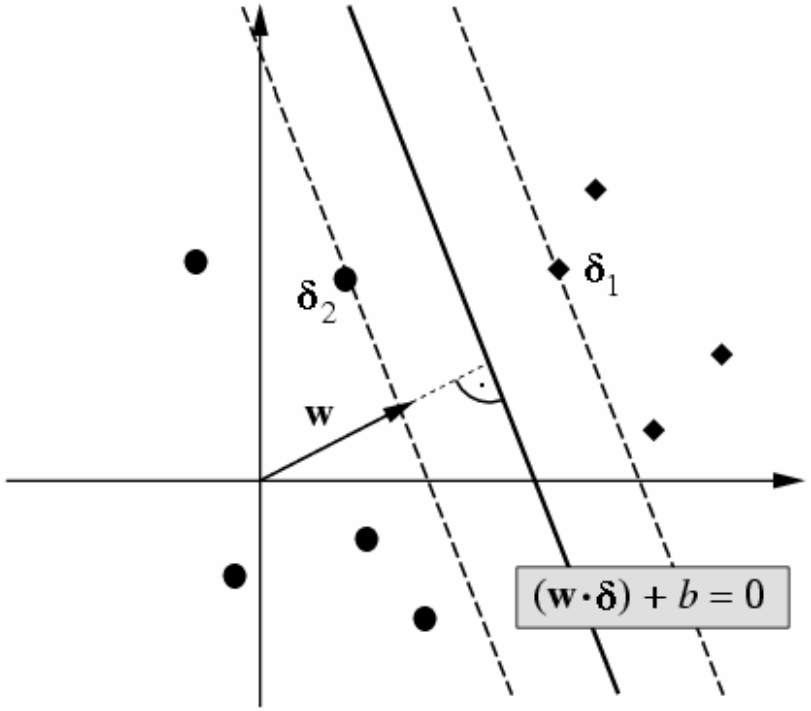
Bayes Classifiers

- Training Procedure:
  - Estimate $P(F_k(x)/Object)$ and $P(F_k(x)/nonObject)$ using a parametric model or using histograms.
  - Each of the histograms represents the statistic of appearance given by $F_k(x)$



SVM (1)
Support Vector Machine

- The idea is to determinate an hyperplane that separates the 2 classes optimally.
- Margin of a given sample: its distance to the decision surface (hyperplane).
- The optimum hyperplane is the one that maximize the margin of the closest samples (for both classes).



- The normal vector of the plane, $\mathbf{w}$, is defined so for the two classes (faces/non-faces):

$$\mathbf{w}\cdot\boldsymbol{\delta} + b > 0 \Rightarrow \boldsymbol{\delta} \in \boldsymbol{\Omega}_I$$

- Then, the value given by the classifier must be: $S(\boldsymbol{\delta}) = \mathbf{w}\cdot\boldsymbol{\delta} + b$

SVM (2)

"KERNEL TRICK":

$$\mathbf{x}^T\mathbf{y} \longrightarrow K(\mathbf{x},\mathbf{y})$$

Examples:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+1)^d$$

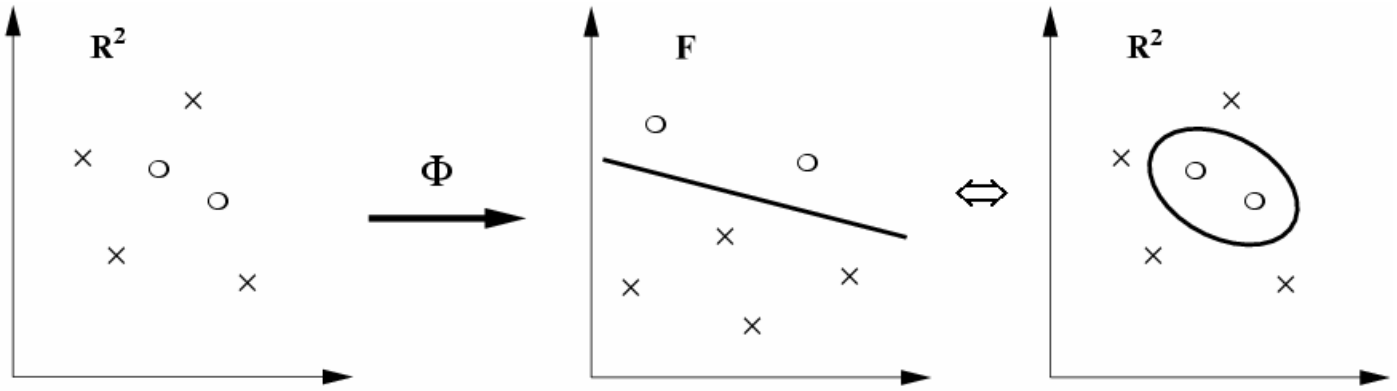
$$K(\mathbf{x},\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \quad \textbf{(RBF)}$$

$$K(\mathbf{x},\mathbf{y}) = \tanh(k\,\mathbf{x}^T\mathbf{y}-\theta)$$

- If K(**x**,**y**) satisfies the Mercer conditions , then the following expansion exists:

$$K(\mathbf{x},\mathbf{y}) = \sum_i \phi_i(\mathbf{x})\,\phi_i(\mathbf{y}) \quad = \boldsymbol{\Phi}(\mathbf{x})^T\,\boldsymbol{\Phi}(\mathbf{y})$$

- This is equivalent to perform a internal product of the mapped vector in F using the function: $\Phi : R^N \rightarrow F$



- The output given by the classifier is:

$$S(\boldsymbol{\delta}) = \sum_{\substack{\text{Support}\\ \text{Vectors}}} y_i\,K(\boldsymbol{\delta}_i,\boldsymbol{\delta}) + b$$

$\boldsymbol{\delta}$: new projected difference

$\boldsymbol{\delta}_i$: projected difference

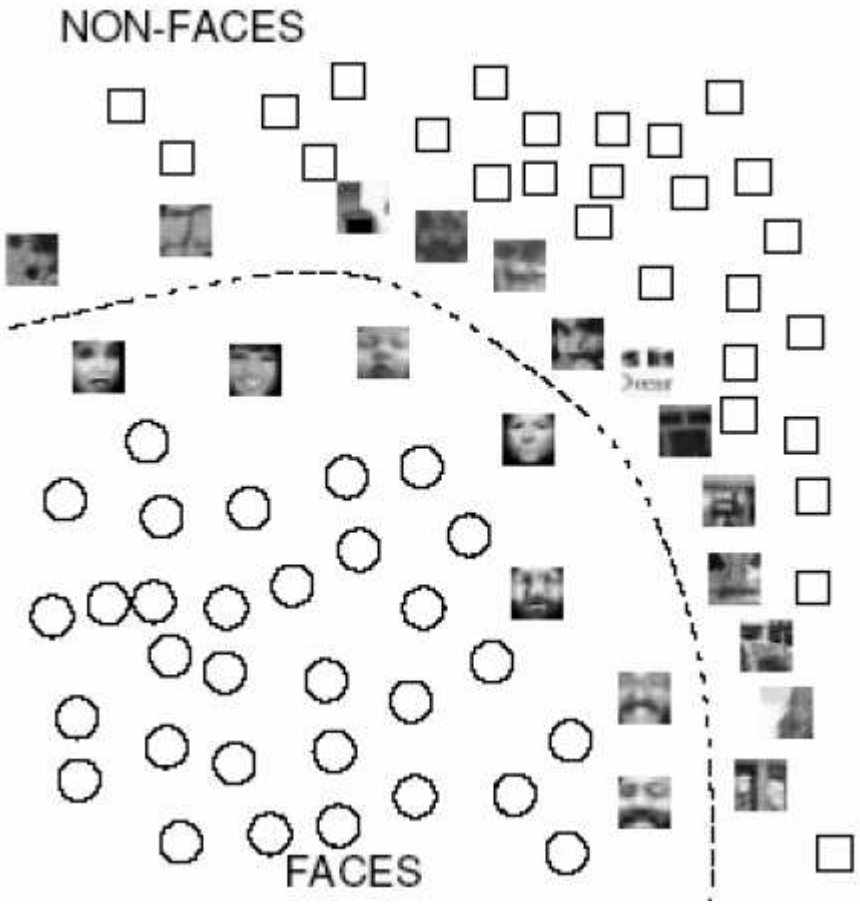
y_i : labels (+1: faces, -1 non-faces)

SVM (3)

SVM main idea:

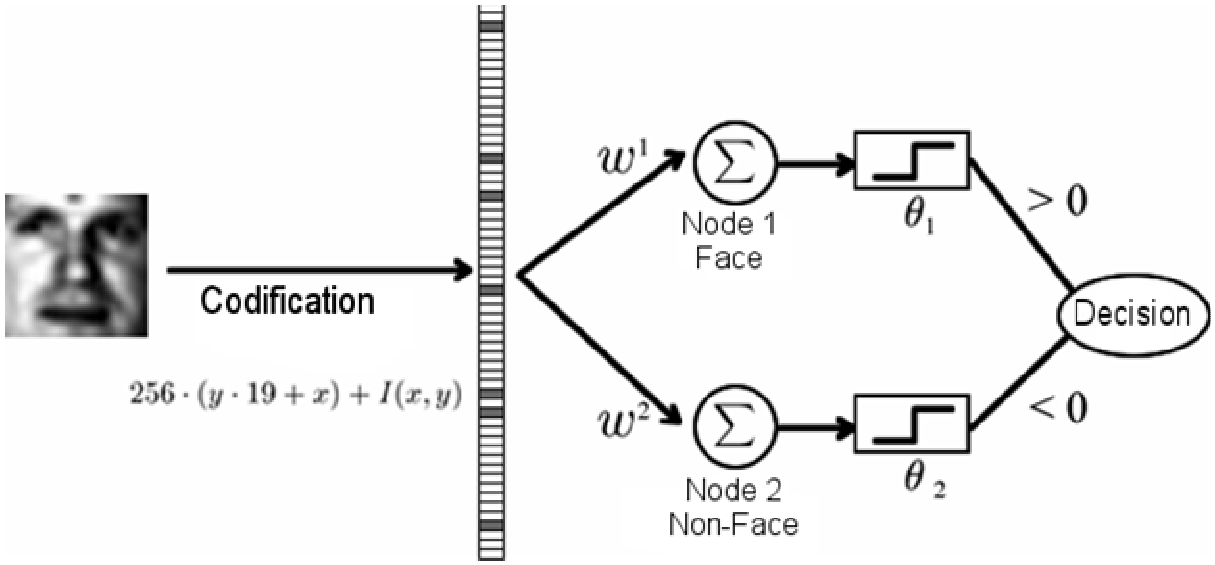
•The best hyperplane (or decision surface) is the one that is far from for the more difficult examples.

• It maximizes the minimal the margin



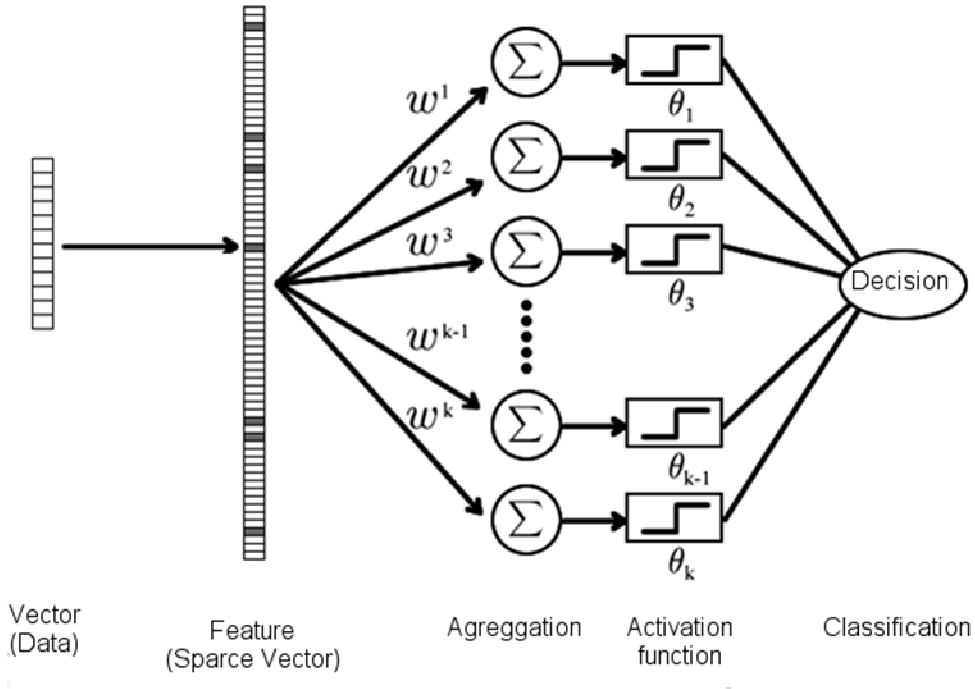
SNoW (1)
Sparse Network of Winnows

- The analysis window is codified as a sparse vector (current activated values from all possible values).
- For example, if windows of 19x19 pixels are being used, only 19x19=361 out of 19x19x256 (= 92416) components of the vector are activated.
- There are two nodes, one for faces and one for non-faces.
- The output of the vectors are a weighted sum of the components of the binary sparse vector.
- The output of the two nodes is used to take the decision of the classification.



SNoW (2)
Sparse Network of Winnows

$\mathcal{A}_t = \{i_1, ..., i_m\}$ $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$



(General diagram for k classes)

Training

If x = -1 and $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$

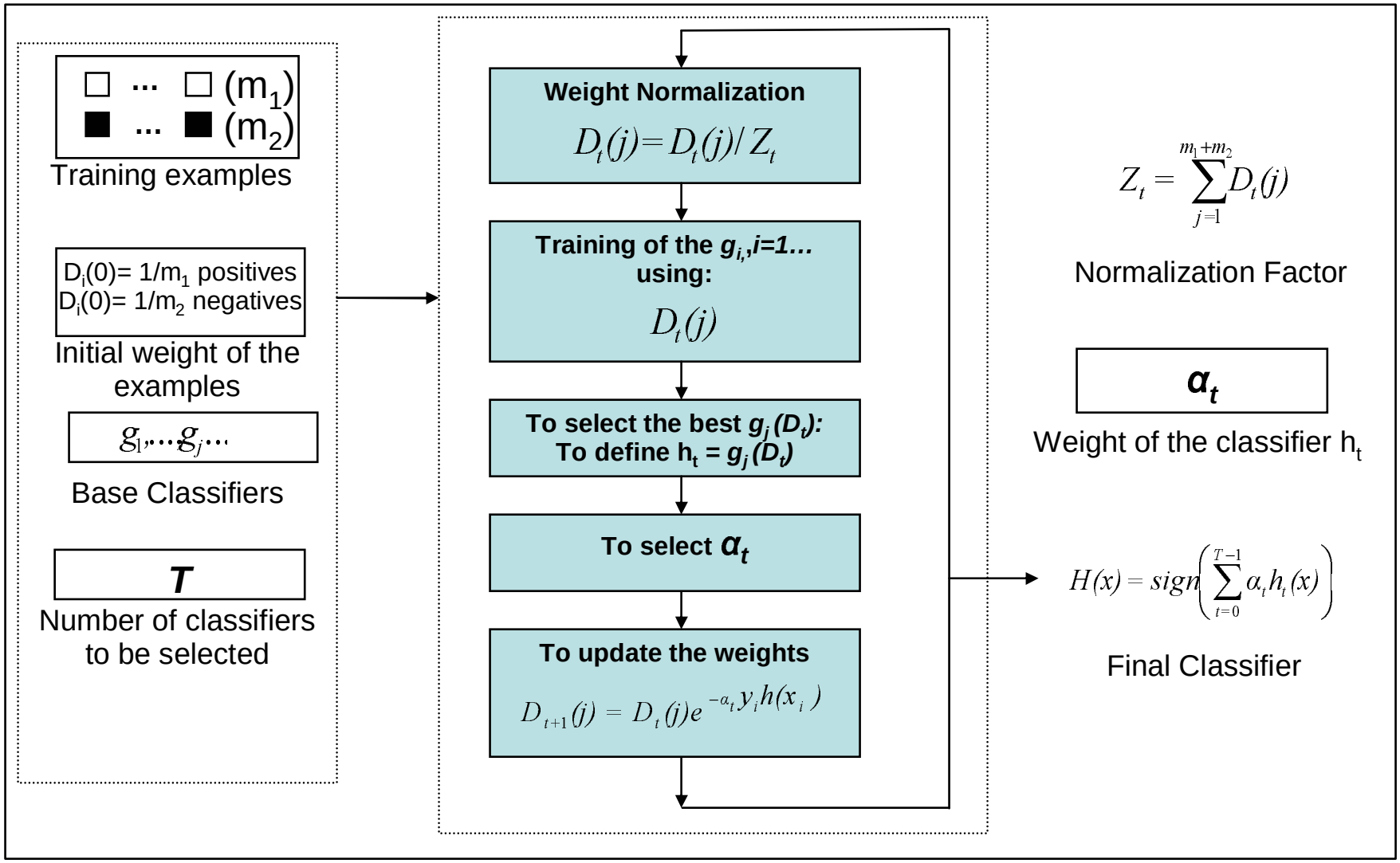
$$\forall i \in \mathcal{A}_t, \quad w_i^t \leftarrow \beta \cdot w_i^t$$

