Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 MMIS First Semester 2010/2011

Lesson 8 8 December 2010

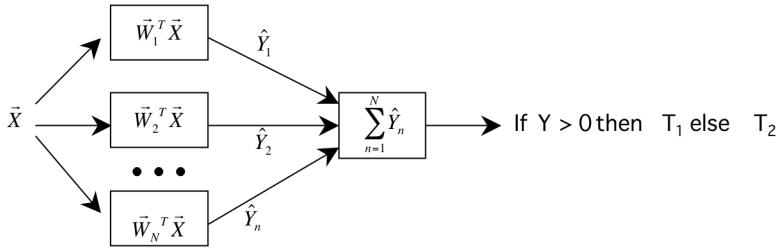
Detection using a Cascade of Boosted Classifiers

Outline:

A Committee of Boosted Classifiers

One of the more original ideas in machine learning the last decade is the discovery of a method by to learn a committee of classifiers by boosting. A boosted committee of classifiers can be made arbitrarily good: Adding a new classifier always improves performance.

A committee of classifiers decides by voting.



A feature vector is determined to be in the target class if the majority of classifiers vote > 0.

$$\text{if } \sum_{i=1}^{I}\vec{W}_i^T\cdot\vec{X}_m > 0 \text{ then } \hat{\omega}_1 \text{ else } \hat{\omega}_2$$

To learn a boosted committee we iteratively add new classifiers to the committee.
In each cycle we change the data set and learn a new classifier, W_i

The data set will be changed by giving additional weight to improperly classified samples. We learn the next class by multiplying the Y labels a weight vector, A_i.

$$\vec{W}_i = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\vec{A}_i\cdot\vec{Y})$$

Learning a Committee of Classifiers with Boosting

We can iteratively apply the above procedure to learn a committee of classifiers using boosting. For this we will create a vector of "weights" a_m for each training sample. Initially, all the weights are 1.

After each new classifier is added, we recalculate the weights to give more weight to improperly classified training samples.

As we add classifiers, whenever a sample is misclassified by the committee we will increase its weight so that it carries more weight in the next classifier added.

Recall the committee vote if $\sum_{i=1}^{I}(\vec{W}_i^T \vec{X}_m) > 0$ then Class 1 (positive detection).

For m = 1 to M: if $(y_m \cdot \sum_{i=1}^{I}(\vec{W}_i^T \vec{X}_m)) < 0$ then $a_m = a_m + 1$

The result is the $(i+1)^{th}$ weight vector A_{i+1}

We then learn the $i+1^{th}$ classifier from the re-weighted set by

$$\vec{W}_{i+1} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T (\vec{A}_{i+1} \cdot \vec{Y})$$

ROC Curve

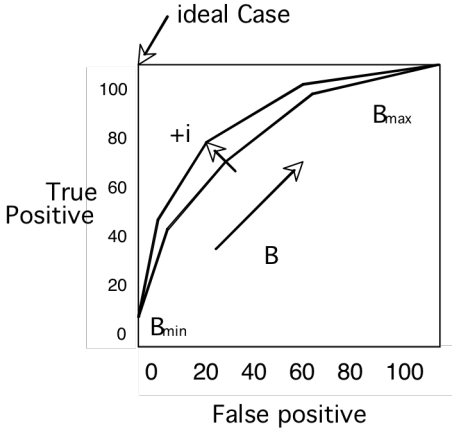
As we saw in lesson 19, The ROC describes the True Positives (TP) and False Positives (FP) for a classifier as a function of the global bias B.

