Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS — Fall Semester 2017
Lessons 4 — 22 November 2017

Face Detection using the Viola Jones Face Detector

Outline

Notation

$\{W_m\}$	Training set of M windows (imagettes) for learning.
$\{y_m\}$	Indicator variable for each training window. $y_m \in \{0,1\}$
	$y_m=1$ if the window contains a face and otherwise $y_m=0$
M	The number of training samples.

$X_n = \langle W, H_n \rangle = \sum_{x=1}^{C} \sum_{y=1}^{R} W(x,y) H_n(x,y)$ A Haar-like feature applied to the window, W

$h_n(W) = \begin{cases} 1 & if\, p_n(X_n + b_n) > 0 \\ 0 & otherwise \end{cases}$ a weak classifier (hypothesis) for the feature X_n.

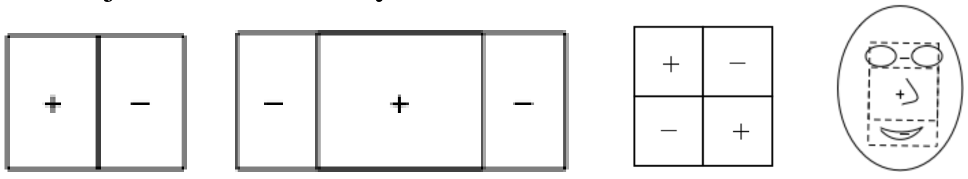
p_n	Polarity (Sign) of the for the classifier $h_n(W)$ $p_n \in \{-1,1\}$
b_n	Constant for the classifier $h_n(W)$. Acts as a threshold for $p_n X_n$

$h(W) = \sum_{t=1}^{T} \alpha_t h_t(W)$ A committee of T weak classifiers.

$E_T = \frac{1}{M} \sum_{m=1}^{M} w_m \left| h(W_m) - y_m \right|$ The error rate for the committee for a data set $\{W_m\}$

1. The Viola Jones Face Detector

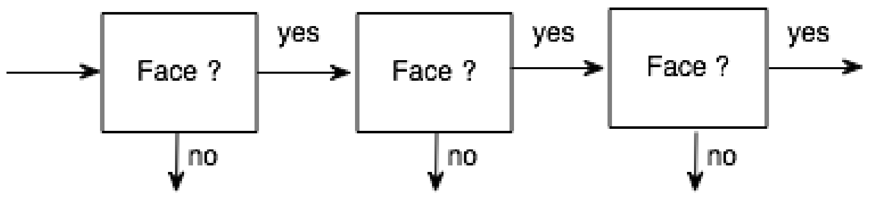
In 2001, Paul Viola and Mike Jones at MERL (Misubishi Research Labs) demonstrated a revolutionary new technique to detect faces in images. Their technique used a very large number of very simple features computed with difference of adjacent boxes. Each window was described with a set of features computed from Difference of Adjacent Boxes. They used 2-box, 3-box and 4 box features:



Box features are sums of pixels over rectangular regions and can be computed with a very fast algorithm known as “integral images”. Viola and Jones referred to these as “Haar-like” features because they are similar to the Haar Transform, a form of binary Discrete Fourier transform used in signal processing. There are gave a VERY large number of possible difference-of-box features.

Each feature provided a simple weak classifier. Each difference-of-box feature defines a weak linear classifier for whether the window contained a face. Some weak classifiers are better than others. Viola and Jones used boosted learning to learn a strong classifier as a committee of weak classifier. Each weak classifier produced a yes/no vote as to whether the window contained a face. A weighted sum of the votes determined whether the committee decided Face or Not face.

Viola and Jones learned a sequence of committees for face detection, where each committee was trained on only those windows that were selected as “face” by the previous committee. They applied this technique with a brute-force “scanning window” approach in which rectangular windows of a given size are independently classified as “Face” or “Not Face”.