If x = +1 and ! $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$

$$\forall i \in \mathcal{A}_t, \quad w_i^t \leftarrow \alpha \cdot w_i^t$$

$0 < \beta < 1$ $\alpha > 1$

Adaboost


□ ... □ (m1)
■ ... ■ (m2)
Training examples
Di(0)= 1/m1 positives
Di(0)= 1/m2 negatives
Initial weight of the examples
g1,...gj...
Base Classifiers
T
Number of classifiers to be selected
Weight Normalization
Dt(j)= Dt(j)/ Zt
Training of the gi,i=1... using:
Dt(j)
To select the best gj (Dt): To define ht = gj (Dt)
To select αt
To update the weights
Dt+1(j) = Dt(j)e −αt yi h(xi)
Zt = Σ_{j=1}^{m1+m2} Dt(j)
Normalization Factor
αt
Weight of the classifier ht
H(x) = sign(Σ_{t=0}^{T−1} αt ht(x))
Final Classifier

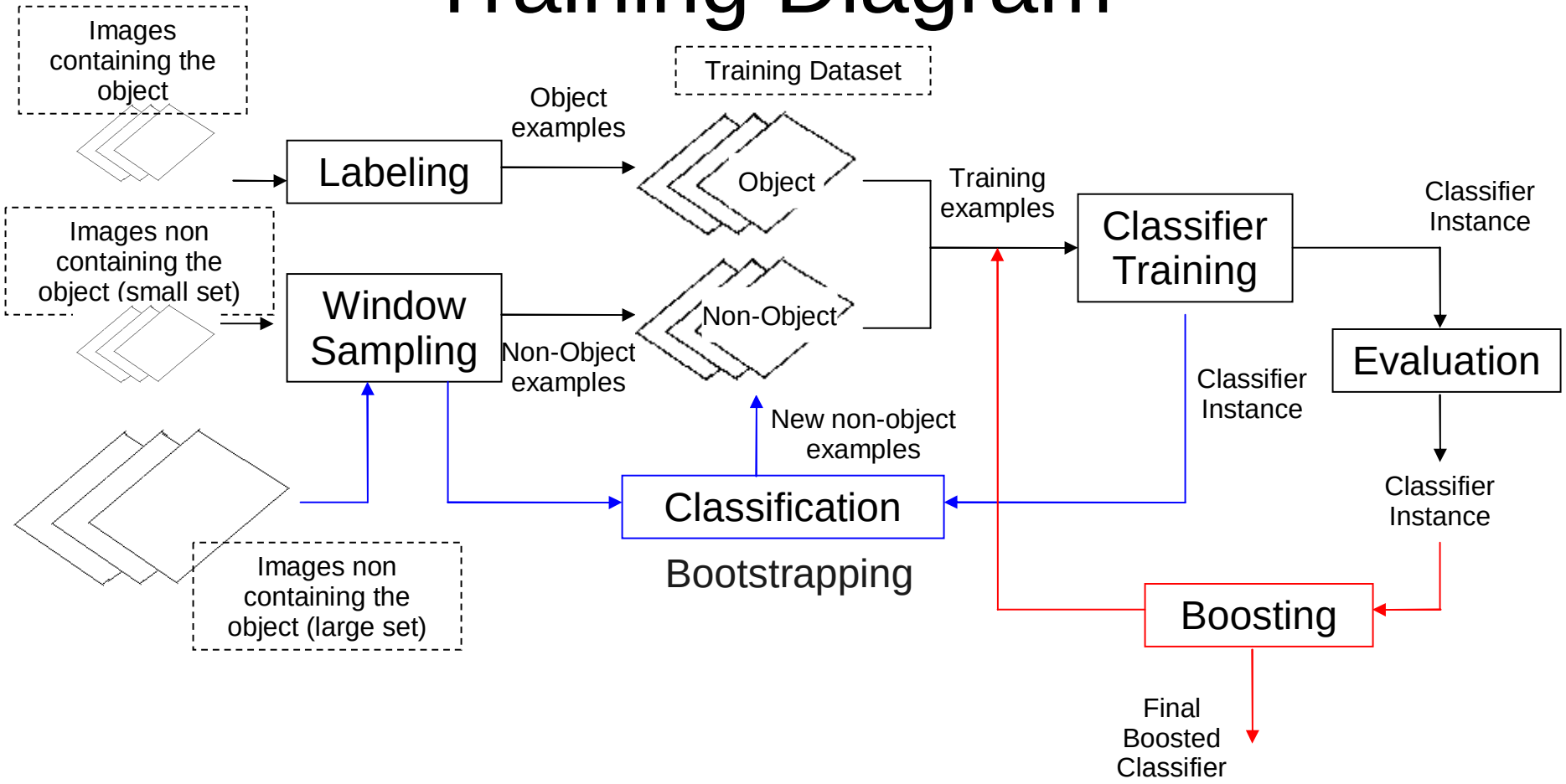
General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Applications to face analysis problems

Classifiers Training

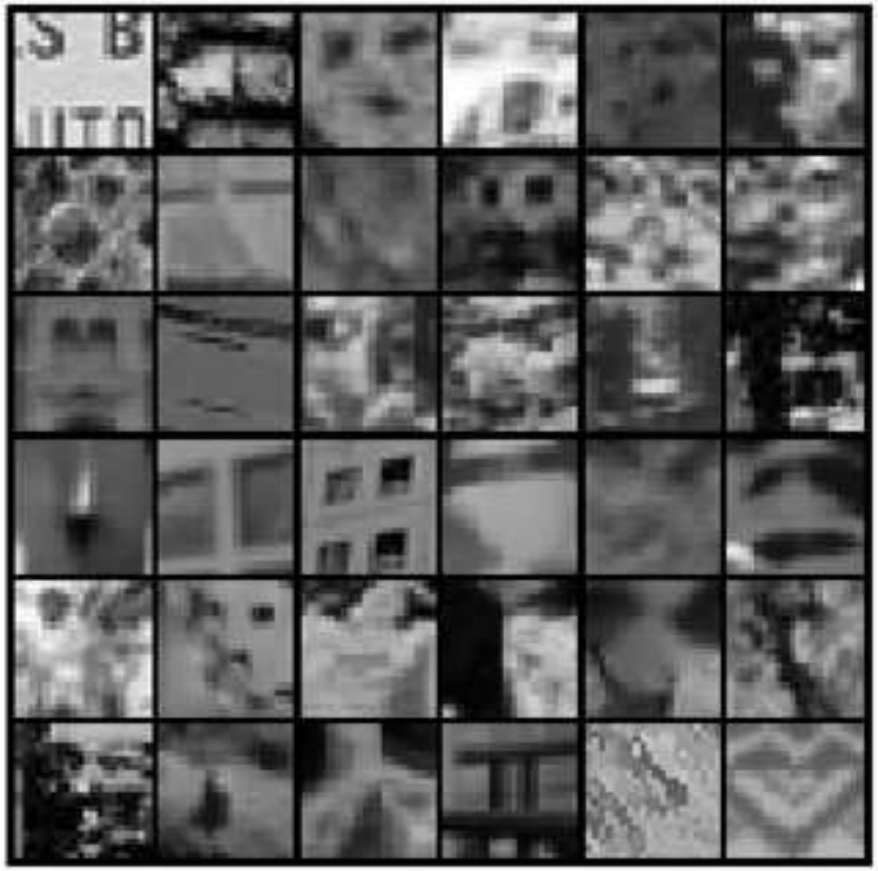
- Training procedures are as important as learning capabilities.
- “When developing a complex learning machine, special attention should be given to the training process. It is not only important the adequate selection of the training examples, they should be statistically significant, but also their distribution between the different classes, and the way in which they are “shown” to the learning machine”
- Evidences:
  - Importance of teachers in education
  - Dogs trainners

Training Diagram



Training Sets

Faces

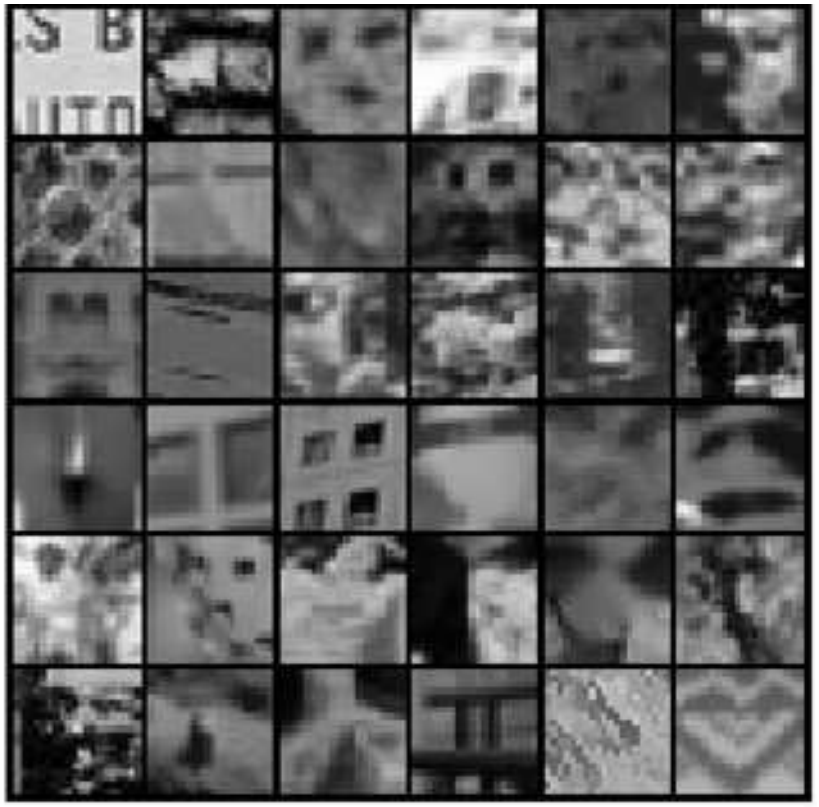


Non-Faces

[Images from: Ce Liu & Hueng-Yeung Shum, 2003]

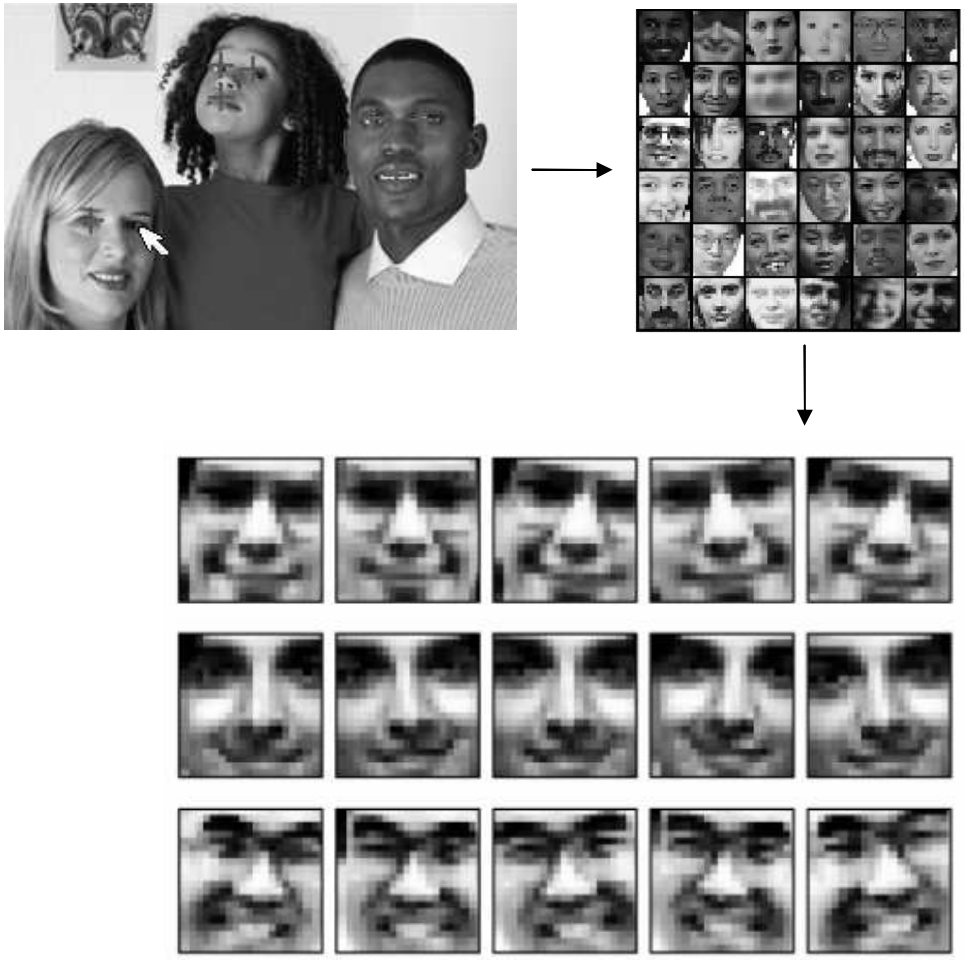
Generating initial training sets

- Non-Object:
  - First: find images that do not contain the object.
  - Then: randomly sample windows from these images.



Generating initial training sets

- Faces:
  - Labeling
    - Reference points need to be annotated
  - Increasing the training set: To generate Variations
    - Mirroring
    - Translations
    - Scale changes
    - Adding Noise
    - Rotation
    - Simulating different illumination



Generating initial training sets

- Which image size should be used?
  - 15x15, 19x19, 20x20, 24x24 or 38x38?
- How much of the object should the window include?
- Does the window should include part of the background?

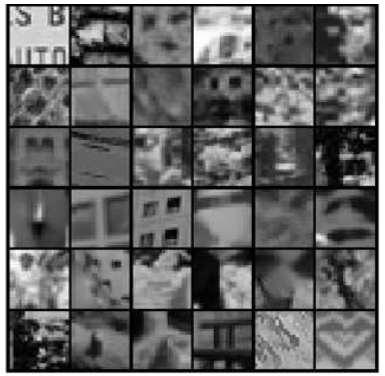
  → How does this affects the preprocessing of the windows?

Training of the Classifier:

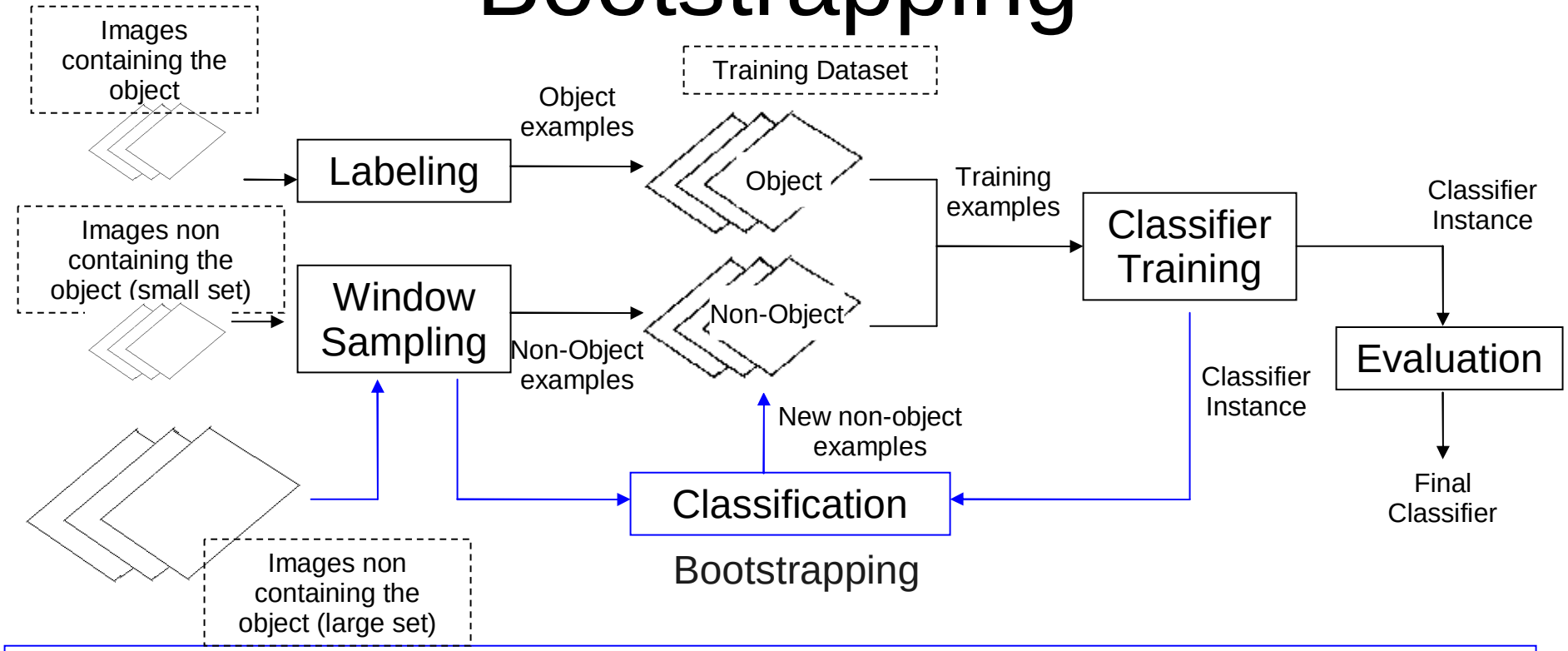
Generation of the initial training sets

- Faces:
  - Annotation: To annotate the reference points
  - Generation of variants of the annotated faces: *Mirrowing*, Translations, Scale Changes, Rotations, to add noise, etc.

