Face Detection

Vinay P. Namboodiri

- Slide credit to Svetlana Lazebnik and Gabor Melli, Kristen Grauman and Derek Hoiem

Image Categorization

Previous Classes

Training

Training Images

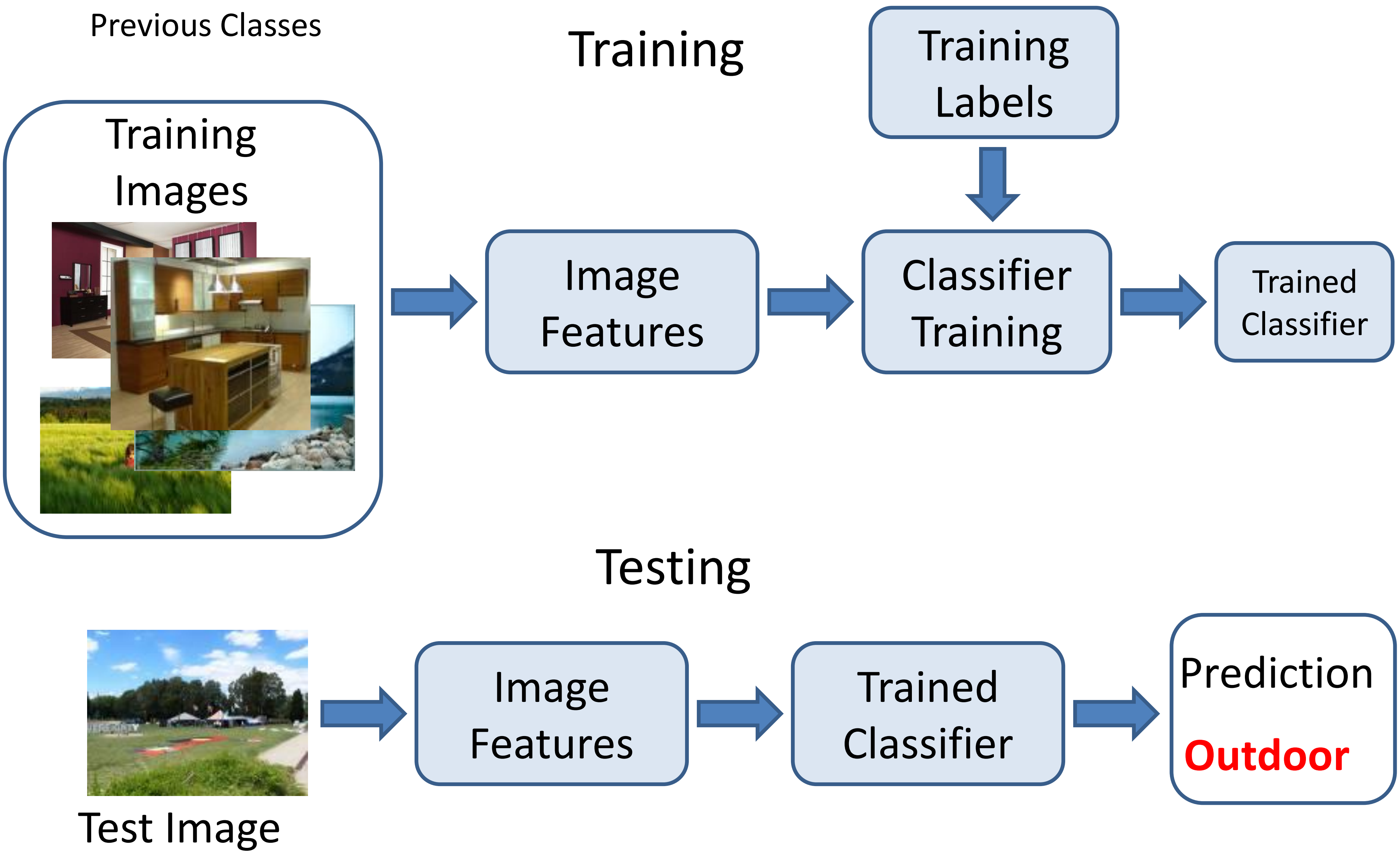
Image Features

Training Labels

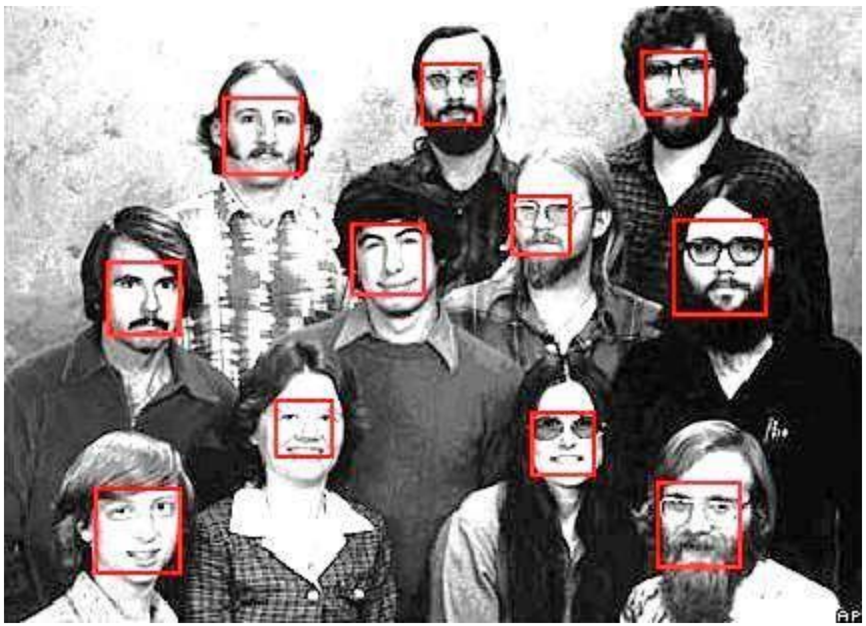
Classifier Training

Trained Classifier