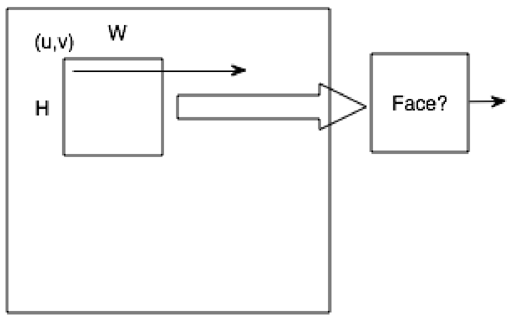


This process took around 3 months to train, but once trained provided revolutionary gains in the ability to detect faces in images. The learned process was patented by MERL, and published in OpenCV, allowing the community to verify the effectiveness of the process and to use it for proto-typing applications.

The Viola Jones face detector was trained with windows of size (24, 24) pixels.

Note that it is possible to use an affine transform to map a rectangular region of the image of any size into the standard sized window, W(x,y). This is called a texture map.

Scanning Window Pattern Detectors (Reminder)



A scanning window is a brute force method to test if a pattern can be found in an image. Assume "gray-scale" image, *P(i,j)*, in which each pixel is an 8 bit luminance value.

An image window, or "imagette" is a rectangular region of the image. A window can be defined by two points: the top-left and bottom-right corners. This may be represented by a vector (t, l, b, r). Note that the origin is the upper left corner. Columns are numbered x=1 to W and rows are numbered y=1 to H

For any pixel *(i,j),* we can define a window, *W(x,y)*, of size C columns by R rows using the pixels from *P(i, j)* to *P(i+C-1, j+R-1)*.

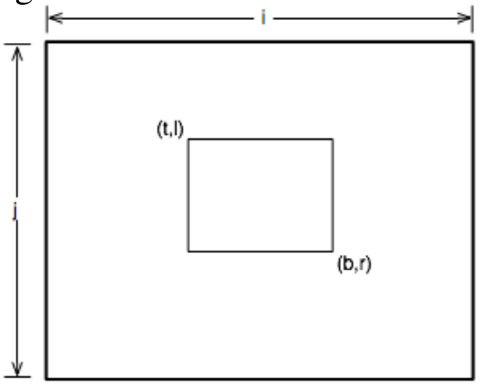
$W(x,y) = p(i+x-1, j+y-1)$ for x from 1 to C and y from 1 to R.

2. Difference of Adjacent Boxes

Box Features

A box feature is the sum of pixels within a rectangle.

Assume a rectangle from top (t) and left (l) to bottom (b) and right (r), with the constraints: top < bottom and right > left.

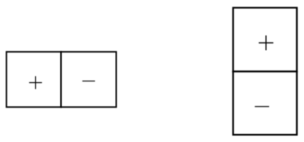


$$b(t,l,b,r) = \sum_{x=l}^{r}\sum_{y=t}^{b} W(x,y)$$

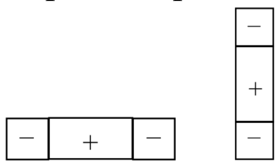
Difference of boxes can be used as features for classification.

Viola-Jones uses three kinds of Difference of Box features:

1) Two-rectangle features, computed as differences of two adjacent rectangular boxes of the same size. The boxes are the same size and shape and are adjacently aligned so that they share one side.

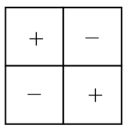


2) Three rectangle features, computed as the sum of two outside rectangles subtracted from an internal center rectangle. The sizes of the boxes are selected so that they have the same number of negative and positive pixels. (The coefficients sum to 0).



These can be both vertical and horizontal.

3) Four rectangle features, computed as the difference of diagonal pairs of rectangles.



For a window of 24 x 24 pixels, this gives more than 100,000 possible features! (136, 336 features according to Wikipedia). Computation is very fast because of the use of integral images.