- Non-faces:
  - To gather images containing no faces
  - To sample windows from these images.

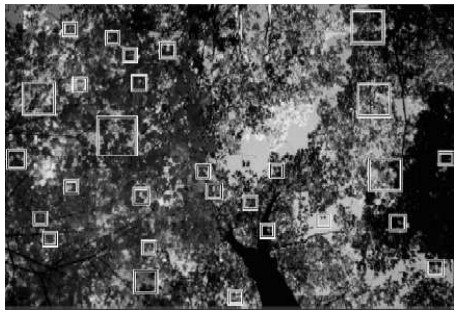


Bootstrapping



- Bootstrapping consists on collecting non-face windows that are wrongly classified (as faces) by the trained classifier and then to add them to the training set.

- This is done to obtain a better representation of the non-face class in the "border" of the two classes.



Training the Classifier:

Bootstrapping

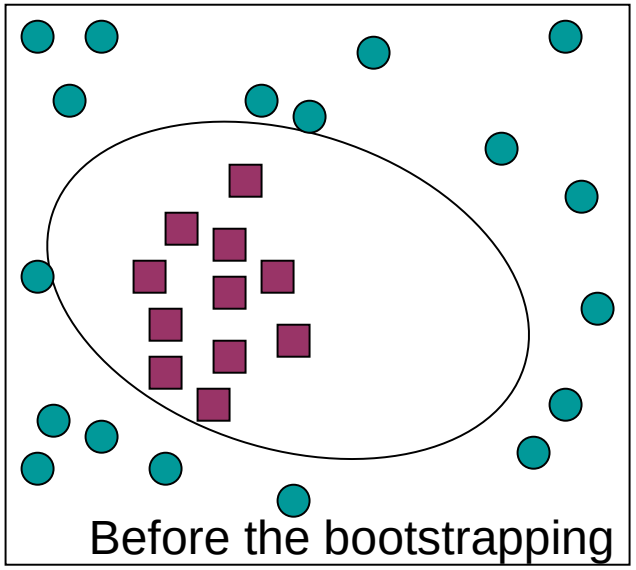
Objective:

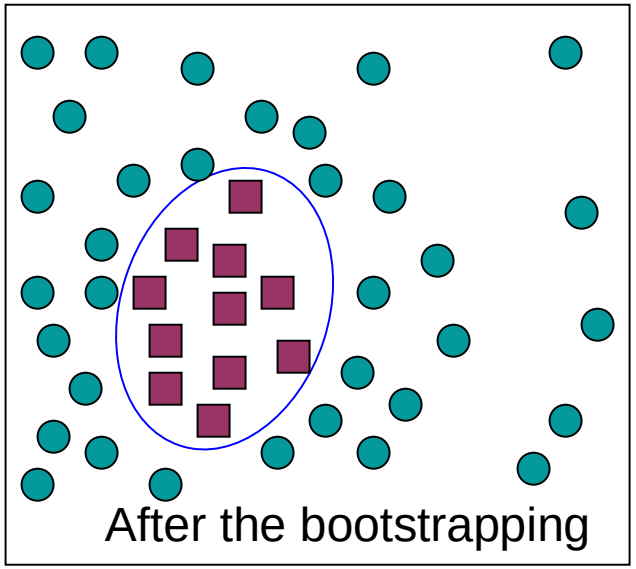
- To Obtain a better representation of the non-face class in the boundary with the face class.

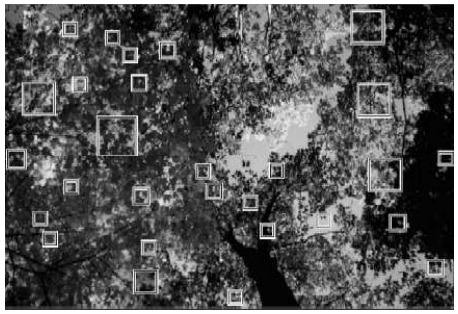
Procedure:

1. To train the classifier.
2. To collect new non-face windows that were wrongly classified as faces by the recently trained classifier.
3. To add these windows to the training set.
4. Go to (1) of one of the following conditions is not achieved:
   a. The false positive rate is not low enough.
   b. There is no improvement in the performance of the classifier.

[Sung and Poggio 96]



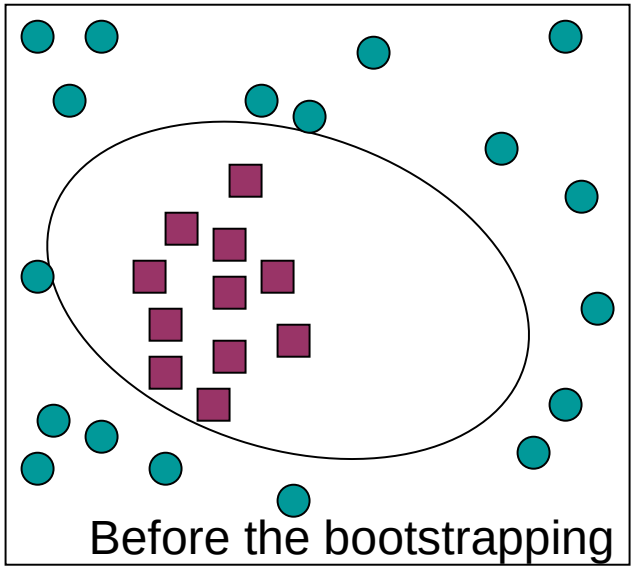


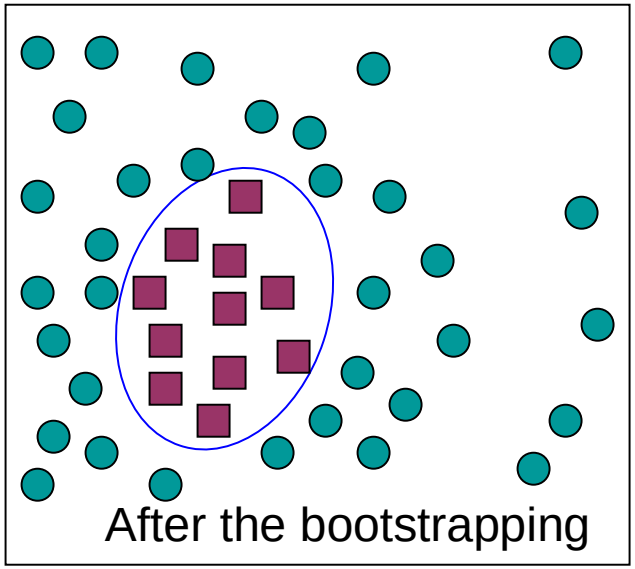


Training the Classifier:

Bootstrapping

- **Boostrapping is very important when the a priori probability of occurrence of the classes is very different**





[Sung and Poggio 96]

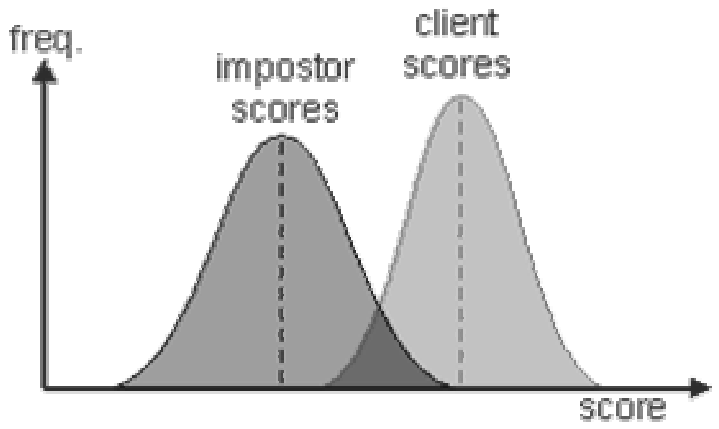
General Outline

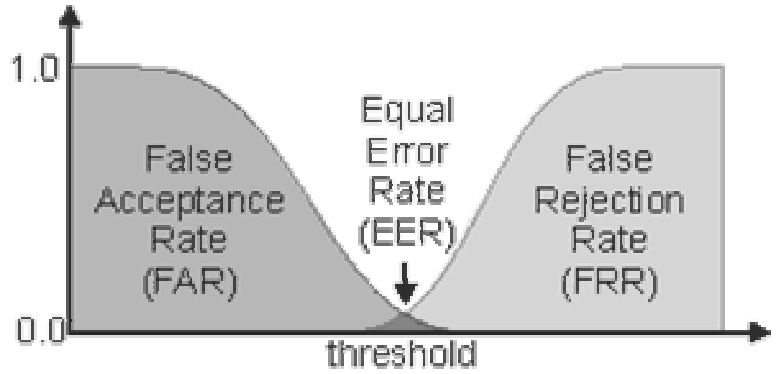
- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Applications to face analysis problems

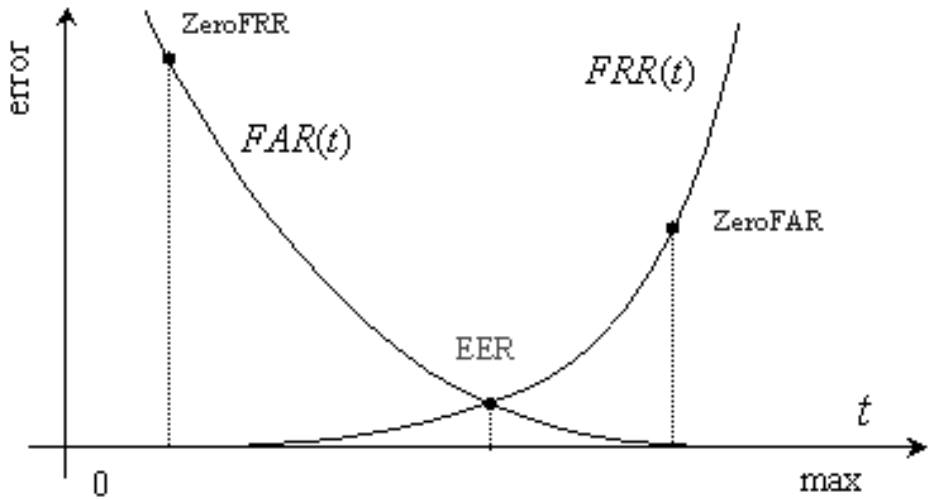
Evaluation protocol

- How to compare the results of different detectors?
- Procedure?
  - Training/test set
  - When does a face is correctly detected?
  - True positives vs False negatives
  - Precision
  - Speed (training/test)

FRR and FAR

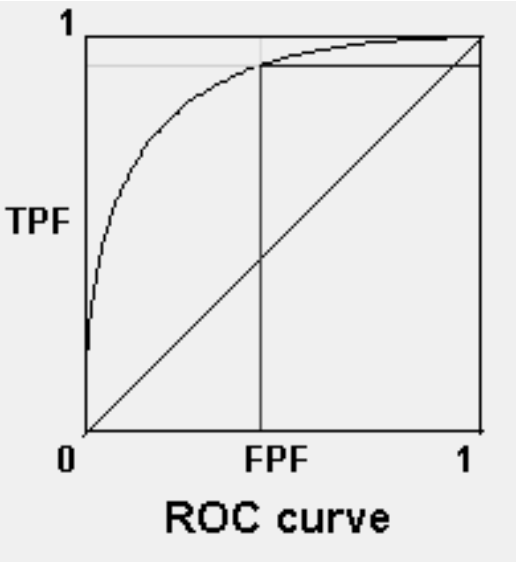

freq.
impostor scores
client scores
score


1.0
0.0
False Acceptance Rate (FAR)
Equal Error Rate (EER)
False Rejection Rate (FRR)
threshold


error
ZeroFRR
FRR(t)
FAR(t)
ZeroFAR
EER
t
0
max

Performance Evaluation

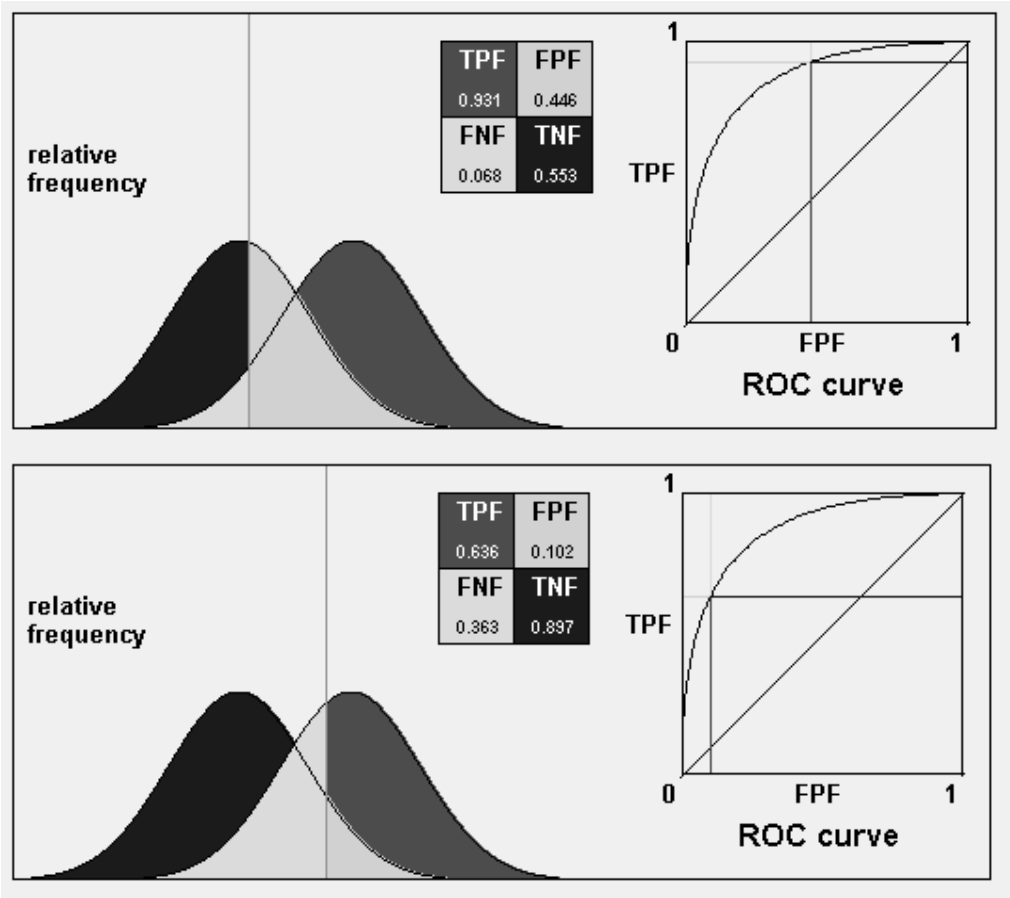
- ROCs (Receiver Operating Characteristic Curves)
  - True Positives Rate (TPR):
    - Fraction of *positives* (faces), classified as *positives.*
  - True Negative Rate (TNR):
    - Fraction of *negatives* (non-faces), classified as *negatives.*
  - False Positives Rate (FPR):
    - Fraction of *positives*, classified as *negatives.*
  - False Negative Rate (FNR):
    - Fraction of *negatives*, classified as *positives.*