Image Categorization



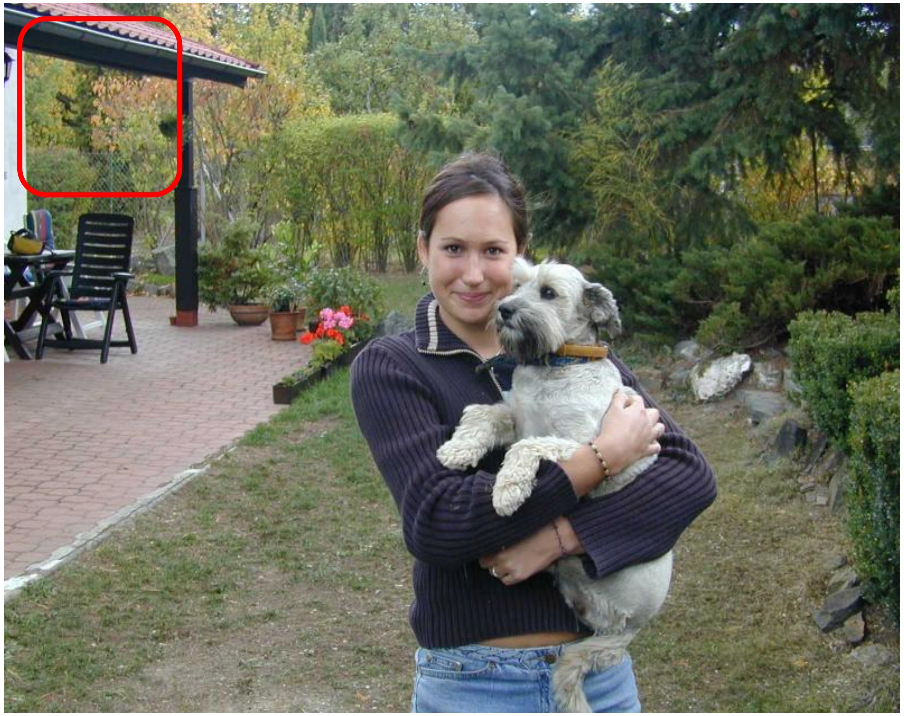
Face detection



Many slides adapted from P. Viola

Face detection

- Basic idea: slide a window across image and evaluate a face model at every location



Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image image, our false positive rate has to be less than 10^{-6}