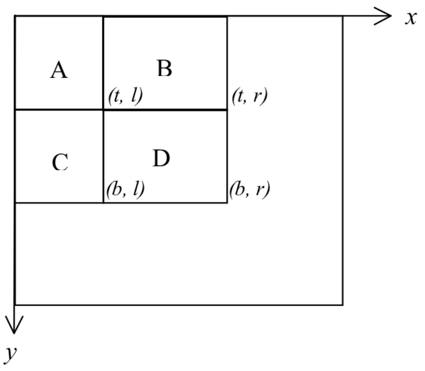
Integral Images

An integral image is an image where each pixel contains the sum from the upper left corner. We can build an integral image from our image window *W(x, y)* with:

$$ii(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{y} W(i,j)$$

Each sample in the integral image represents the sum of pixels from the upper left corner.

An integral image provides a structure for very fast computation of box features. Note that a sum of pixels in a rectangle can be computed from an integral image using only 4 operations (additions/subtractions).



Consider four adjacent rectangular regions A, B, C, D.

Note that *ii(t, l) = A*. *ii (t,r) = A+B* *ii(b,l)=A+C* *ii(b,r)=A+B+C+D*

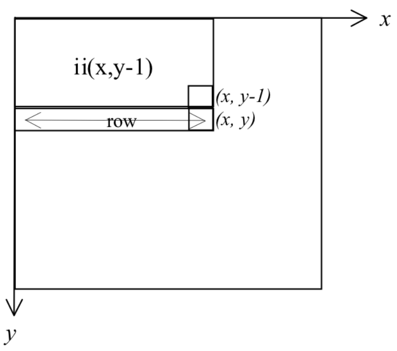
The sum of pixels for box D is D = *A+B+C+D – (A+B) – (A+C) + A*

box(t, l, b, r) = D
box(t, l, b, r) = A+B+C+D – (A+B) – (A+C) + A
box(t, l, b, r) = ii(b, r) – ii(t, r) – ii(b, l) + ii(t, l)

Fast Integral Image algorithm.

Integral images have been used for decades to compute local energy for normalization of images. For an R x C window of an image, W(i,j), the integral image, ii(x,y), is computed with a recursive algorithm that uses an intermediate buffer, "row", for a running sum of pixels within the current row.

```
ii(1,1) = W(1, 1)
For x = 2 to C
        ii(x,1) = ii(x-1,1) + W( x, 1)
For y = 2 to R
{       row=0   // reset the row buffer //
        For x = 1 to W
        {       row = row +  W(x,y)
                ii(x,y) = ii(x,y-1) + row
        }
}
```



Note that many authors use a less efficient algorithm that requires a running sums of all the columns.

Difference of Adjacent Boxes Features

A box feature is the sum of pixels in a rectangle. With integral images, a box feature costs 3 ops. (an add, subtract, or multiply is 1 op)

$$B(t, l, b, r) = ii(b,r) - ii(t,r) - ii(b,l) + ii(t,l)$$

Two rectangle features

A first order Difference of Boxes (DoB) feature is a difference of two boxes

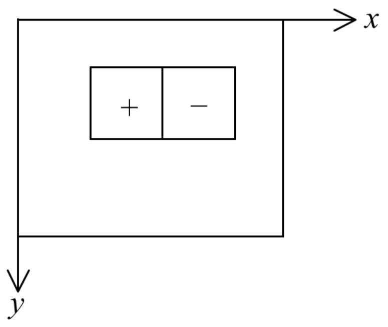
$$DoB(t_1,l_1,b_1,r_1,t_2,l_2,b_2,r_2) = box(t_1,l_1,b_1,r_1) - box(t_2,l_2,b_2,r_2)$$

An arbitrary 1st order difference of boxes costs 7 ops.