For m = 1 to M:

if $\sum_{i=1}^{I}(\vec{W}_i^T \vec{X}_m) + B > 0$ and $y_m > 0$ then TP=TP+1

if $\sum_{i=1}^{I}(\vec{W}_i^T \vec{X}_m) + B > 0$ and $y_m < 0$ then FP=FP+1

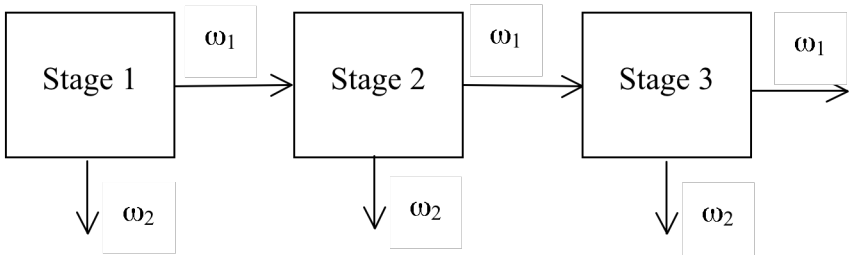
The Boosting theorem states that adding a new boosted classifier to a committee always improves the committee's ROC curve. We can continue adding classifiers until we obtain a desired rate of false positives and false negatives.



However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end with a very very large number of classifiers.

Learning a Multi-Stage Cascade of Classifiers

We can optimize the computation time by using a multi-stage cascade.
With a multi-stage classifiers, only events labeled as positive are passed to the next stage.



Each stage is applied with a bias, so as to minimize False negatives.

Stages are organized so that each committee is successively more costly and more discriminant.

Assume a set of M training samples $\{X_m\}$ with labels $\{y_m\}$.
Set a desired error rate for each stage j : (FP_j, FN_j).

For each stage, j, train the j+1 stage with all positive samples from the previous stage.

Each stage acts as a filter, rejecting a grand number of easy cases, and passing the hard cases to the next stage. The stages become progressively more expensive, but are used progressively less often. Globally the computation cost decreases dramatically.

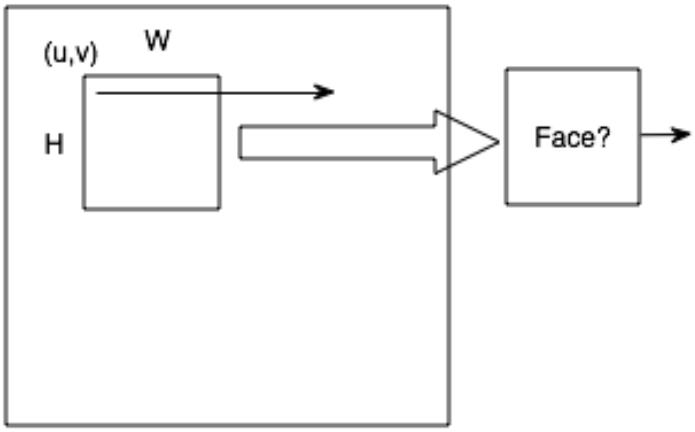
Boosted Linear Classifiers for Face Detection in Images

The Detection Process

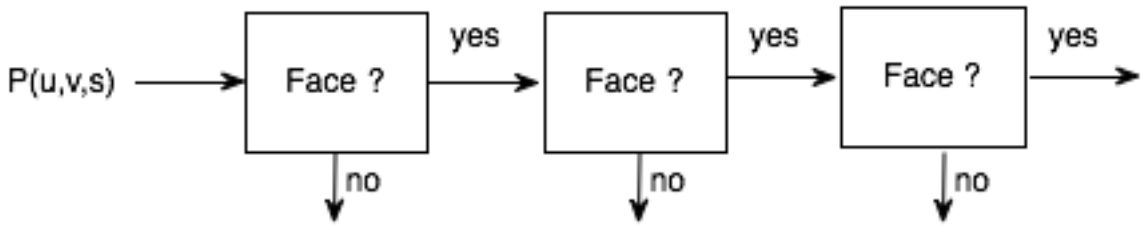
A Cascade of Classifiers detects faces with a window scanning approach.

An image window, P(i, j; u, v, s) is an (sW x sH) pixel window with its upper left corner at pixel (u, v). During the detection proces will note this as P(u,v,s), leaving (i,j) as implicit. Training will us a collection of M labeled windows X_m=P(u,v,s).
Of course, each X_m is a 2D signal X_m(i,j).