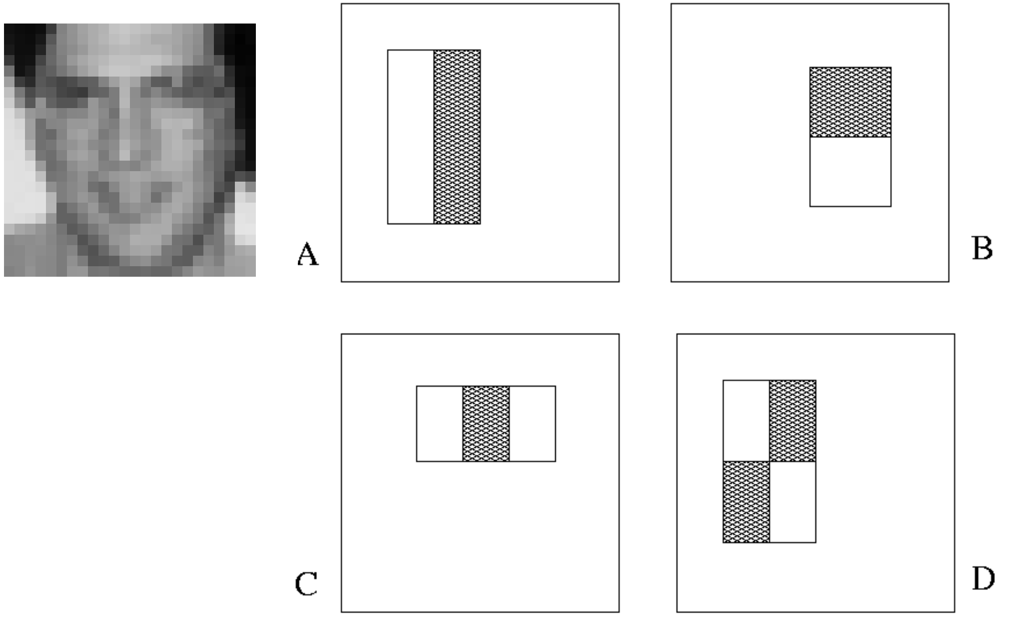
The Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Image Features

"Rectangle filters"

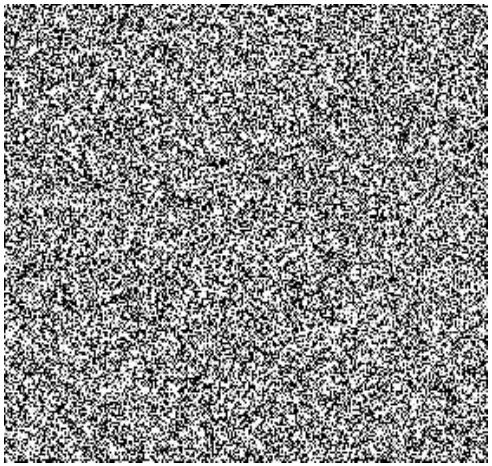


Value =

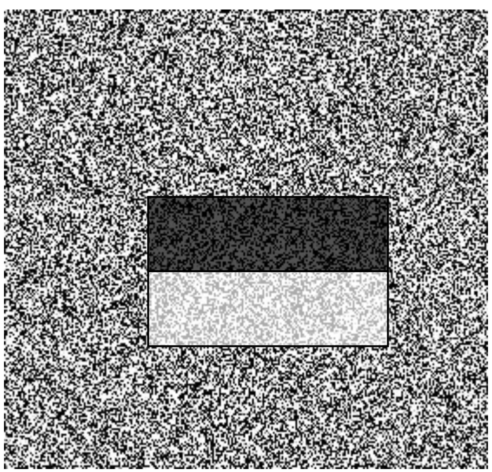
∑ (pixels in white area) –
∑ (pixels in black area)