TPR + FNR = 1
TNR + FPR = 1

Performance Evaluation

- A ROC represents the accuracy of the classifier for differentiating the two classes.
- They define how overlapped are the outputs for the two classes (the classifier output distribution for the two classes).
- When different threshold levels are applied to the classifier output, different operation points are obtained

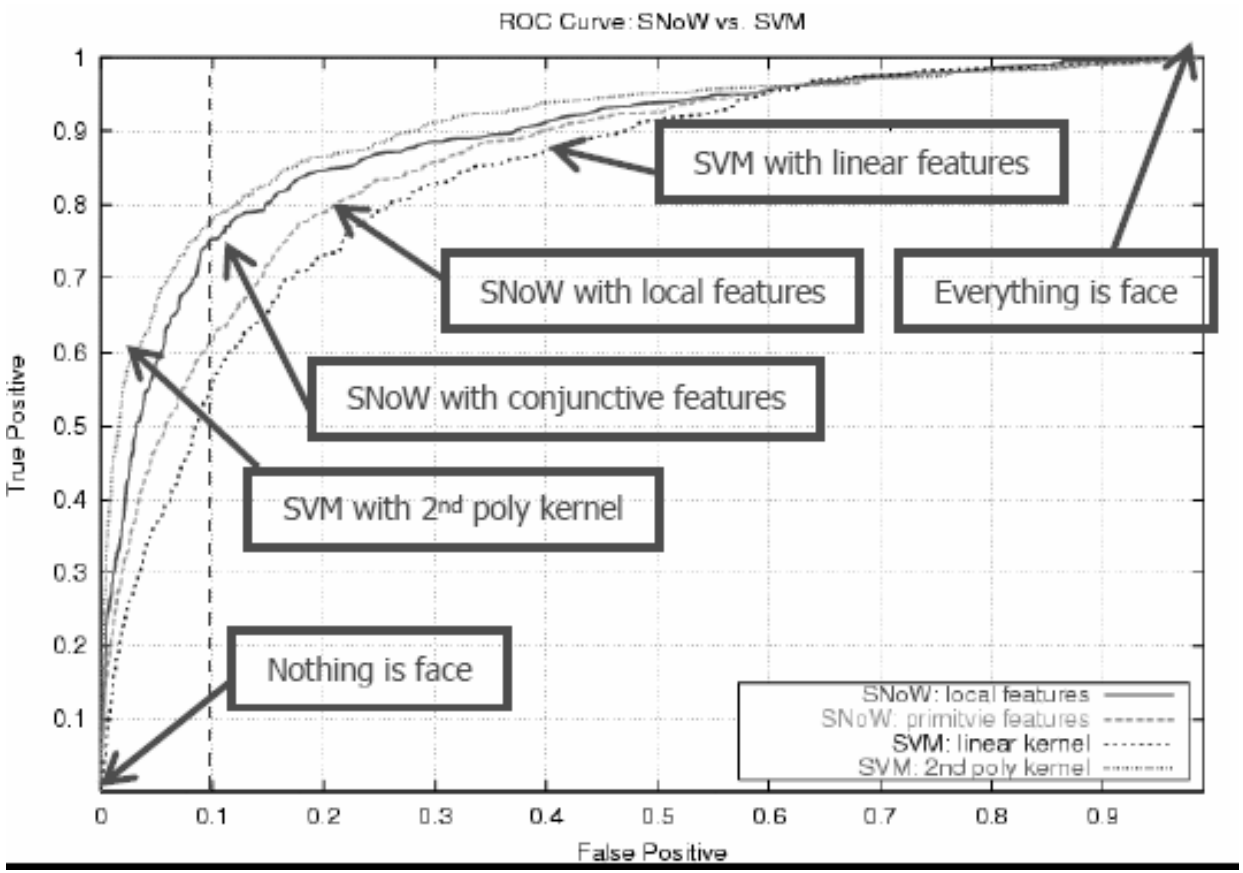


In the example the output of the classifiers of each of the classes follows a Gaussian distribution

Images from: http://www.anaesthetist.com/mnm/stats/roc/

Results

- ROCs: Result comparison example



Part 2

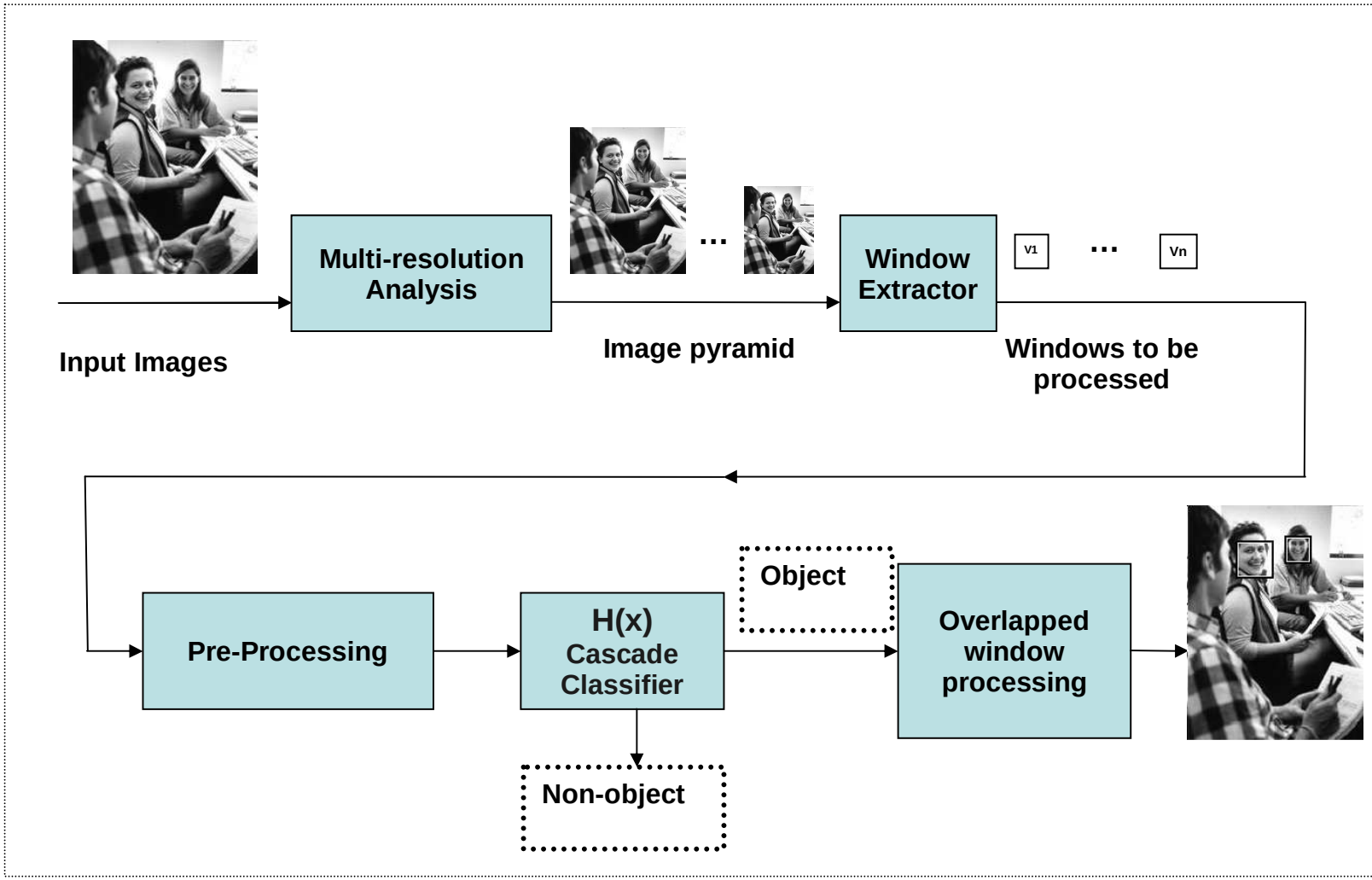
Nested Cascade of Boosted Classifiers

General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Face analysis using cascade classifiers

Detecting Objects:

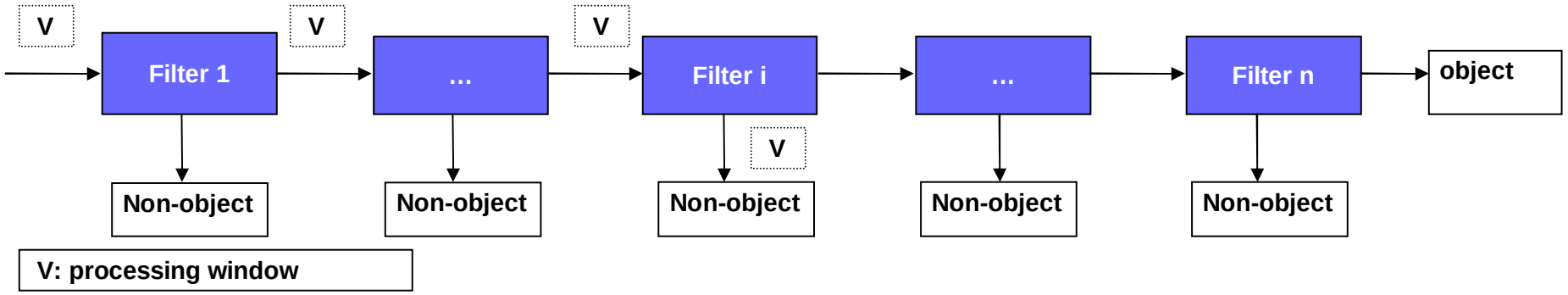
Standard Architecture for detecting objects



State of the Art Cascade Detectors

- Faces (Multiview)
  - Jones M., Viola P., “Fast Multi-view Face Detection”, CVPR, June, 2003.
  - Wu B., et al, “Fast Rotation Invariant Multi-view Face Detection based on Real Adaboost”, FG 2004
- Cars, Faces (frontal and profile), Traffic signs, etc. (10 different objects)
  - Schneiderman H., “Feature-Centric Evaluation for Efficient Cascaded Object Detection”, CVPR 2004
- Pedestrians (detection, in videos, of people walking)
  - Viola P., Jones M., Snow D., “Detecting Pedestrians Using Patterns of Motion and Appearance”, ICCV 2003
- Hands
  - Koelsch M. ,Turk M., “Robust Hand Detection”, FG2004
- People, Cars, Faces, Traffic signs, Computers Monitors, Chairs, Keyboards, etc. (21 different objects)
  - Torralba A., Murphy K., and Freeman W., “Sharing Visual Features for Multiclass and Multiview Object Detection”, AI Memo 2004-008, April 2004, massachusetts institute of technology — computer science and artificial intelligence laboratory

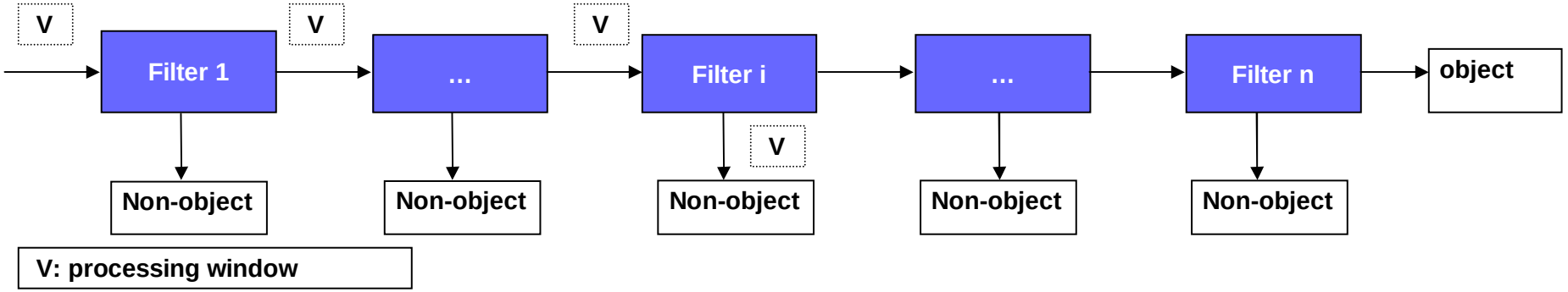
Cascade Classifier



- Each filter:
  - rejects (most) non-object windows and
  - let object windows past to the next layer of the cascade.
- A window will be considered as a object if and only of all layers of the cascade classifies it as object.
- The *filter i* of the cascade will be designed to:
  - (1) reject the larger possible number of non-object windows,
  - (2) to let pass the larger possible number of object windows and
  - (3) to be evaluated as fast as possible (there is always a trade-off between these 3 objectives)

[Viola & Jones 2001]

Cascade Classifier



- Attentional cascade
  - False Positive Rate:

$$f_{total} = \prod_{i=0}^{N-1} f_t \qquad if \forall\, t,\ f_t = 0.5\ ,\ N = 20 \Rightarrow f_{total} \approx 1\text{x}10^{-6}$$

    - For example, in a 600x600 pixels image the expected number of false positives is 1.
  - Detection Rate

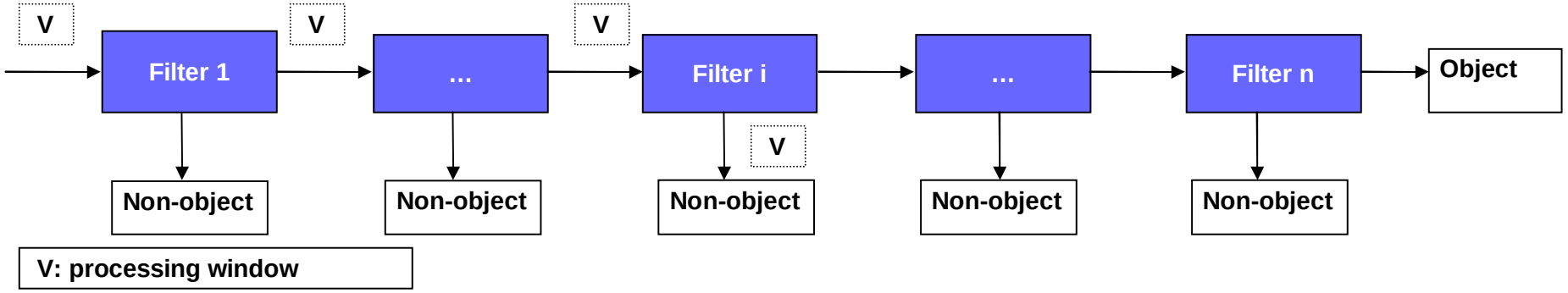
$$d_{total} = \prod_{i=0}^{N-1} d_t \qquad if \forall\, t, d_t = 0.999\ ,\ N = 20 \Rightarrow d_{total} \approx 0.9802$$

$$if \forall\, t, d_t = 0.995\ ,\ N = 20 \Rightarrow d_{total} \approx 0.9046$$

  - Processing speed

$$nf_{total} = nf_0 + \sum_{i=1}^{N-1} nf_i \prod_{j=0}^{i-1} f_j$$

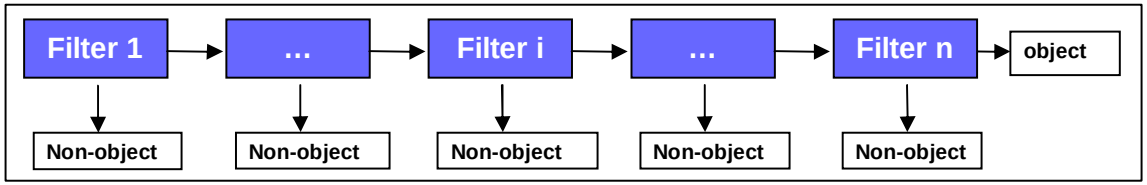
¿Why to use a cascade Classifier?