The window is texture mapped (transformed) so that it fits into a standard size window of a given size (W, H). For faces, the window size is typically approximately (24, 24) pixels.



The decision of whether the window P(u,v,s) contains a face is provided by a cascade of boosted linear classifiers.



The algorithm requires a large number of local "features" to classify the window.

They can also be provided by Haar wavelets computed using a Difference of Boxes as shown below.

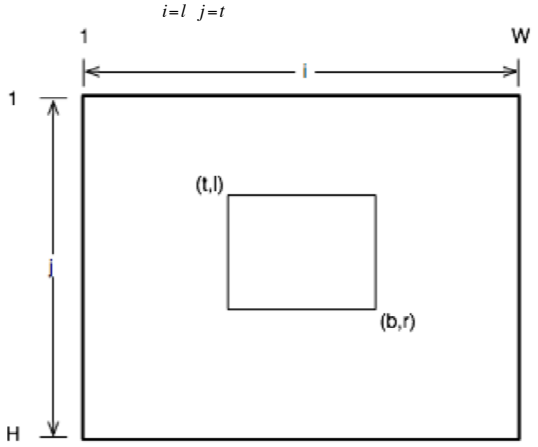
Image Description with Difference of Boxes

An image rectangle is defined by the top-left and bottom-right corner.
This may be represented by a vector (t, l, b, r). The sum of pixels in a rectangle defines a box feature.

Box Features

A box feature is the sum of pixels over a rectangle from top (t) to left (l) and bottom (b) to right (r), with the constraints : top < bottom and right > left.

$$b(t,l,b,r) = \sum_{i=l}^{r}\sum_{j=t}^{b} p(i,j)$$



(Window of image at P(u,v,s).

For a window of size WxH there are $N=W^2H^2/4$ possible boxes.

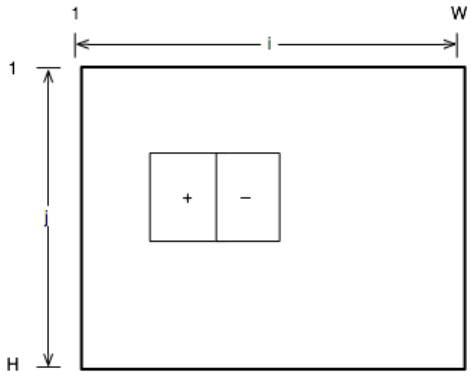
$$N = \frac{W^2}{2} \cdot \frac{H^2}{2}$$

Difference of Boxes

A first order Difference of Boxes (DoB) feature is a difference of two boxes $box(t1,l1,b1,r1)$.

$$DoB(t1,l1,b1,r1,t2,l2,b2,r2) = box(t1,l1,b1,r1) - box(t2,l2,b2,r2)$$

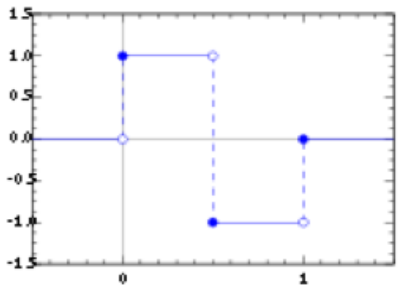
There are N^2 possible 1st order (2 box) DoB features in an image
There are N^3 possible 2nd order (3 box) DoB features in an image.
Not all DoBs are useful.



An interesting subclass are Difference of Adjacent Boxes where the sum of pixels is 0. These are Haar wavelets.