Example

Source

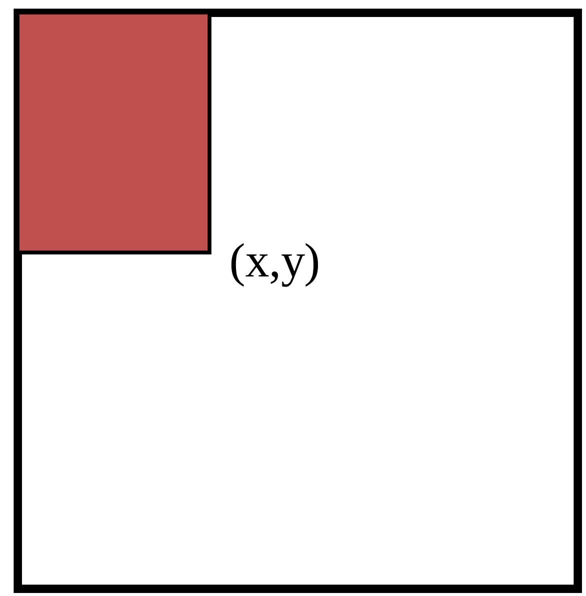


Result

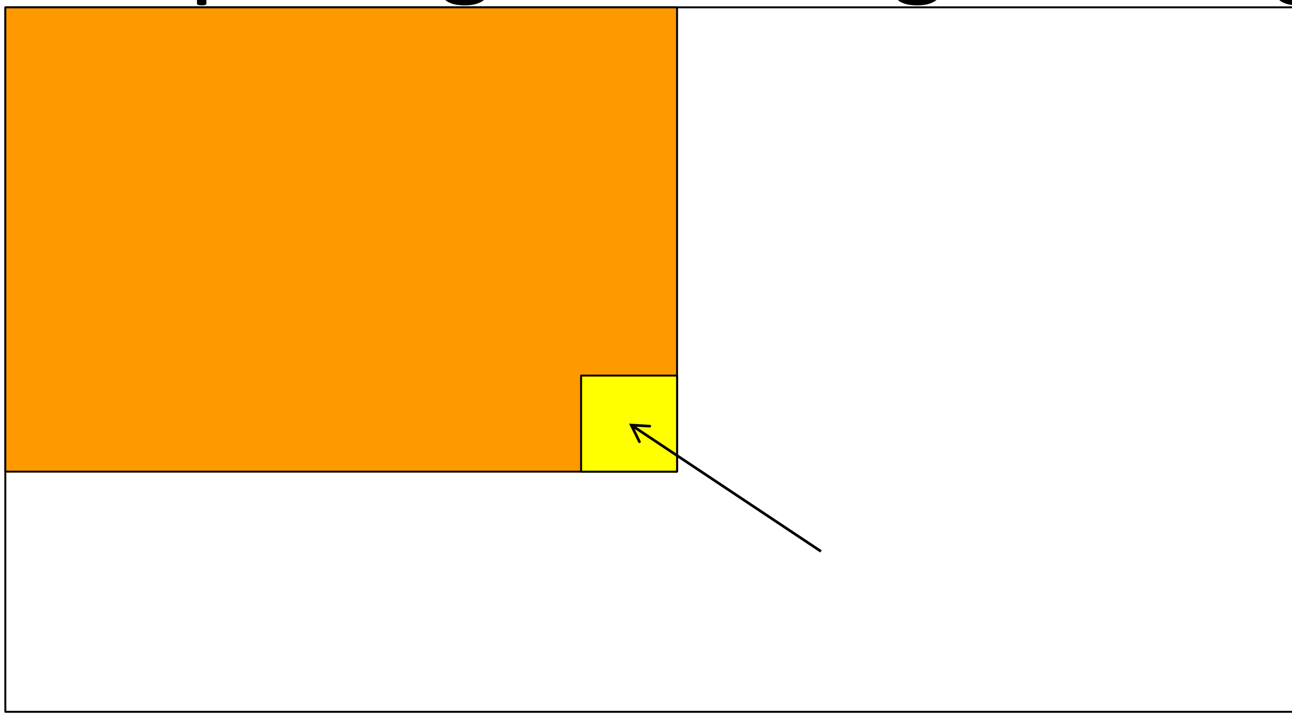


Fast computation with integral images

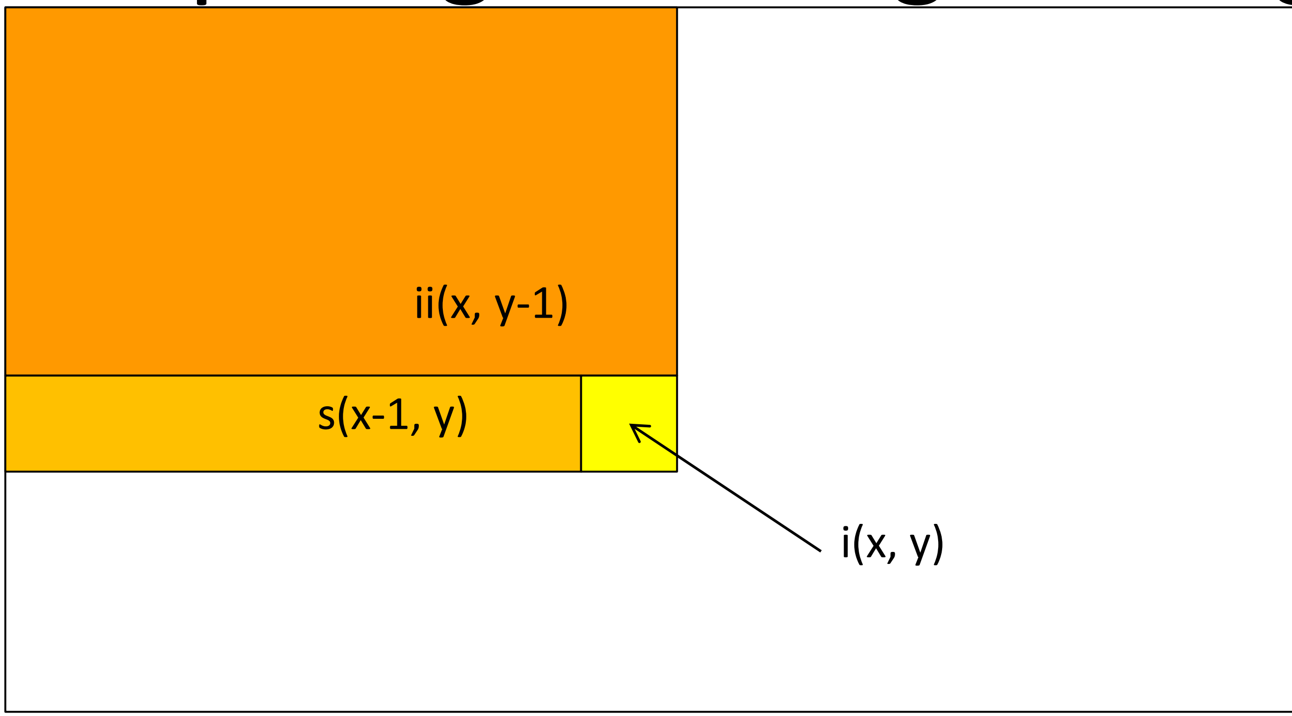
- The *integral image* computes a value at each pixel (*x*,*y*) that is the sum of the pixel values above and to the left of (*x*,*y*), inclusive