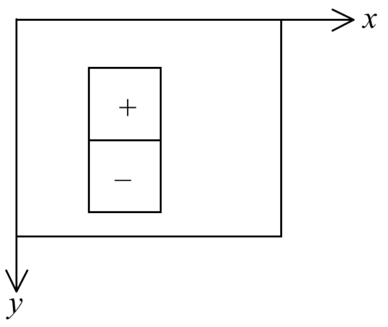
$$DoB(t_1,l_1,b_1,r_1,t_2,l_2,b_2,r_2) = box(t_1,l_1,b_1,r_1) - box(t_2,l_2,b_2,r_2)$$

$$= ii(b_1,r_1) - ii(t_1,r_1) - ii(b_1,l_1) + ii(t_1,l_1) - [\ ii(b_2,r_2) - ii(t_2,r_2) - ii(b_2,l_2) + ii(t_2,l_2)\ ]$$

Difference of Adjacent Boxes uses boxes of the same size that share a side.
There are two possible cases: Difference of Horizontally Adjacent Boxes (DoHAB) and Difference of Vertically Adjacent Boxes DoVAB



DoHAB()



DoVAB()

If the boxes share a vertical boundary, then $t_1 = t_2 = t$, $b_1 = b_2 = b$ and $r_1 = l_2$, and the boxes are horizontally adjacent.

$$DoHAB(t_1,l_1,b_1,r_1,b_2,r_2) = box(t_1, l_1, b_1, r_1) - box(t_2, l_2, b_2, r_2)$$
$$= box(t, l_1, b, r_1) - box(t, l_2, b, r_2)$$

but since $r_1 = l_2$:

$$= box(t, l_1, b, r_1) - box(t, r_1, b, r_2)$$

$= ii(b,r_1)–ii(t,r_1)–ii(b,l_1)+ii(t,l_1) – [ii(b,r_2)–ii(t,r_2)–ii(b, r_1)+ii(t, r_1)]$
$= ii(b, r_1) – ii(t, r_1) – ii(b, l_1) + ii(t, l_1) – ii(b, r_2) + ii(t, r_2) + ii(b, r_1) – ii(t, r_1)$
$=2\cdot ii(b, r_1) – 2\cdot ii(t, r_1) – ii(b,l_1) + ii(t,l_1) – ii(b,r_2) + ii(t,r_2)$

If the boxes share a horizontal boundary then $b_1=t_2$ and the boxes are vertically adjacent and $l_1=l_2$ and $r_1=r_2$

$DoVAB(t_1,l_1,b_1,r_1,b_2,r_2) = box(t_1, l_1, b_1, r_1)–box(t_2, l_2, b_2, r_2)$
$= box(t, l_1, b, r_1)–box(t, l_2, b, r_2)$

since $r_1 = l_2$:

$= box(t, l_1, b, r_1)–box(t, r_1, b, r_2)$

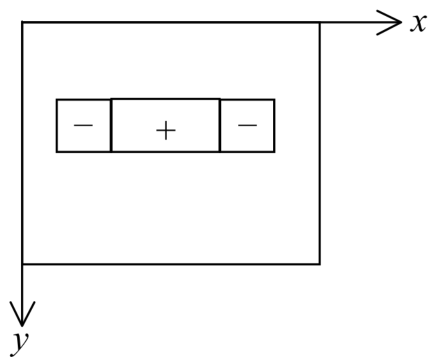
$=ii(b,r_1)–ii(t,r_1)–ii(b,l_1)+ii(t,l_1) – [ii(b, r_2) – ii(t, r_2) – ii(b, r_1) + ii(t, r_1)]$
$=ii(b, r_1)–ii(t, r_1)–ii(b, l_1)+ii(t, l_1) – ii(b, r_2) + ii(t, r_2) + ii(b, r_1) –ii(t, r_1)$
$=2\cdot ii(b, r_1) – 2\cdot ii(t, r_1) – ii(b, l_1) + ii(t, l_1) – ii(b, r_2) + ii(t, r_2)$