Haar Wavelets:

The Haar wavelet is a difference of rectangular Windows.



$$h(t) = \begin{cases} 1 & \text{for } 0 \le t < 0.5 \\ -1 & \text{for } 0.5 \le t < 1 \\ 0 & \text{for } t < 0 \text{ and } t \ge 1 \end{cases}$$

The Haar wavelet may be shifted by d and scaled by s

$$h(t;s,d) = h(t/s - d)$$

Note that the Haar Wavelet is zero gain (zero sum).

$$G = \int_{-\infty}^{\infty} h(t)dt = 0$$

The Digital (discrete sampled) form of Haar wavelet is

$$h(n;d,k) = \begin{cases} 1 & \text{for d} \le n < d + k/2 \\ -1 & \text{for d} + \text{k}/2 \le n < d + k \\ 0 & \text{for } n < d \text{ and } n \ge d + k \end{cases}$$

Haar wavelets can be used to define an orthogonal transform analogous to the Fourier basis.

Haar Functions, and the Walsh-Hadamard transform have been used in Functional Analysis and signal processing for nearly a century.

In the 1980s the Wavelet community re-baptized the Haar functions as "wavelets" and demonstrated that the Walsh-Hadamard transform is the simplest form of wavelet transform.

A 2-D form of Walsh-Hadamard transform may be defined using DoB features.
These can be calculated VERY fast using an algorithm known as Integral Images.

Fast 2D Haar Wavelets using Integral Image

Integral Images.

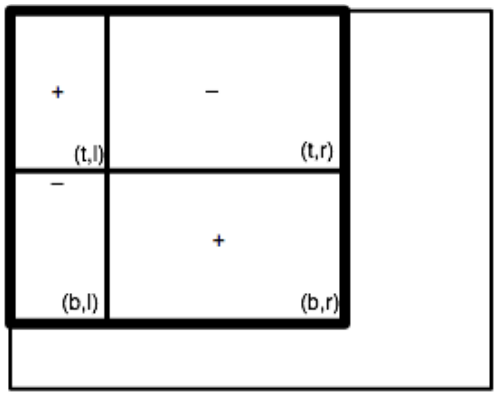
An integral image is an image where each pixel contains the sum from the upper left corner.

$$ii(u,v) = \sum_{i=1}^{u}\sum_{j=1}^{v} p(i,j)$$

In many uses, p(i,j) is a window extracted from a larger image at (u,v,s).
An integral image provides a structure for very fast computation of 2D Haar wavelets.

Any box feature can be computed with 4 operations (additions/subtractions).

box(t,l,b,r)=ii(b,r)–ii(t,r)–ii(b,l)+ii(t,l)

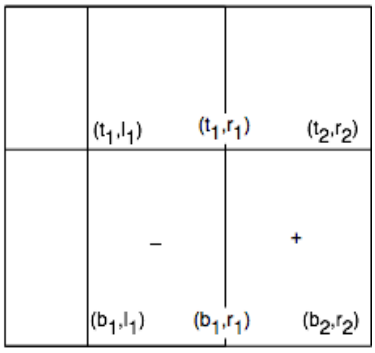


An arbitrary 1st order difference of boxes costs 8 ops.

$$\begin{aligned} DoB(t_1,l_1,b_1,r_1,t_2,l_2,b_2,r_2) = \; & ii(b_1,r_1)–ii(t_1,r_1)–ii(b_1,l_1)+ii(t_1,l_1) \\ & – ii(b_2,r_2)–ii(t_2,r_2)–ii(b_2,l_2)+ii(t_2,l_2) \end{aligned}$$