- This can quickly be computed in one pass through the image



Computing the integral image



Computing the integral image



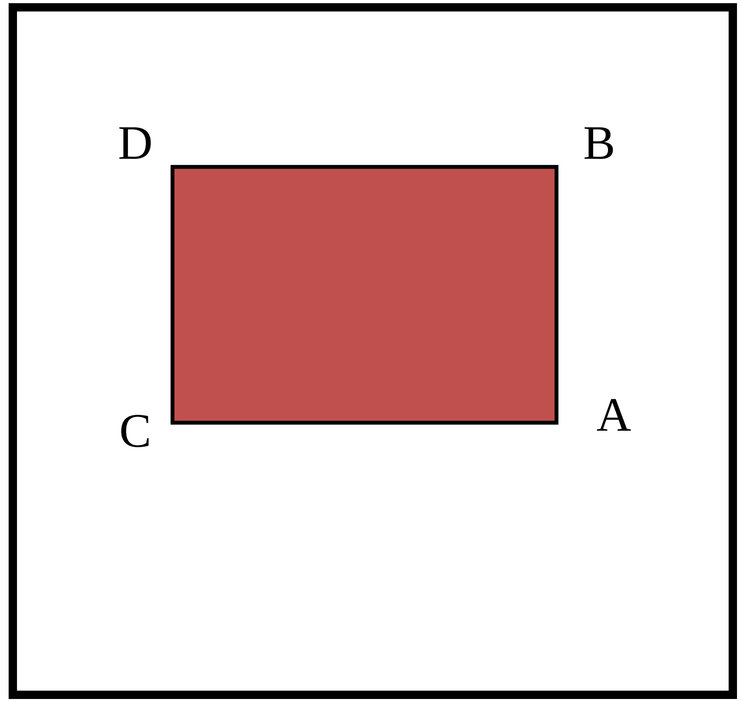
- Cumulative row sum: s(x, y) = s(x–1, y) + i(x, y)
- Integral image: ii(x, y) = ii(x, y–1) + s(x, y)