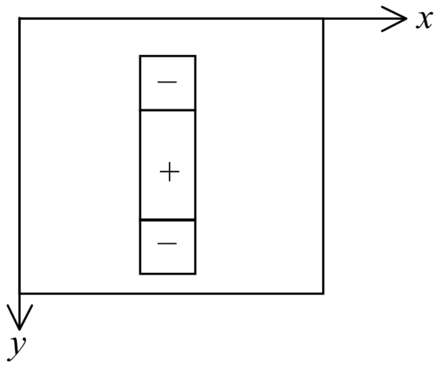
$DoVAB(t, l_1,b, r_1,b, r_2) =2\cdot ii(b, r_1)–2\cdot ii(t, r_1)–ii(b, l_1) + ii(t, l_1) – ii(b, r_2) + ii(t, r_2)$

The fact that both rectangles are the same size, guarantees that the feature is zero for a constant region. The difference of adjacent boxes costs 7 ops. (2 mults, 3 subtracts, 2 adds)

Three rectangle features

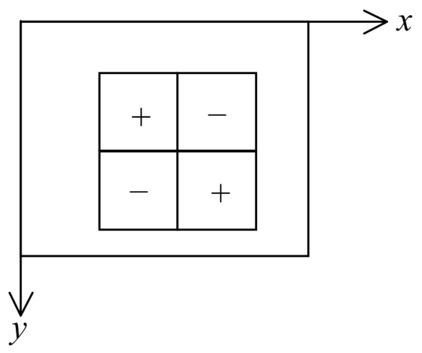
Three rectangle features are computed as the sum of two outside rectangles subtracted from an internal center rectangle. The size of the inner rectangle is twice the size of the outer rectangles. This guarantees that the sum is zero when covering a uniform region.





Three rectangle features cost 11 ops.

Four rectangle features



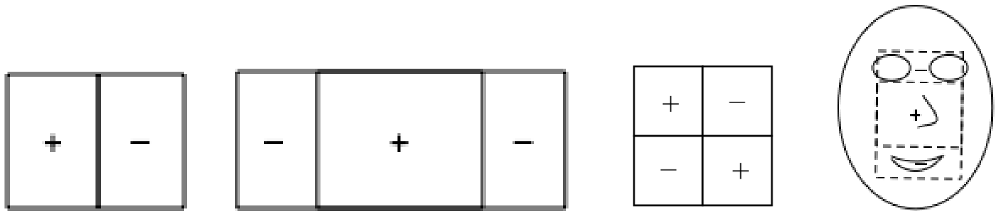
Four rectangle features, computed as the difference of diagonal pairs of rectangles. Difference of adjacent boxes are similar to Haar wavelets.

Note that a difference of boxes can be seen as a computing an inner product of the window with a filter *$H_n(x,y)$* (also called a mask, or a receptive field).

$$X_n = \sum_{x=1}^{C} \sum_{y=1}^{R} W(x,y) H_n(x,y)$$

3. Linear Classifiers for Face Detection

Let us assume a set of N difference of adjacent boxes, $H_n(x,y)$ composed of two, three and four box features.



For a 24x24 window, there will be over 100,000 (N > 100,000)
Each can be used to define a feature, $\{X_n\}$ for an imagette W, computed from

$$X_n = \langle W, H_n \rangle = \sum_{x=1}^{C} \sum_{y=1}^{R} W(x,y) H_n(x,y)$$

These can be seen as defining more than 100,000 dimensional space for classifying imagettes of size 24x24 pixels.

Some features respond to the appearance of a face. These can be used to determine if the imagette contains a face or not.

Each image feature specifies a weak classifier for the window: $h_n(W)$

$$h_n(W) = \begin{cases} 1 & \text{if } p_n(X_n + b_n) > 0 \\ 0 & \textit{otherwise} \end{cases}$$

where p_n is a “polarity” of +1 or -1 and b_n is a bias. $h_n(W)$ represents a hypothesis.

Each weak classifier, $h_n(W)$ can be seen as a hyper-plane that partitions the hyper-dimensional feature space of imagettes of size 24x24. The problem is to choose the best $h_n(W)$ so that most non-face windows are on one side of the hyper-plane and most face windows are on the other.