- Asymmetric Distribution of the classes:
  - In an image there are much more non-object windows than object windows
  - Therefore the average processing time of the windows is defined by the processing time of the non-object windows.
- To dedicate less time (in average) to the non-object windows:
  - Many non-object windows are “vey different” to object windows, i.e. it is easy to classify them as non-objects.
  - Most windows will be classified as non-object by the first filters (also called stages or layers) of the cascade.
  - Therefore the *filter i* should be evaluated faster than the *filter i+1*
  - The firs and second filters are the most important ones for the processing speed

[Viola & Jones 2001]

Cascade Classifier: Training



f: maximum allowed of false positives per layer
d: minimum allowed detection rate per layer
F_{total}: final false positive rate
P: training set of Objects
N: training set of non-Objects

H_i : layer i of the cascade
H: final classifier
Initialization: $F_{all} = 1.0$, $D_{all} = 1.0$, $ND = 0$, $i = 0$

While $F_{all} > F_{total}$

$i = i+1$ — Start training new layer

Train H_i such that $f_i \leq f$, $d_i \geq d$ — Train new layer

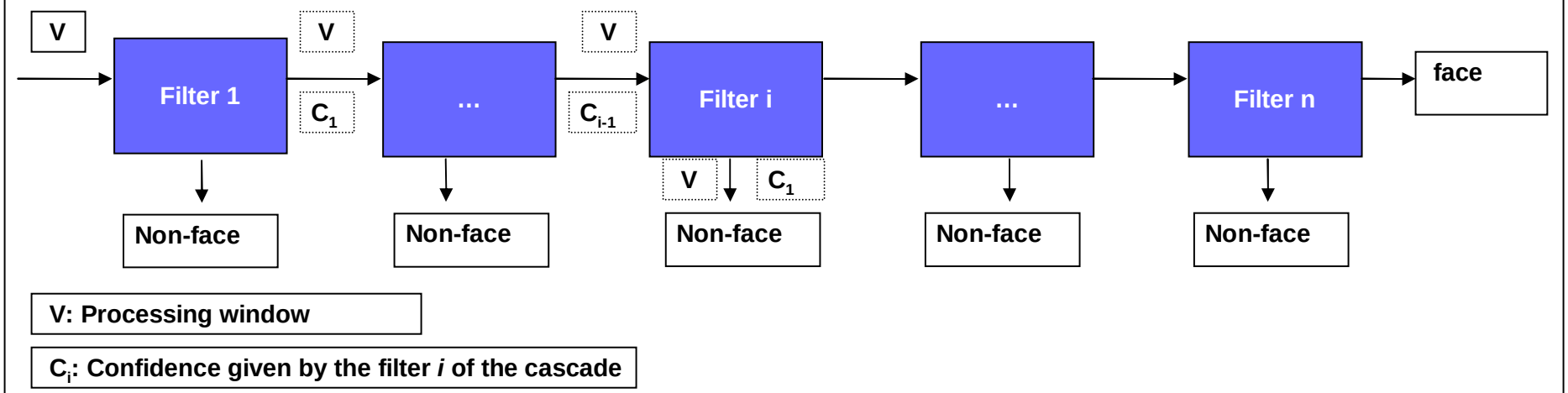
Update the detection and false positive rates of the cascade $F_{all} = F_{all} * f_i$, $D_{all} = D_{all} * d_i$ — Get new rates

Add the new layer to the cascade $H = H \Theta H_i$ — Add new layer to the cascade

Re-sample the set of non-faces, *N,* using *H()* — Get non object training set

Nested Cascade:
Concept

- To use the confidence of the output of a layer of the cascade as a part of the next layer of the cascade
- Objective:
  - To share information between layers of the cascade
  - To obtained a more compact and robust cascade.



[Bo WU et al. 2004]

General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Face analysis using cascade classifiers

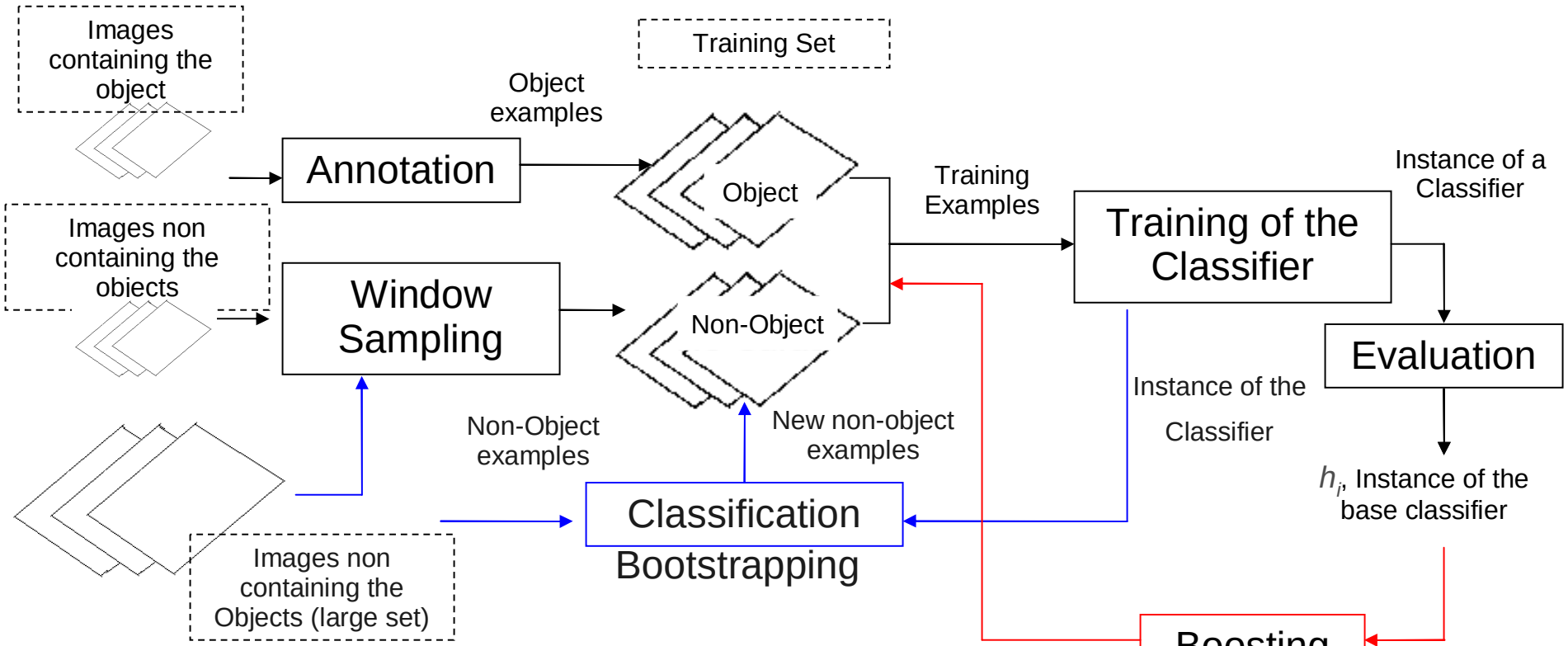
Boosting:
Description

- It's a particular case of ensemble learning
- Concept
  - To train the same classifier several times (classifier base).
  - In each iteration a different training set (or distribution) is used, obtaining a weak classifier (also called hypothesis o instance of the base classifier).
  - The obtained weak classifiers will be combined to form a final classifier (also called robust classifier).
- Questions
  - ¿How to choose the distribution/training set in each iteration?
  - ¿How to combine the obtained classifiers?
- There are several different boosting algorithms that answer these questions in different ways :
  - Bagging (Bootstrap aggregation)
  - AdaBoost (Adaptive Boosting)
  - Cross-validated committees
  - LogitBoost
  - etc.

Discrete Adaboost : [Y. Freund and R. Shapire 1995]
Real Adaboost, also Multi-class: [R. Shapire and Y.Singer 1999]
Firsts use of Adaboost in face detection: [Viola & Jones 2001] and [Schneiderman 2000]

Training the classifier:

Block Diagram:



- The classifier is trained several times
- In each iteration the training set is modified (sampled or weighted)
- The classifiers obtained at each iteration are combined in a final classifier

Boosting

- Boosting:
  - To train a base learning algorithm repeatedly, each time with a different subset of (or distribution on) the training examples.
  - At each iteration an instance of the base classifier is obtained.
  - The obtained instances are usually called weak classifiers, weak learners, weak rules, or hypothesis.
  - The weak classifier does not need to have a high accuracy.
  - The base classifiers are to be combined in a called strong classifier, which is much more accurate.

$$H(x) = H(h_1(x),...,h_T(x))$$

$$H_t(x) = H_t(H_{t-1}(x), h_t(x))$$

Boosting

- How should each distribution be chosen?
- How to combine the base classifiers?
- Some boosting algorithms:
  - Bagging (Bootstrap aggregation)
  - Cross-validated committees
  - Discrete AdaBoost (Adaptive Boosting)
  - Real AdaBoost
  - Arc-gv
  - LogitBoost
  - …
- Boosting can also be used for the feature/classifier selection.
- Boosting works well with unstable base learners, i.e. algorithms whose output classifier undergoes mayor changes in response to small changes in training data for example,
  - decision trees
  - neural networks
  - decision stumps

Adaboost

- Adaboost (Adaptive Boosting):
  - It “adapts” the distribution of the training examples

    → every weak learner is trained with a different “adapted” distribution.
  - The distribution of the examples is adapted, so that during the training every weak learner focuses more on previously wrongly classified examples.

    → the algorithm focuses on hard examples
  - The training error diminishes exponentially with the number of boosted classifiers.

    → greedy search

Adaboost

- Adaboost adapts a distribution of weights on the training set; each example of the training set has an associated weight

$$\{x_i, y_i D(i)\}_{i=1,\ldots,m}, x \in X, y_i \in \{-1,+1\}, D_0(i) = \frac{1}{m}, i = 1,\ldots,m$$

- In each iteration:
  - A weak classifier that minimizes a error function depending on the weights D_0, is obtained.
  - The weight distribution D_0 is updated.
- Weights are update:
  - new classifier will focus on the examples wrongly classified by the previously trained classifiers.
- The final classifier is a weighted sum of the weak classifiers:

$$H(x) = sign\left(\sum_{t=0}^{T-1} \alpha_t h_t(x)\right)$$

Adaboost

$$\text{margin} : yf(x) \qquad H(x) = \text{sign}(f(x))$$

- Many learning algorithm (can) generate classifiers with large margin:
  - SVM
  - Neural Networks
  - Adaboost

If $\text{sign}(y) = \text{sign}(f(x))$ then $f()$ classifies x correctly.

$|f(x)|$ indicates the confidence of the classification.

Therefore we want $yf(x) \geq \theta \geq 0$ for large θ

(Real) Adaboost

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

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- 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 normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

(Real) Adaboost

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

- ~~Train weak learner using distribution D_t.~~
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$h_t = \arg\min_{g \in G} Function(D_t, (x_1, y_1), ..., (x_m, y_m))$$

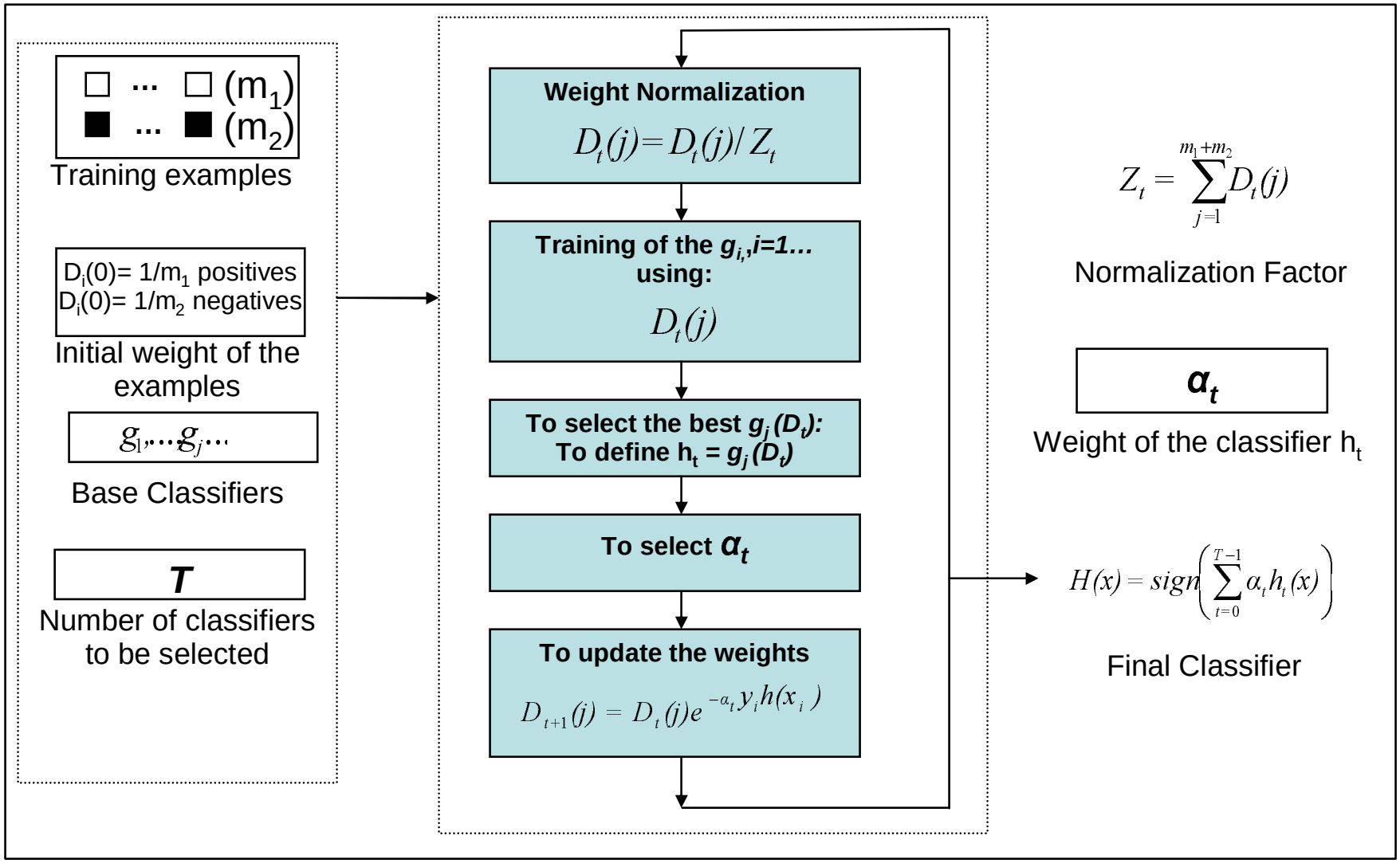
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Adaboost


□ ... □ (m1)
■ ... ■ (m2)
Training examples
Di(0)= 1/m1 positives
Di(0)= 1/m2 negatives
Initial weight of the examples
g1,...gj...
Base Classifiers
T
Number of classifiers to be selected
Weight Normalization
Dt(j)= Dt(j)/ Zt
Training of the gi,i=1... using:
Dt(j)
To select the best gj (Dt): To define ht = gj (Dt)
To select αt
To update the weights
Dt+1(j) = Dt(j)e −αt yi h(xi)
Zt = Σ_{j=1}^{m1+m2} Dt(j)
Normalization Factor
αt
Weight of the classifier ht
H(x) = sign(Σ_{t=0}^{T−1} αt ht(x))
Final Classifier

Adaboost

- Selecting α_t
  - Training Error: $\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right|$
  - Bound to the training error:

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} Z_t \qquad Z_t = \sum_i D_t(i) e^{-y_i \alpha_t h_t(x_i)}$$

  - Real Adaboost (bounded case): $h_t \in [-1, +1]$

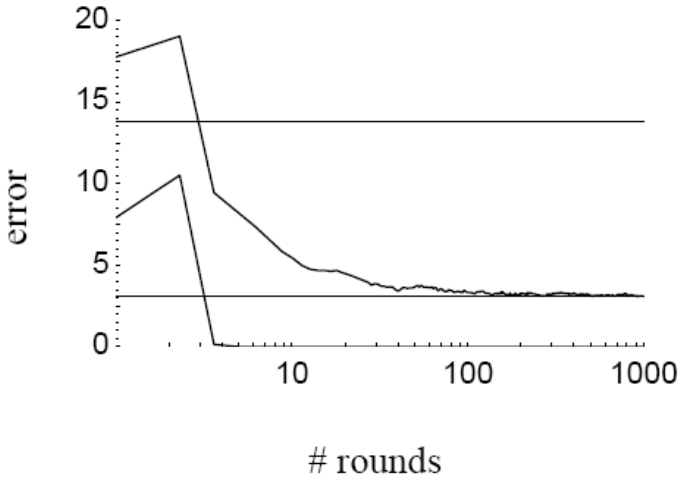
$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right) \qquad r_t = \sum_i D_t(i) y_i h_t(i) \leq 1$$

Adaboost

- "Properties"
  - The training error diminishes very fast when adding new weak classifiers (Real Adaboost, bounded case):

$$h_t \in [-1,+1] \quad r_t = \sum_i D_t(i) y_i h_t(i) \le 1 \quad \frac{1}{m} \left| \{ i : H(x_i) \neq y_i \} \right| \le \prod_{t=1}^{T} \sqrt{1 - r_t}$$

  - The final classifiers usually do not over fit (low out-off training set error) even when the training error is Zero and we continue to add weak classifier to the final robust classifier (empirical result).



Adaboost

- Why Adaboost works?