MATLAB: ii = cumsum(cumsum(double(i)), 2);

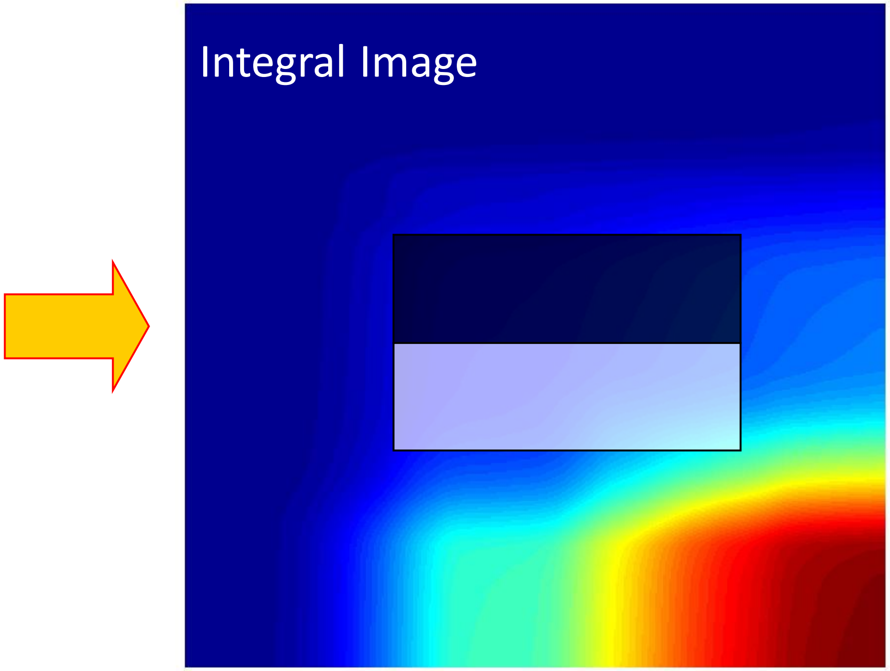
Computing sum within a rectangle

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

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!