To do this we will use a "training" set of M windows, $\{W_m\}$.
Each training window is labeled with an "indicator variable" y_m.
For imagettes that contain faces, $y_m = 1$. Imagettes that do not contain faces, $y_m = 0$.

Training a committee of classifiers

Assume a set of *M* face windows $\{W_m\}$ that have been labeled by a set of labels $\{y_m\}$ such that y=+1 if face and y=0 if not face.

Then for any imagette, W_m, each feature "votes" for a face (1 or P for positive) or not a face (0 or N for negative).

$$h_n(W) = \begin{cases} 1 & \text{if } p_n(X_n + b_n) > 0 \\ 0 & otherwise \end{cases}$$

For a training set of M windows, $\{W_m\}$, the detection rate (or positive rate) for a weak classifier $h_n(W)$ is the percentage of positive detections.

$$P_n = \frac{1}{M}\sum_{m=1}^{M} h_n(W_m)$$

Positive detections can be true positive and false positives.

Whether a detection is true (T) of false (F) can be determined by the indicator variable. $y_m=1$ if W_m contains a face, and $y_m=0$ otherwise.

if | $h_n(W_m) - y_m$ | = 1 then FALSE else TRUE.

For the training set of *M* windows, $\{W_m\}$, the error rate for a weak classifier $h_n(W)$ is the percentage of false classifications.

$$E_n = \frac{1}{M}\sum_{m=1}^{M} \left|h_n(W_m) - y_m\right|$$

Note that the error rate is a number between 0 and 1.

The classifier $h_n(W)$ that minimizes the error rate is

$$h_n = \arg\text{-}\min_n \{\sum_{m=1}^{M} \left|h_n(W_m) - y_m\right|\}$$

4. AdaBoost

AdaBoost (adaptive Boosting) is a meta-algorithm for learning a 2-class detection functions. Adaboost builds a strong classifier from a large number of weak classifiers. The outputs of the weak classifiers are combined as a weighted sum of votes. The resulting strong committee can be made arbitrarily good by adding more weak classifiers.

Adaboost is particularly useful in problems with a large number of features or large numbers of possible weak classifiers.

The Boosted Classifier

The boosted classifier can be seen as a form of Committee that decides using weighted votes by a set of T weak classifiers $h_i(W)$.

A weighted committee has the form:

$$h(W) = \begin{cases} 1 & \text{if} \sum_{t=1}^{T} \alpha_t h_t(W) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $h_t(W)$ is a weak classifier and α_t is a learned weight for each weak classifier that depends on the error rate E_t where $\alpha_t = \log \frac{1}{\beta_t}$ and $\beta_t = \frac{E_t}{1 - E_t}$

A set of N weak classifiers, $h_n(W)$, maps a window W into N votes.

$$h_n(W) = \begin{cases} 1 & \text{if } p_n(X_n + b_n) > 0 \\ 0 & otherwise \end{cases}$$

These can be counted and compared to a threshold (N/2 for simple majority).

Committee decision: $h(W) = \begin{cases} 1 & \sum_{n=1}^{N} h_n(W) \geq \frac{N}{2} \\ 0 & otherwise \end{cases}$

However, with Adaboost, not all votes are equal! Adaboost learns a "weight", a_n, for each vote.

$$h(W_m) = \begin{cases} 1 & \sum_{n=1}^{N} \alpha_n h_n(W_m) \geq \frac{1}{2} \sum_{n=1}^{N} \alpha_n \\ 0 & otherwise \end{cases} \qquad \text{where: } h_n(W) = \sum_{x=1}^{W} \sum_{y=1}^{H} W(x,y) H_n(x,y)$$

Boosted learning is an iterative procedure to choose weak classifiers and weights from a training set of *M* training windows, $\{W_m\}$ with their indicator variables $\{y_m\}$. Let #P be the number of positive training samples, and #N be the number of negative training samples.
(# is the cardinality operator - it counts the number of times something happens).