In other words:

- Why Adaboost has a low generalization error?

Partial explanations:
→ Margin
→ Consistency

(Real) Adaboost

$$Z_t = \sum_i D_t(i) e^{-y_i \alpha_t h_t(x_i)} = \frac{\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)}}{\prod_{j=1}^{t-1} Z_j} = \frac{\sum_i e^{-y_i f_t(x_i)}}{\prod_{j=1}^{t-1} Z_j}$$

$$\prod_{j=1}^{t} Z_j = \sum_i e^{-y_i f_t(x_i)}$$

$$H(x) = sign\left(\sum_{t=0}^{T-1} \alpha_t h_t(x)\right) = sign\big(f(x)\big)$$

- Adaboost minimizes an exponential loss on the margin

and at the same time minimizes a bound of the training error

$$\frac{1}{m} \left|\left\{ i : H(x_i) \neq y_i \right\}\right| \leq \prod_{t=1}^{T} Z_t = \sum_i e^{-y_i f_t(x_i)}$$

Adaboost

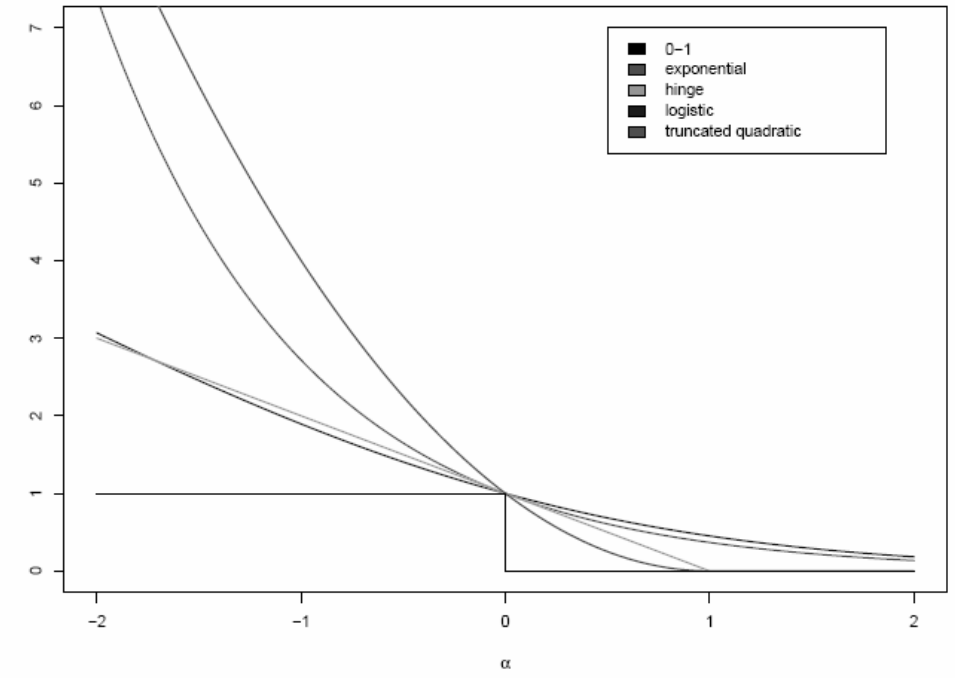
- Real Adaboost iteratively (greedy) minimize an exponential loss on the margin
- Real Adaboost iteratively (greedy) minimize the training error

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} Z_t = \sum_i e^{-y_i f_t(x_i)}$$

$$\text{margin} \quad : y\, f(x)$$

- Loss/Cost/Error Function

$$L = \exp(-\alpha)$$



Adaboost

- Theoretical Bound

  $S = \{(x_1, y_1), \ldots (x_m, y_m)\}$ is chosen according to a distribution P

  – Generalization error: $\Pr_{(x,y)\sim P}[H(x) \neq y] = \Pr_{(x,y)\sim P}[yf(x) \leq 0]$

  – With high probability:

$$\Pr_{(x,y)\sim P}[H(x) \neq y] \leq \Pr_{(x,y)\sim S}[yf(x) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m}}\frac{\log(m/d)}{\theta}\right)$$

  (There is a “similar” bound for Neural Networks)

Schapire, Freund, Bartlett and Lee, 1998

Adaboost

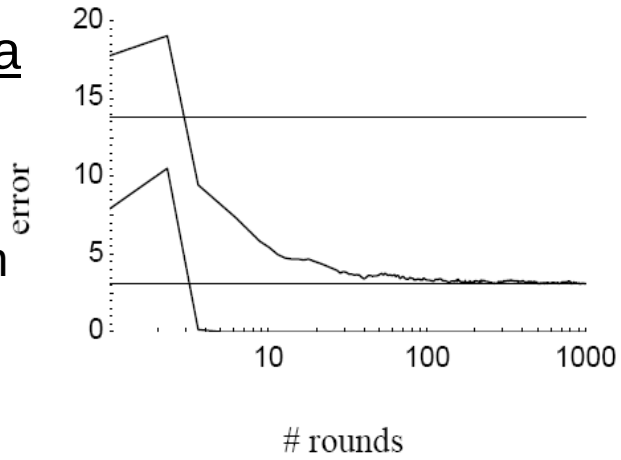
$$\Pr_{(x,y)\sim P}[H(x) \neq y] \leq \Pr_{(x,y)\sim S}[yf(x) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m}\frac{\log(m/d)}{\theta}}\right)$$

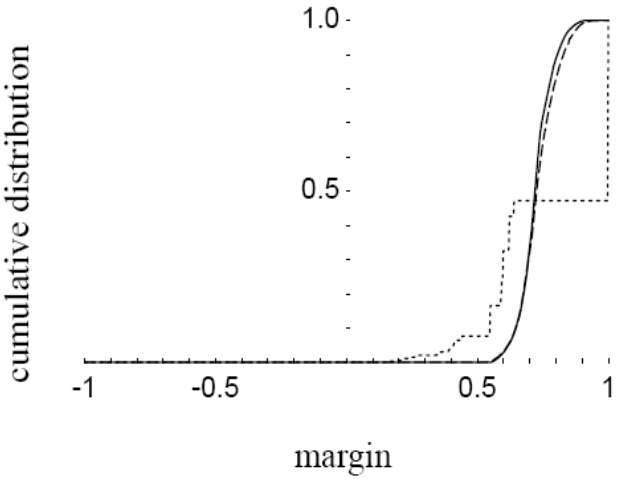
if $\varepsilon_t \leq \frac{1}{2} - \gamma, \gamma > 0$ **and** $\theta < \gamma$ **then**

$$\Pr_{(x,y)\sim S}[yf(x) \leq \theta] \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t^{1-\theta}(1-\varepsilon_t)^{1+\theta}}$$

$$\leq \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T \leq O(\exp(-T))$$

Adaboost and Margin, a real example:
left: training and test errors,
right: accumulated margin for 5, 100 and 1000 iterations





Adaboost

$$\Pr_{(x,y)\sim P}[H(x) \neq y] \leq \Pr_{(x,y)\sim S}[yf(x) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m}}\frac{\log(m/d)}{\theta}\right)$$

- $d = VC_{\dim}(G)$ the Vapnik-Chervonenkis dimension of the *G*,
- d measures the "complexity" of the family of functions G
- *m,* the number of training samples
- Does not depend on T, the number of weak classifiers!!
- (There is a "similar" bound for Neural Networks)

Consistency of Adaboost

- Definitions:
  - Risk: $R(f) = \Pr(\text{sign}(f(X)) \neq Y)$
  - Bayes Risk: $R^*(f) = \inf_{g \in G} R(g)$
  - Consistency:

  We say that learning method is universally consistent if, for all distributions *P*,

$$\lim_{m \longrightarrow \infty} R(f_m) \xrightarrow{a.s} R^*(f) = \inf_{g \in G} R(g)$$

- Adaboost
  - Under some assumptions Adaboost is consistent, including:

$$VC_{\dim}(G) < \infty$$
$$t_m = O(m^{\nu}), \nu < 1$$

(e.g. SVM are also consistent)

Bartlett 2006

Adaboost

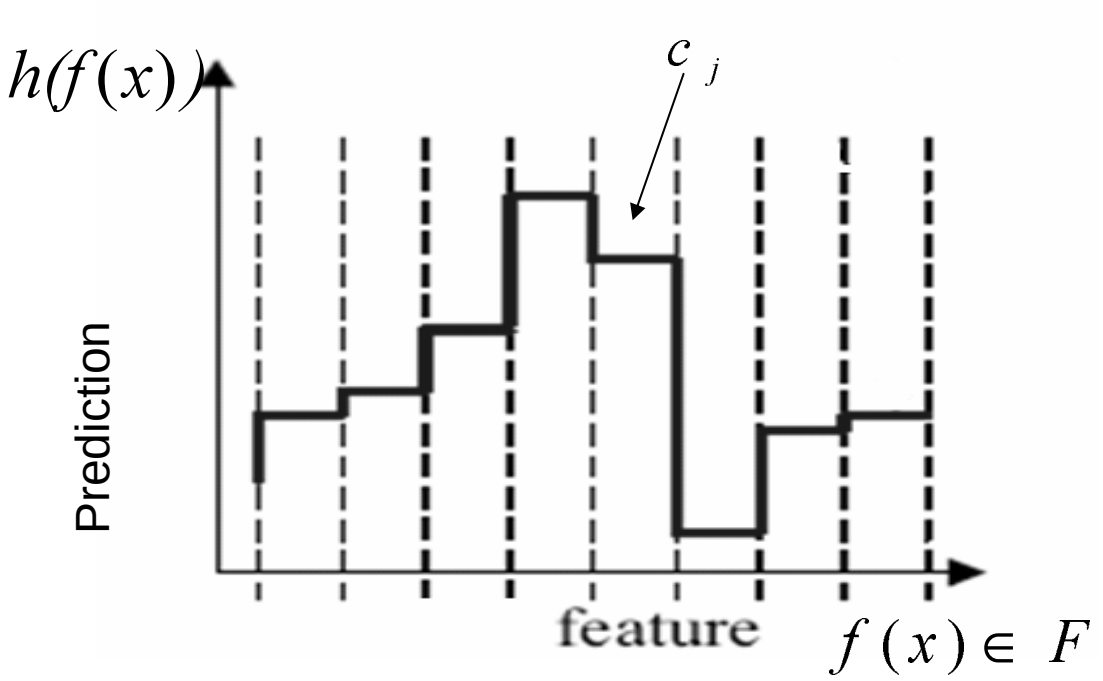
- Adaboost tell us how to design a strong classifier by running a base classifier
- Why not to use Adaboost to design the base classifier?
- i.e. design the base learner to directly minimize Z_t

[R. Shapire and Y. Singer 1999]

Adaboost:

Domain-partitioning

- The base classifier will make predictions based on a partition of a feature domain *F*
- The domain *F* will be partitioned in disjoint blocks, $F_1,...,F_n$ covering the whole domain
- To each block of the partition, a confidence and a class will assigned:



[R. Shapire and Y. Singer 1999]

Adaboost:

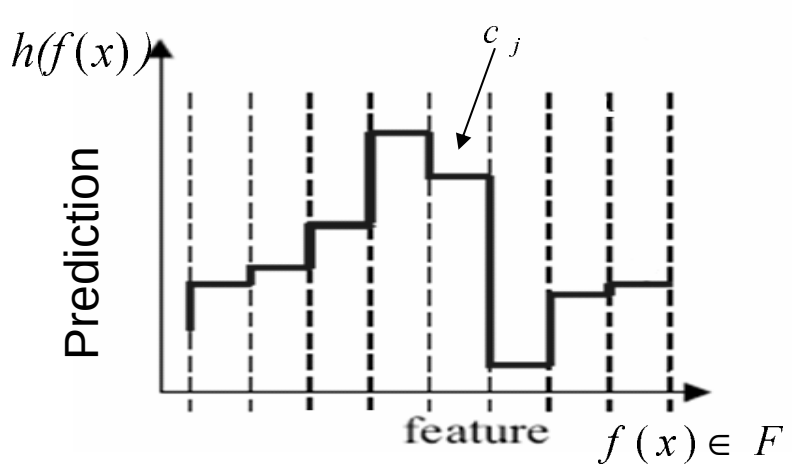
Domain-partitioning

- The base classifier will make predictions based on a partition of a feature domain *F*
- The domain *F* will be partitioned in disjoint blocks, $F_1,...,F_n$ covering the whole domain
- To each block of the partition, a confidence and a class will assigned:

$$c_j = h(f(x)), f(x) \in F_j$$

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

$$W_b^j = \sum_{i: f(x_i) \in F_j \wedge y_i = b} D(i)$$



- For a fast evaluation, the classifier is implemented through LUTs (Look Up Tables)

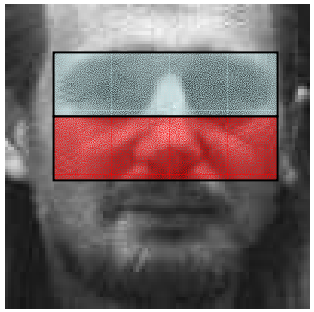
[R. Shapire and Y. Singer 1999]

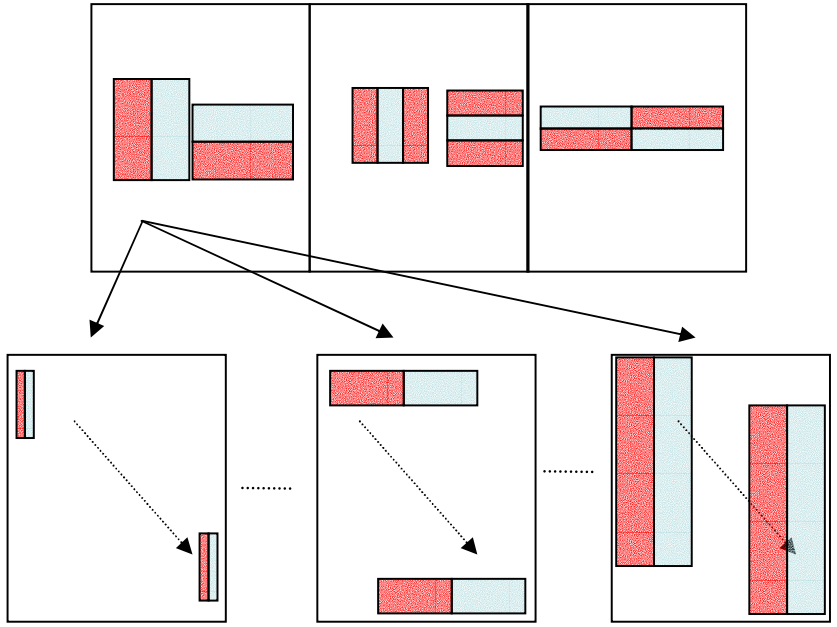
Features

- Processing speed:
  - The features should be fast to be evaluated (at least at the first layers of the cascade)
- Features selection :
  - With Adaboost the features (and weak-classifiers) can be selected at the same time during training
- How to partition the feature domain?

Features

- Rectangular Elements: a kind of haar wavelet filter (120.000 possibilities for windows of 24x24))

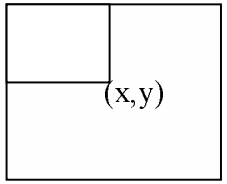


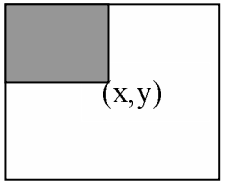


[Viola & Jones, 2001]

Features

- Rectangular features: Integral Image
  - The Integral Image is the accumulated sum of the image.