However, a 1st order Haar wavelet costs only 6 ops because $r_1=l_2$ and thus

$ii(t_1,r_1) = ii(t_2,l_2)$ and $ii(b_1,r_1)= ii(b_2,l_2)$



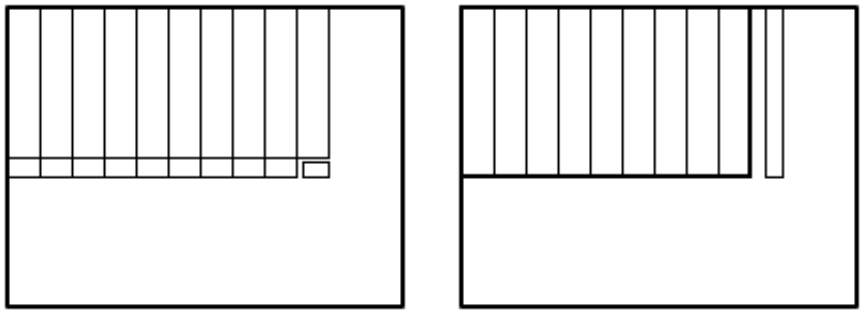
$$Haar(t_1,l_1,b_1,r_1,b_2,r_2) = ii(b_2,r_2)–2ii(b_1,r_1)+ii(b_1,l_1)–ii(t_2,r_2)+2ii(t_1,r_1)–ii(t_1,l_1)$$

Fast Integral Image algorithm.

Integral images have been used for decades to compute local energy for normalization of images. The fast algorithm involves a row buffer that contains the sum of each row

For j = 1 to I
For i=1 to J
    r(i) := r(i) + p(i,j)
    ii(i,j) = ii(i-1,j)+r(i)

Cost = 2IJ ops.



In 2001, Paul Viola and Mike Jones at MERL (Misubishi Research Labs) showed that Haar wavelets could be used for real time face detection using a cascade of linear classifiers.

They computed the Haar Wavelets for a window from integral images.

Linear Classifiers for Face Detection

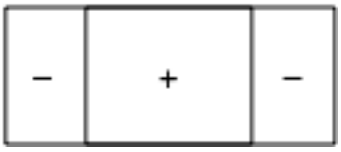
The innovation in the Viola-Jones face detector resulted from

1) A very large number of very simple features (Haar wavelets).
2) The use of the Ada boost algorithm to learn an arbitrarily good detector.

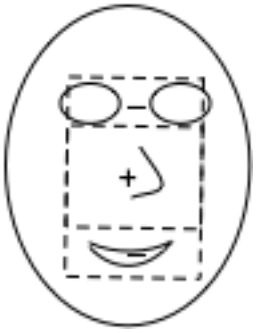
HAAR wavelets are computed using difference of Boxes, with Integral Images.

A WxH image contains $N = W^2H^2/4$ possible 1st order Haar wavelets.
(difference of adjacent boxes of same size).

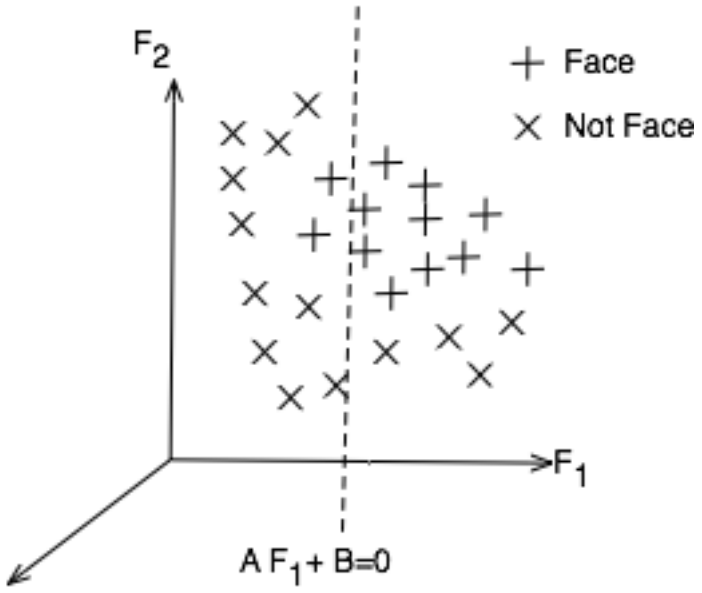
Similarly, any 2nd Haar wavelet can be computed with 8 ops.