The algorithm estimates a weight for each training sample, w_m.
The weights are initially set to

$$w_m = \begin{cases} \frac{1}{2\#P} & \text{if } \mathrm{y_m} = 1 \\ \frac{1}{2\#N} & \text{if } \mathrm{y_m} = 0 \end{cases}$$

to compensate for an unequal number of positives and negatives.

The algorithm then iterates over the number of weak classifiers.

The AdaBoost Algorithm:

Initialize the weights as: $w_m = \begin{cases} \frac{1}{2\#P} & \text{if } \mathrm{y_m} = 1 \\ \frac{1}{2\#N} & \text{if } \mathrm{y_m} = 0 \end{cases}$

Initialize the algorithm with a weak classifier $h_1(W)$

$$h_1 = \arg\text{-}\min_{h_n}\{\sum_{m=1}^{M} w_m |h_n(W_m) - y_m|\}$$

the weight, α_1, is determined from the error rate: $E = \frac{1}{M}\sum_{m=1}^{M} w_m |h_1(W_m) - y_m|$

$$\alpha_1 = \log\left(\frac{1-E_T}{E_T}\right) \qquad \text{where } \beta_T = \frac{E_T}{1-E_T}$$

This is classifier t=1. Set T=1. Remove h_1 from the set of available classifiers.

Loop until the error rate E_t is below a specified error rate :

Let T=T+1

1) Normalize the weights to sum to 1. This converts w_m to a probability

$$S = \sum_{m=1}^{M} w_m ; \qquad w_m \leftarrow \frac{w_m}{S}$$

2) For each Difference of Box feature, n, determining the polarity p_n and the threshold b_n that gives the best error rate with the current weights.

$$p_n, b_n = \arg\text{-}\max_{p,b} \left\{ \sum_{m=1}^{M} w_m \left| h_n(W_m) - y_m \right| \right\}$$

This gives a new weak classifier $h_n(W) = \begin{cases} 1 & \text{if } p_n(X_n + b_n) > 0 \\ 0 & otherwise \end{cases}$

with error rate $E_n = \frac{1}{M} \sum_{m=1}^{M} w_m \left| h_n(W_m) - y_m \right|$

In this step the weights will bias the vote to give more strength to training samples that are improperly classified by the committee.

3) Choose the new weak classifier with the lowest error rate using the current weights. This is the T^th^ weak classifier is h_t, a_t:

$$h_T = \arg\text{-}\max_{h_n} \sum_{m=1}^{M} w_m \left| h_n(W_m) - y_m \right|$$

with the coefficient $\alpha_t = \log \frac{1 - E_T}{E_T}$ where $E_T = \frac{1}{M} \sum_{m=1}^{M} w \left| h(W_m) - y_m \right|$

4) Use the error rate to update the weights to give more strength to windows that are in error. For each training sample, each weight w_m is multiplied by a factor β_m

$$w_m = w_m \beta_m$$

where $\beta_m = \begin{cases} \frac{E}{1-E} & \text{if } \sum_{t=1}^{T} \alpha_t h_t(W_m) - y_m < 0 \quad \text{FALSE} \\ 1 & otherwise \quad \text{TRUE} \end{cases}$

END Loop

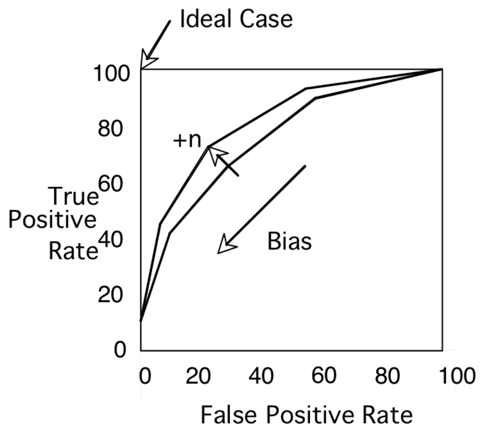
The final strong classifier is

$$h(W_m) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(W_m) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

The new error rate for the committee is $E_T = \frac{1}{M} \sum_{m=1}^{M} w_m \left| h(W_m) - y_m \right|$

ROC Curve for a weighted committee

The ROC plots the True Positive Rate (TPR) against False Positive Rate (FPR) for a classifier as a function of the global bias B.