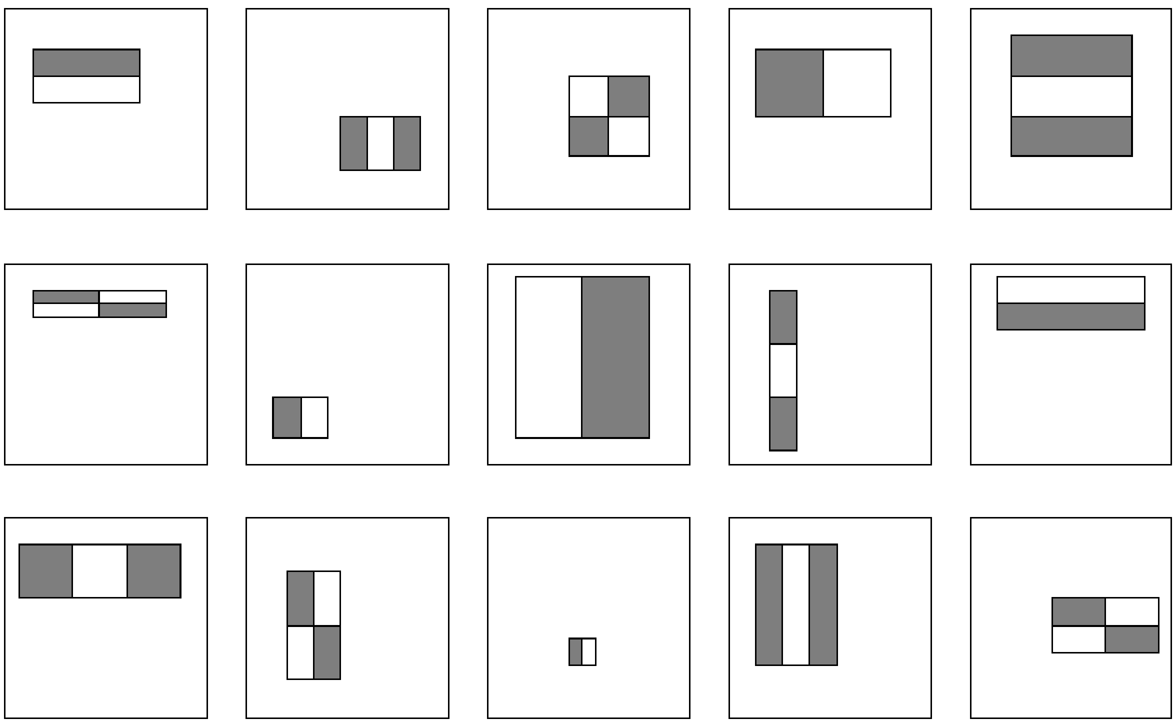


Example



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
  - A weak learner need only do better than chance
- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - "Hardness" is captured by weights attached to training examples

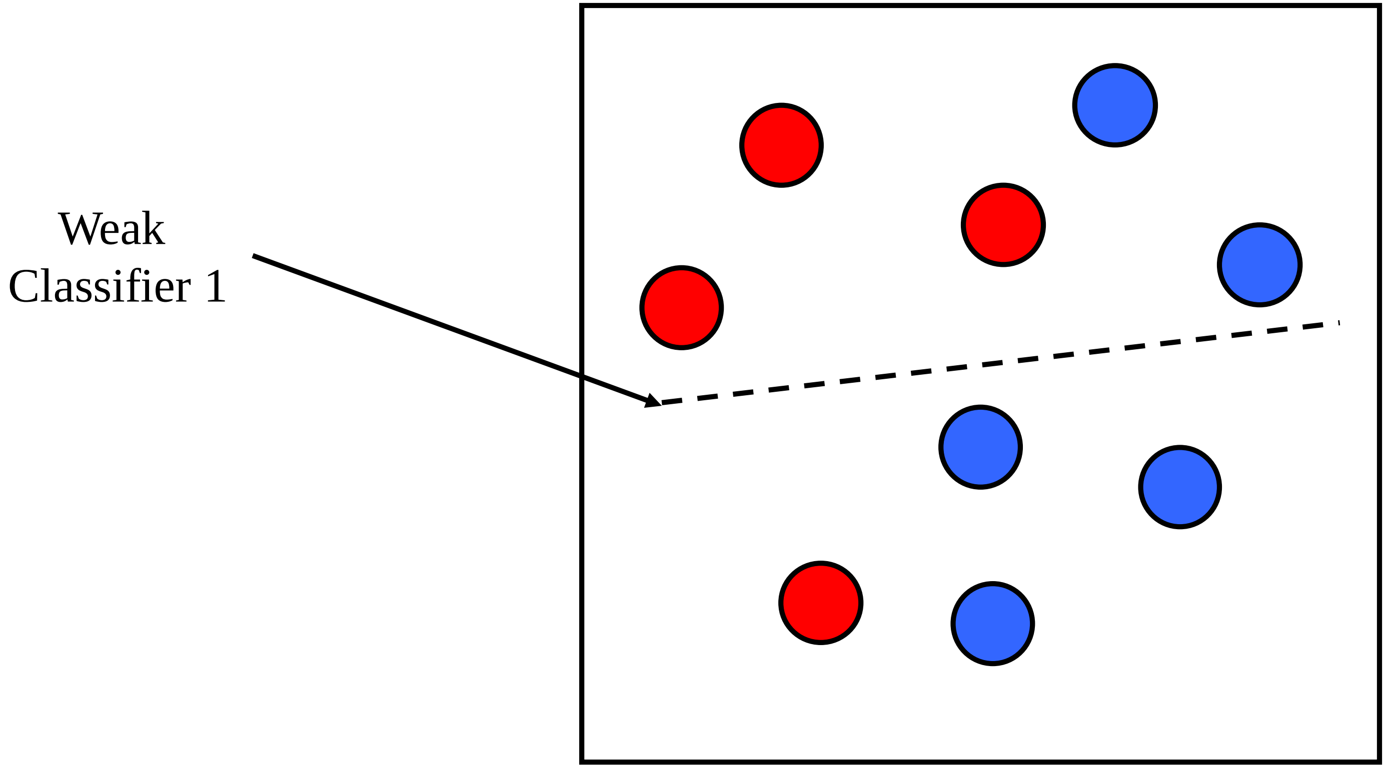
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Training procedure