A WxH window contains $W^2H^2/8$ possible 1st and 2nd order Haar wavelets.
These provide a space of $N = W^2W^2/8$ features for detecting Faces.
Each feature, F_n is defined as a difference of boxes.



For a give position (u,v) and scale (s) any window W(u, v, s) that contains a face is a point "+" in this very N-dimensional space.



Each feature can be used to define a hyper-plane $\vec{F}_n^T \cdot \vec{X} + b_n = 0$

For 2D features, the inner product is

$$F_n^T \cdot \vec{X} = \langle F_n, P(i,j) \rangle = \sum_{i=1}^{W} \sum_{j=1}^{H} F_n(i,j) \cdot P(i,j)$$

Thus our hyperplane is simply the difference of adjacent boxes (Haar Wavelet) n plus a bias:

$$\langle F_n, P(i,j) \rangle + b_n = 0$$

The problem is to choose b_n so that most non-face windows are on one side of the hyperplane and most face windows are on the other.

Training a single classifier:

Assume a very large training set of M labeled face windows $\{\vec{X}_m\}$ that have been labeled by a set of labels $\{y_m\}$ such that y=+1 if face and y=–1 if not face.
Each $\vec{X}_m = P_m(u,v)$

For a given Haar wavelet, *n*: Compute $b_n = -E\{(\vec{F}_n^T \cdot \vec{X}_m) \cdot y_m\} = 0$

Note that for a True Positive or True Negative $(\vec{F}_n^T \cdot \vec{X}) \cdot y_m + b_n \geq 0$

Then for any window, *X(i,j)*=P(i,j; u,v,s), we can classify it as Face or Not face using:

$(\vec{F}_n^T \cdot \vec{X}) \cdot y_m + b_n \geq 0$ then Face else not Face.

Thus each DoB feature, *n*, is a linear classifier.

For the training set $\{X_m\}$, the error rate for the feature F_n is

$$E_n = \underset{m}{Card}\{(F_n^T \cdot \vec{X}_m) \cdot y_m + b_n < 0\}$$

(add 1 for each FP and FN.)

For a set $\{(X_m, y_m)\}$, $\{(F_n, b_n)\}$ the best classifier is minimum error rate.

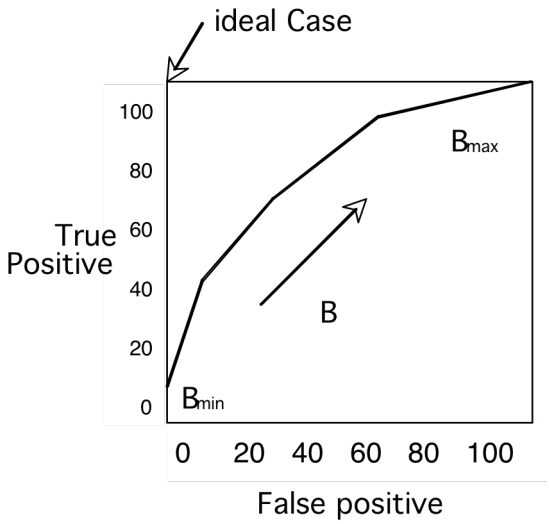
$\text{Arg-Min}_n\{E_n\}$

We want to learn the set of *i* best classifiers.
We can improve learning with Boosting.

Note that there are two parts to the errors:
False Positives (FP) and False Negatives (FN)

We can trade FPs for FNs by adding a Bias B,

$(\vec{F}_n^T \cdot \vec{X}) \cdot y_m + b_n + B \geq 0$ then Face else not Face



Boosted Learning

To boost the learning, after selection of each "best" classifier, (F_n, B_n)
we re-weight the labels y_n to increase the weight of incorrectly classed windows:

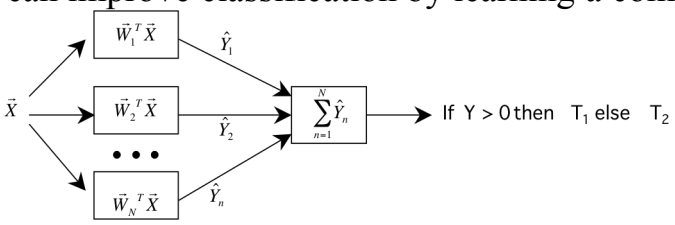
For all m = 1 to M if $(\langle X_m, F_n \rangle + b_n) \cdot y_m^{(i-1)} < 0$ then $y_m^{(i)} = 2\, y_m^{(i-1)}$

We then learn the i^{th} classifier from the re-weighted set

$E_{min} = M$
For n=1 to N do
$B_m = E\{ \langle X_m, F_n \rangle \cdot y_m^{(i)}\}$
$E_n = Card\{(\langle X_m, F_n \rangle + b_n) \cdot y_m < 0\}$
if $E_n < E_{min}$ then $E_{min} := E_n$

Learning a Committee of Classifiers with Boosting

We can improve classification by learning a committee of the best I classifiers.



The decision is made by voting. A window X(i,j)=P(i, j; u,v,s) is determined to be a Face if the majority of classifiers vote ≥ 0.

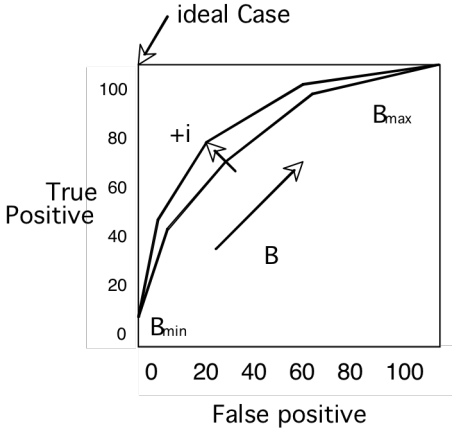
If $\sum_{i=1}^{I}\vec{F}_i^T\cdot\vec{X}+b_i\geq 0$ then Face else Not-Face.

ROC Curve

We can describe a committee of classifiers with an ROC curve, but defining a global bias, B. The ROC describes the number of False Positives (FP) and False Negatives (FN) for a set of classifier as a function of the global bias B.

FP = Card{(<X_m, F_n> + b_n+B) > 0 and $y_m = -1$}
FN = Card{(<X_m, F_n> + b_n+B) < 0 and $y_m = +1$}

The Boosting theorem states that adding a new boosted classifier to a committee always improves the committee ROC curve. We can continue adding classifiers until we obtain a desired rate of false positives and false negatives.



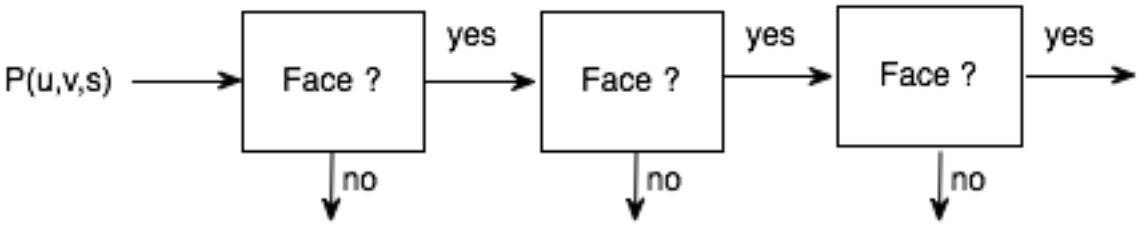
Learning a Multi-Stage Cascade of Classifiers

We can optimize the computation time by using a multistage cascade.

Algorithm:

1) Set a desired error rate for each stage j : (FP_j, FN_j).
2) For j = 1 to J

For all windows labeled as Face by j-1 stage, learn a boosted committee of classifiers that meets (FP_j, FN_j).



Each stage acts as a filter, rejecting a grand number of easy cases, and passing the hard cases to the next stage.