The Boosting theorem states that adding each new weak 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.

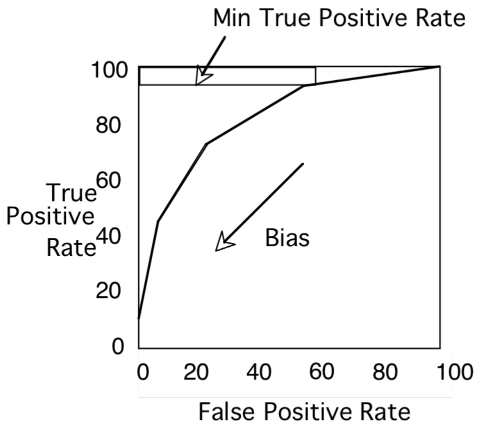
However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end up with a very very large number of classifiers.

The halting criteria for boosted learning is set in terms of the FPR and TPR. When the ROC curve goes above for point (FPR, TPR) for some Bias B, the algorithm halts.

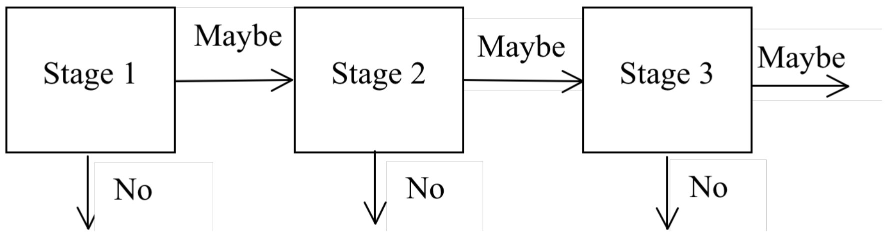
Learning a multi-stage cascade of classifiers

We can optimize the computation time by separating the committee into a multi-stage cascade of committees.

Each stage is composed of a committee that is designed with avoids rejecting possible true positives (high TPR: True Positive Rate) at the cost of accepting many False Positives (high False Positive Rate).



We construct each stage using only training data that passed the previous stage. Later stages are more expensive but are used less often.

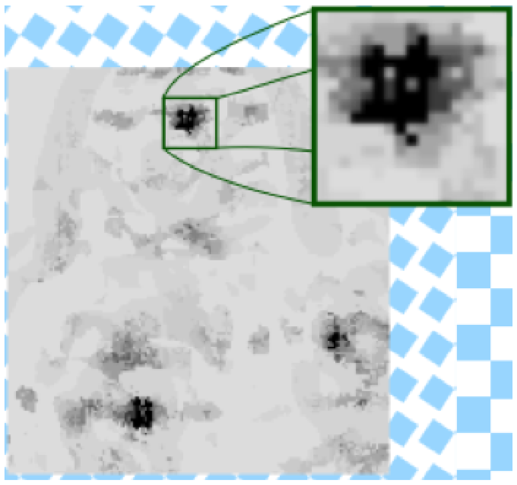


For each stage we set a minimum acceptable target for True Positives using the training data and accept the false positive rate that results.

Note that this can result in over-fitting the training data. It is important that the training data represent as large a variety of data as possible.

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

This is called a "cascade classifier"
Note that applying this to every position gives an "image" of cascade depths.



Faces can be detected as the center of gravity of "deep" detections.
Faces can be tracked using the Bayesian tracking described in the previous session.

This algorithm is part of the OpenCV. It is widely used in digital cameras and cell phones for face detection and tracking.