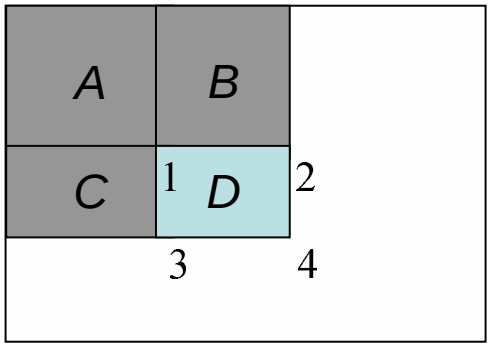
$I(x, y)$ 

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x', y')$$ 

  - The sum of any rectangle within the image can be quickly evaluated:

$$D = (A+B+C+D)+(A)-(A+B)-(A+C)$$

$$D = \quad II(4) \quad +II(1)- II(2) - \quad II(3)$$



Features

- Census Transform: (LBP or Texture Numbers)
  - To take a 3x3 Neighborhood and to compare the pixel in the middle against its neighbors

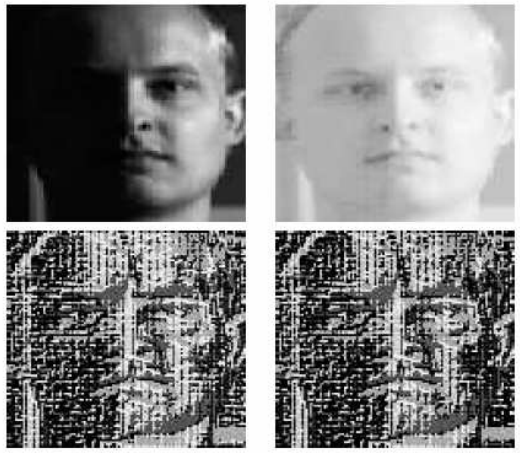
y_1	y_2	y_3
y_4	x	y_5
y_6	y_7	y_8

  - Example:

90	45	50
5	50	75
89	70	36

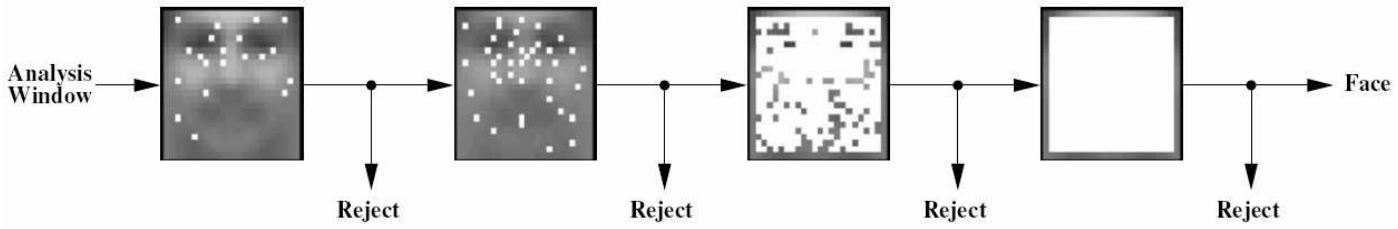
1	0	1
0	x	1
1	1	0

  - Feature value = 10101110 (8 bits)



[Fröba et al. 2004]

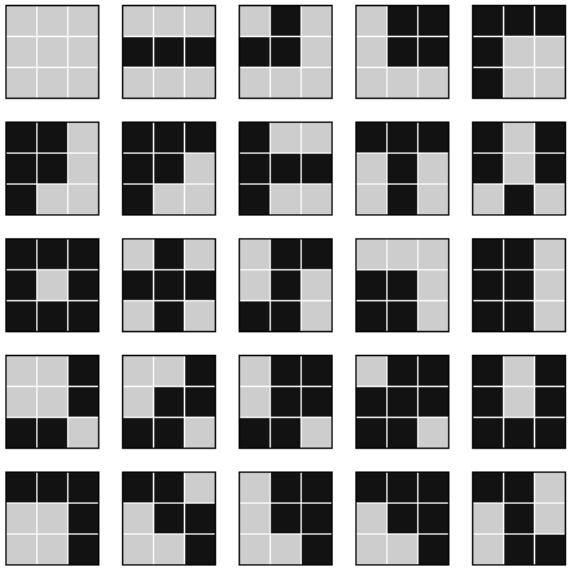
Features



- Modified Census Transform:
  (Formal definition)
  - Concatenation: $\bigotimes$
  - Comparison Function: $\zeta(\bar{\mathbf{I}}(\mathbf{x}), \mathbf{I}(\mathbf{y}))$.
  - Feature:
$$\Gamma(\mathbf{x}) = \bigotimes_{\mathbf{y}\in\mathcal{N}'} \zeta(\bar{\mathbf{I}}(\mathbf{x}), \mathbf{I}(\mathbf{y})).$$
  - 3x3 Neighborhood:

y_1	y_2	y_3
y_4	x	y_5
y_6	y_7	y_8

  - Census Transform: (8 bits)
$$\bar{I}(x) = I(x)$$
  - Modified Census Transform: (9 bits)
$$\bar{I}(x) = \frac{\sum_{y\in N'} I(y) + I(x)}{9}$$

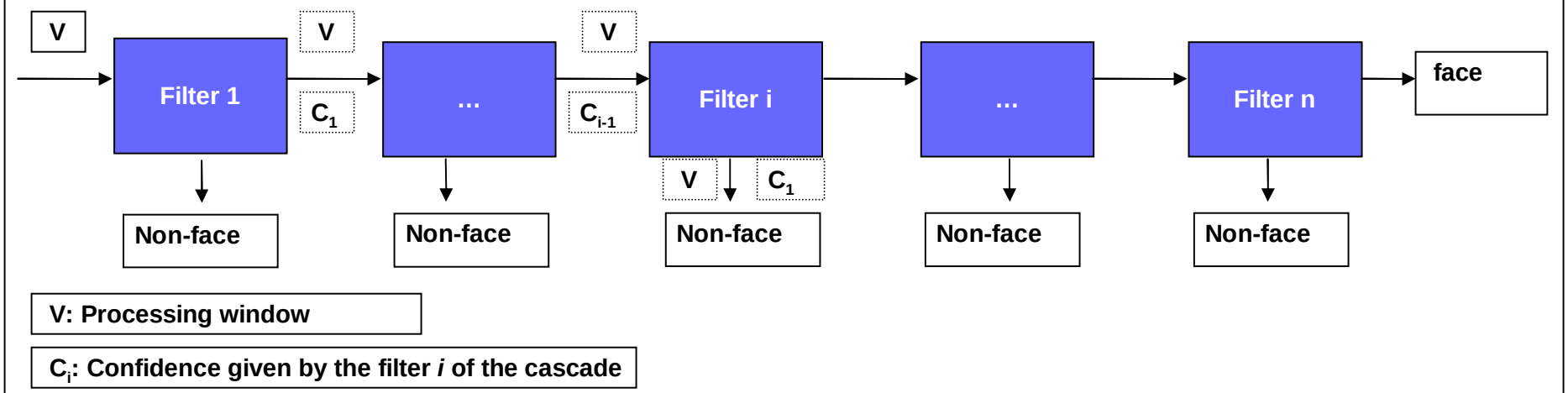


A randomly chosen subset of 25 out of $2^9 - 1$ possible Local Structure Kernels in a 3×3 neighborhood.

[Fröba et al. 2004]

Nested Cascade:
Concept

- To use the confidence of the output of a layer of the cascade as a part of the next layer of the cascade
- Objective:
  - To share information between layers of the cascade
  - To obtained a more compact and robust cascade.

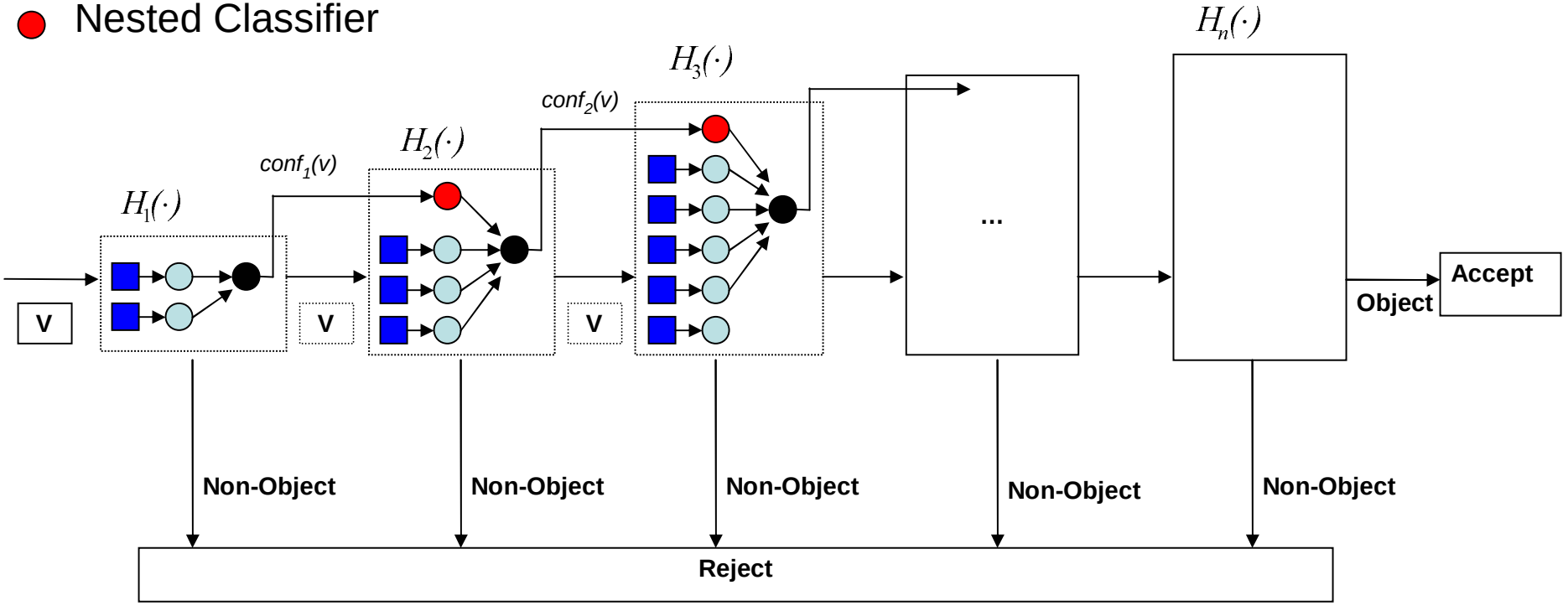


[Bo WU et al. 2004]

Nested Cascade and Adaboost:

Block Diagram

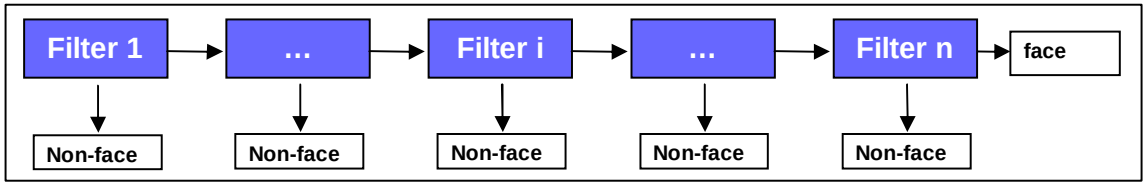
- Feature
- Weak Classifier
- Robust Classifier
- Nested Classifier



$$H_1(x) = \sum_{t=0}^{T_1 - 1} h_t^1(x) \qquad H_i(x) = \sum_{j=1}^{i-1} H_j(x) + \sum_{t=0}^{T_i - 1} h_t^i(x), i > 1$$

[Bo WU et al. 2004]

Nested Cascades and Adaboost: Training



f: maximum allowed of false positives per layer
d: minimum allowed detection rate per layer
F_{total}: final false positive rate
P: training set of faces
N: training set of non-faces

L_i : Layer i of the cascade
H: final classifier
Initialization: $F_{all} = 1.0$, $D_{all} = 1.0$, $ND = 0$, $i = 0$

While $F_{all} > F_{total}$

$i = i+1$

If $i > 1$

If $i = 1$

Update the weigths D_i using H_i

To Use Adaboost to add weak classifiers to H_i until $f_i \le f$, $d_i \ge d$

Train new layer

To Use Adaboost to train H_i *such that* $f_i \le f$, $d_i \ge d$

Update the detection and false positive rates
$F_{all} = F_{all} * f_i$, $D_{all} = D_{all} * d_i$

Get new rates

To add the new layer H_i to the cascade: $H = H\Theta H_i$

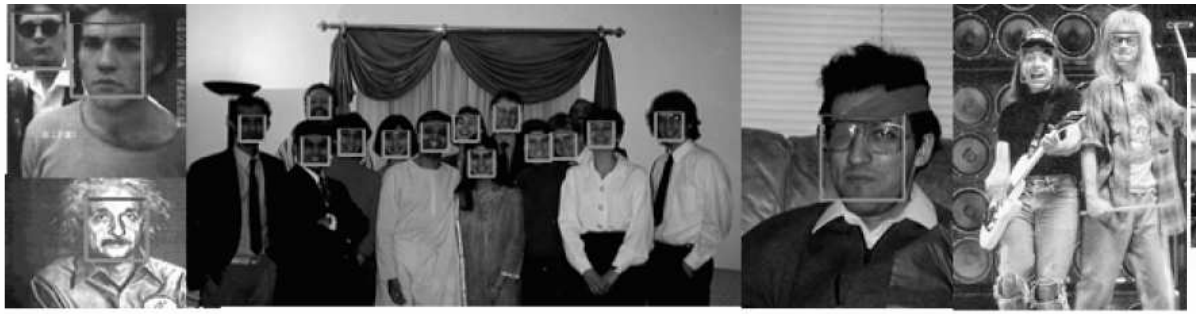
Add new layer to the cascade

To re-sample the set of non-faces *N* using *H*

Get non object training set

General Outline

- This tutorial has two parts
  - First Part:
    - Object detection problem
    - Statistical classifiers for object detection
    - Training issues
    - Classifiers Characterization
  - Second part:
    - Nested cascade classifiers
    - Adaboost for training nested cascades
    - Face analysis using cascade classifiers



Some frontal face detection results on the CMU+MIT frontal face test set.

Frontal Face Half Profile Face Full Profile Face

Some MVFD results on the CMU profile face test set.

Some Rotation Invariant MVFD results

Bo WU et al. 2004

Figure 2: Examples of rotation invariant face detection

Figure 3: Samples of detection results of faces of various poses

Figure 1: Detection results of occluded faces

Figure 4: Example of detecting different sized faces

Figure 5: Detection result of a face with changes of expression

Liu et al. 2004

Results

- Methods comparison

Face detector	CMU	MIT
Sung and Poggio		79.9% / 5
Rowley *et al.*	86.2% / 23	84.5% / 8
Viola and Jones	85.2% / 31	77.8% / 5
Li *et al.*	90.2% / 31	
Féraud *et al.*	86.0% / 8	
Garcia and Delakis	90.3% / 8	90.1% / 7

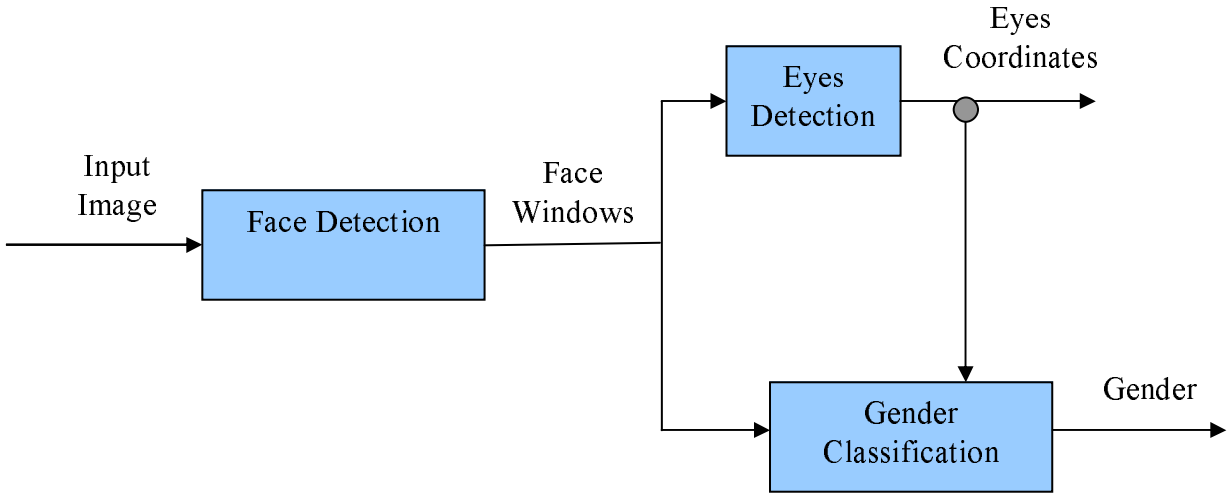
CMU: 130 images, 507 faces
MIT: subset of CMU (23 images, 57 Faces)

Table 1. Frontal face detection results on CMU+MIT frontal face test set (130 images, 507 faces)

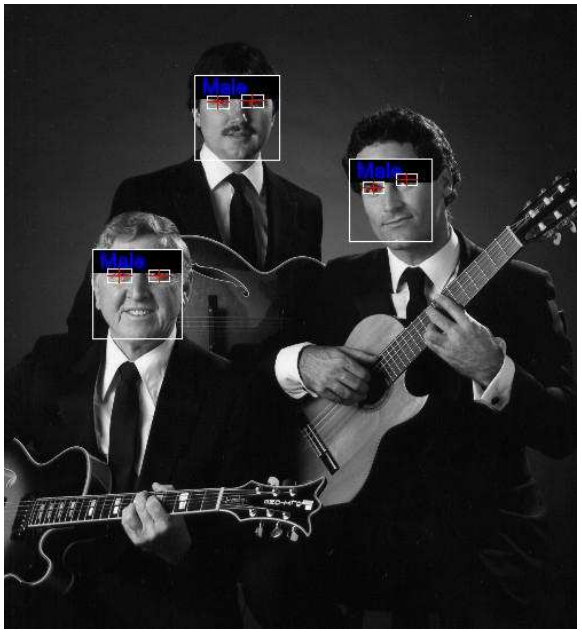
False Alarm / Method	3	10	13	31	50	57	95	213	422
Bo WU	89.0%	90.1%	90.7%	-	-	94.5%	-	96.5%	-
Viola-Jones	-	78.3%	-	85.2%	88.8%	-	90.8%	-	93.7%
Rowley	-	83.2%	-	86.0%	-	-	89.2%	-	89.9%
Ours	77.3		86.6	90.1		92.1			

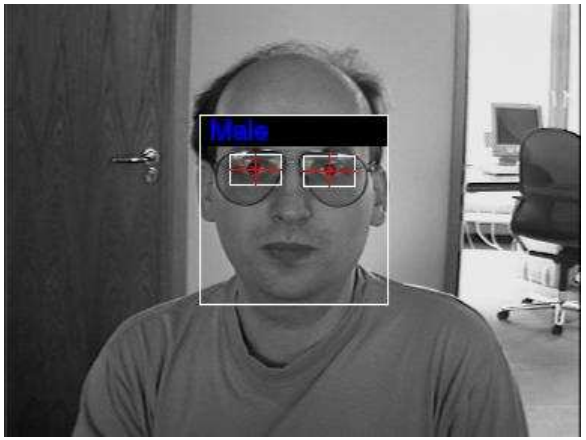
Face detection and Classification

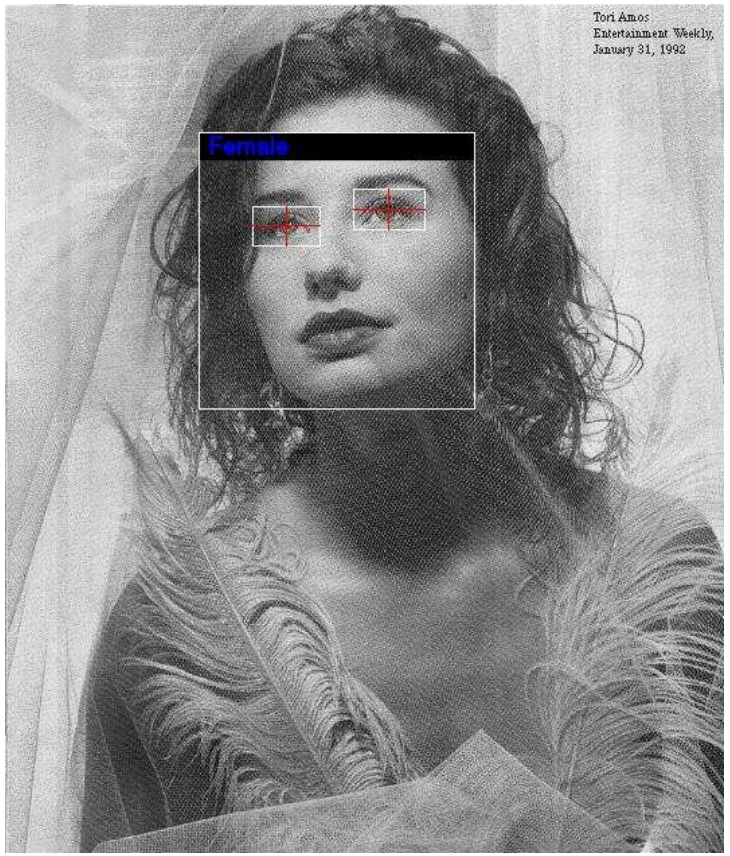
- Face detection
- Eye detection
- Gender classification
- Race, age classification,…



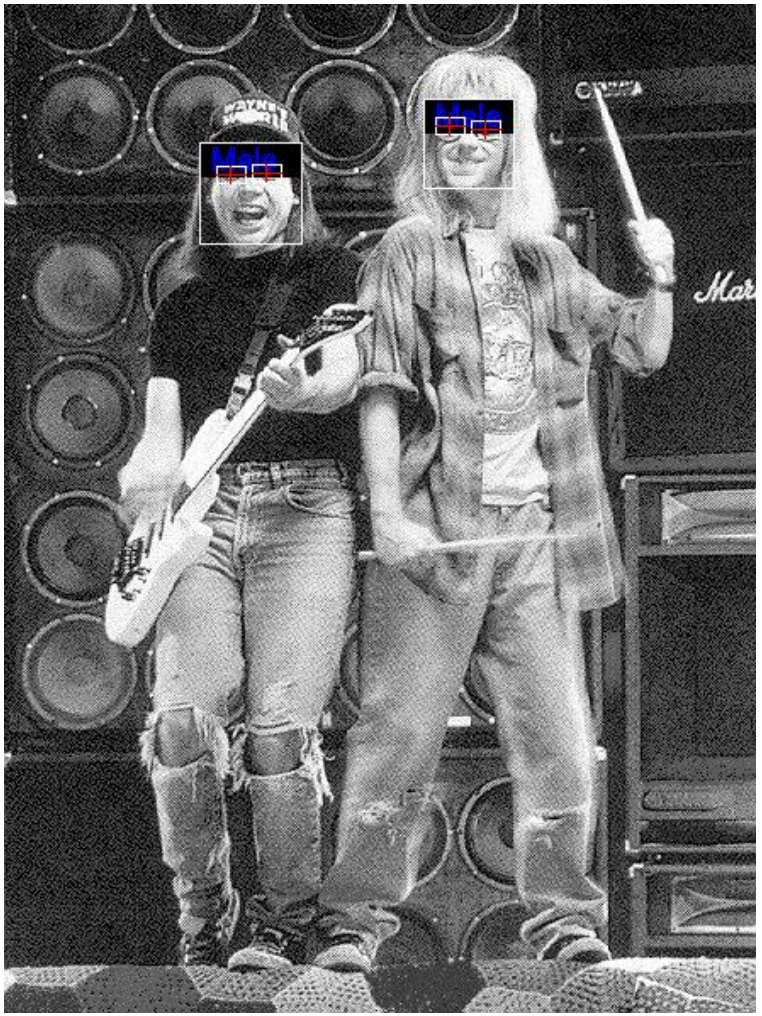
Results


Male
Male
Male


Male


Tori Amos
Entertainment Weekly,
January 31, 1992
Female

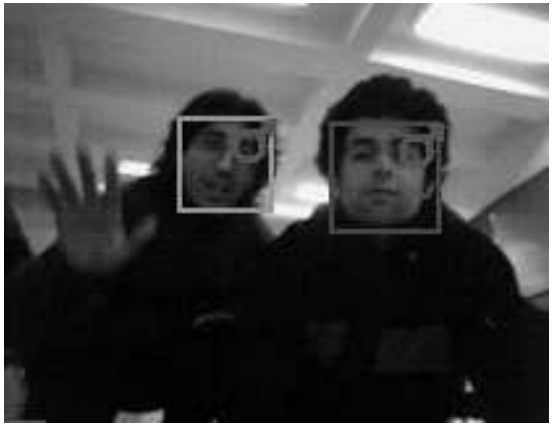
Results

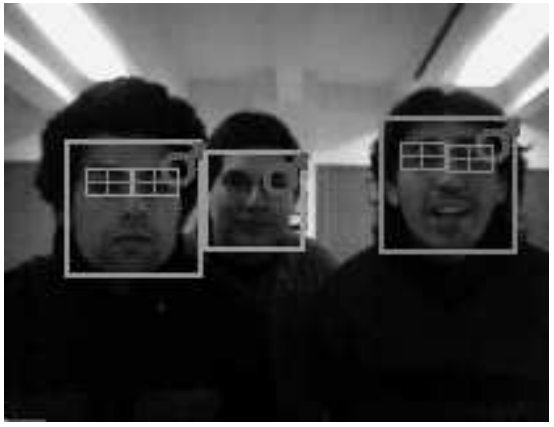


Results

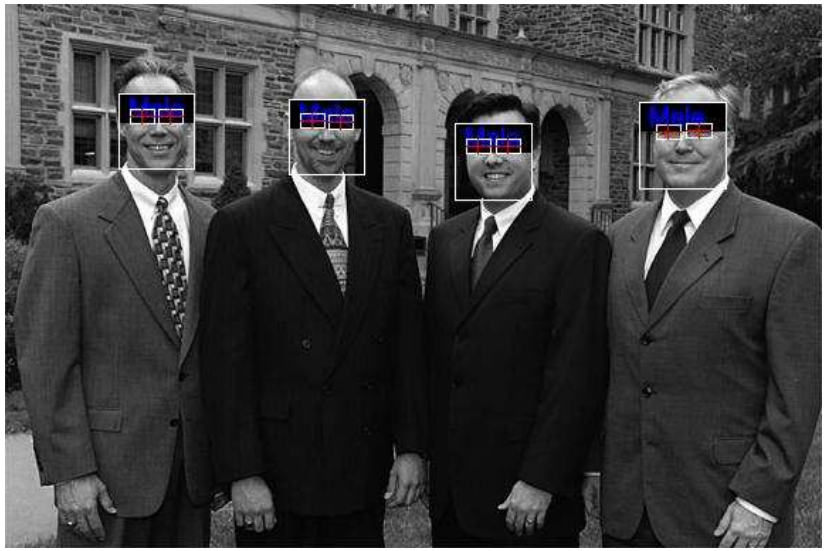
(Aibo Robots)







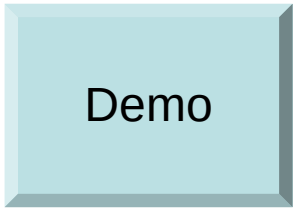

Results



- Rates:
  - ~90% face detection / ~0.2 FP per image (CMU-MIT)
  - ~80% correct gender recognition
  - ~99% eye detection with d_error < 5% d_eyes

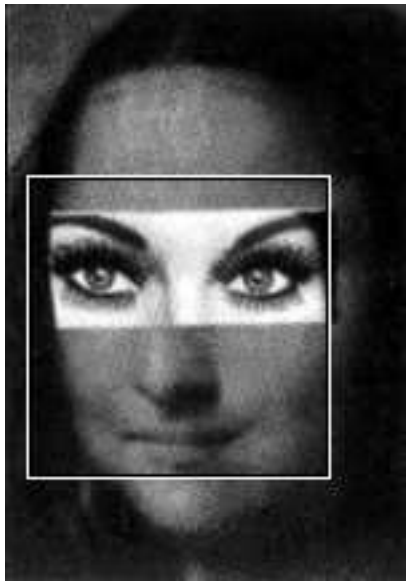
Demo

- Nested Cascade
  - Trained using Adaboost (domain-partitioning).
- Training
  - Training set:
    - ~5000 frontal faces
    - 3200 non-faces, 5946 images (bootstrapping on 1.327.068.822 windows for the last layer of the cascade)
  - Training time: 12 hours.
- Processing time (for detecting multiple-faces)
  - About 200 ms for an 320x240 pixel image (≈ 5 frames per second)
  - About 1000 ms for an 640x480 pixel image (≈ 1 frames per second)
- Performance:
  - CMU-MIT db:
    - Detection Rate: 90%
    - 0.20 False positives per image (~1.3×10^{-6} false positive rate)
  - Localization Rate (When only one face is searched): ~95-99%
- Representation
  - Features base on rectangular features (first layers) y mLBP (last layers).
- Heuristic for accelerating the search:
  - Hierarchical grid using a coarse-to-fine procedure in image space



Summary

- The following concepts were presented:
  - Statistical learning:
    - Introduction
    - Examples (Bayes, SVM, Adaboost)
  - Training/testing Issues
    - How to obtain a representative training set for the non-object: Bootstrapping
    - How to evaluate a classifier: ROCs.
  - Cascade classifiers
    - How to take advantage of the asymmetry on the a priori probability of appearance of the classes: Cascade Classifier
    - How to re-use information of previous layers on a cascade: Nested Cascade
  - Adaboost:
    - How to improve and combine classifiers: Adaboost
    - Why adaboost works: margin and consistency
    - How to design classifiers using Adaboost: Domain-Partitioning



End

Main References

- Ming-Hsuan Yang, David Kriegman, and Narendra Ahuja, *Detecting Faces in Images: A Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pp. 34-58, 2002
- Yoav Freund and Robert Schapire, *A short introduction to boosting*, http://www.boosting.org
- Robert Shapire, Yoram Singer, Improved Boosting Algorithms Using Confidence-rated Predictions, Machine Learinig, 37(3):297-336, 1999
- Paul Viola, Michael Jones, *Fast and Robust Classification using and Asymetric AdaBoost and a Detector Cascade*, Neural Information Systems 14, December 2001
- Bo WU, Haizhou AI, Chang HUANG and Shihong LAO, Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboost. *Int. Conf. on Face and Gesture Recognition – FG 2004*, 79 – 84, Seoul, Korea, May 2004, IEEE Press.
- Bernhard Fröba, Andreas Ernst, *Face Detection with the Modified Census Transform*, Int. Conf. on Face and Gesture Recognition – FG 2004, 91 - 96, Seoul, Korea, May 2004, IEEE Press.
- Ce Liu, Hueng-Yeung Shum, Kullback-Leibler Boosting, CVPR 2003.
- M. Delakis & and C. Garcia, *Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection*, IEEE transactions on pattern analysis and machine intelligence, 1408 - 1423 , vol. 26, 11, November 2004
- Lin-Lin Huang, Akinobu Shimizu, Hidefumi Kobatake, *Classification-Based Face Detection Using Gabor Filter Features*, Int. Conf. on Face and Gesture Recognition – FG 2004, Seoul, Korea, May 2004, IEEE Press.
- Yen-Yu Lin, Tyng-Luh Liu and Chiou-Shann Fuh, *Fast Object Detection with Occlusion*, ECCV 2004.

Summary

- Robust under:
  - Varying pose:
    - Wavelet-Bayesian (Schneiderman & Kanade, 98)
    - Multiple Nested-cascades (Bo WU et al., 2004)
  - In-plane Rotation:
    - NN for rotation estimation and NN (Rowley & Kanade, 98)
    - Multiple Nested-cascades (Bo WU et al., 2004)
    - “Tree”-cascade (S.Z. Li et al., 2004)
    - Boosted tree for rotation estimation and multiple cascades (Viola & Jones, 2003)
  - Occlusion:
    - Cascade trained with occluded examples (Yen-Yu Lin et al., 2004)
    - Cascade with Modified census transform based features (Bernhard Fröba et al., 2004)
  - Expression:
    - CFF (Garcia & Delakis, 2004)

Web Resources

- Google
- Robert Frischholz, Face detection homepage:
  http://home.t-online.de/home/Robert.Frischholz/face.htm
- CMU Robotics Institute, Face Group Homepage
  http://www.ri.cmu.edu/labs/lab_51.html
- MIT CBCL face dataset
  http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html
- Ming-Hsuan Yang, Resources for Face Detection
  http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html
- Online demos:
  - Garcia & delakis: http://www.csd.uoc.gr/~cgarcia/FaceDetectDemo.html
  - Schneiderman & Kanade: http://www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi
  - WebFaces: http://www.cwr.cl/webfaces

Performance Evaluation

- Benchmark datasets:
  - Standard (de facto):
    - CMU test set (Rowley et al, Sung & Poggio): 507 faces, 130 images
  - Others:
    - Subsets of CMU
    - BioID Face DB (HumanScan): 1521 faces, 1521 images (23 persons)
    - Web (Garcia & Delakis): 499 faces, 125 images
    - Cinema (Garcia & Delakis): 276 faces ,162 images
    - XMSVTS (Kesser et al.): 2360 faces, 2360 images
  - Profile:
    - CMU Test set II (Schneiderman): Frontal and non Frontal
  - In-plane rotation:
    - CMU rotated set (Rowley)

Performance Evaluation

- Training datasets:
  - Rowley:
    - 1500 Faces (full images).
  - Viola & Jones: 24x24 pixels
    - 4916 Faces, 8000 Non-faces.
  - MIT (CBCL FACE DATABASE #1): 19x19 pixels
    - Training set 2,429 faces, 4,548 non-faces
    - Test set: 472 faces, 23,573 non-faces
  - Face Recognition datasets: is it fare to used them?
    - Yale
    - Feret
    - …
  - …

M. Delakis & and C. Garcia 2004