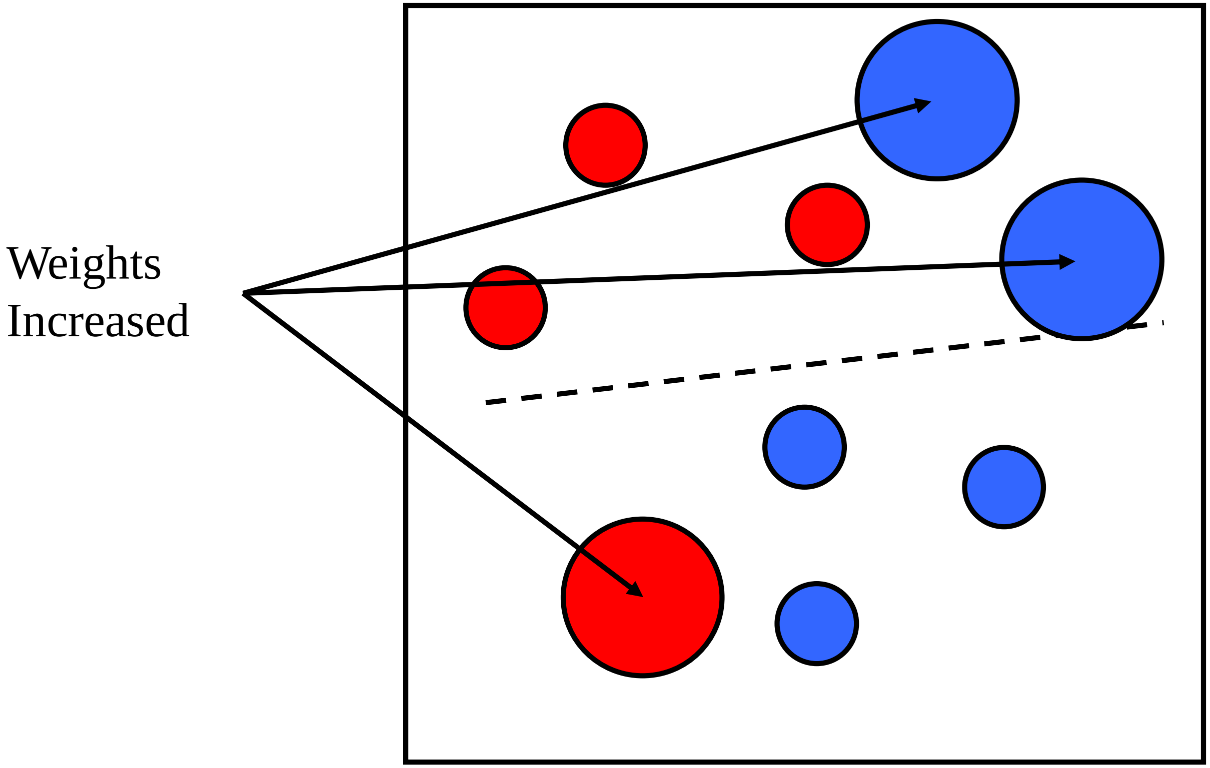
- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Y. Freund and R. Schapire, [A short introduction to boosting](), *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

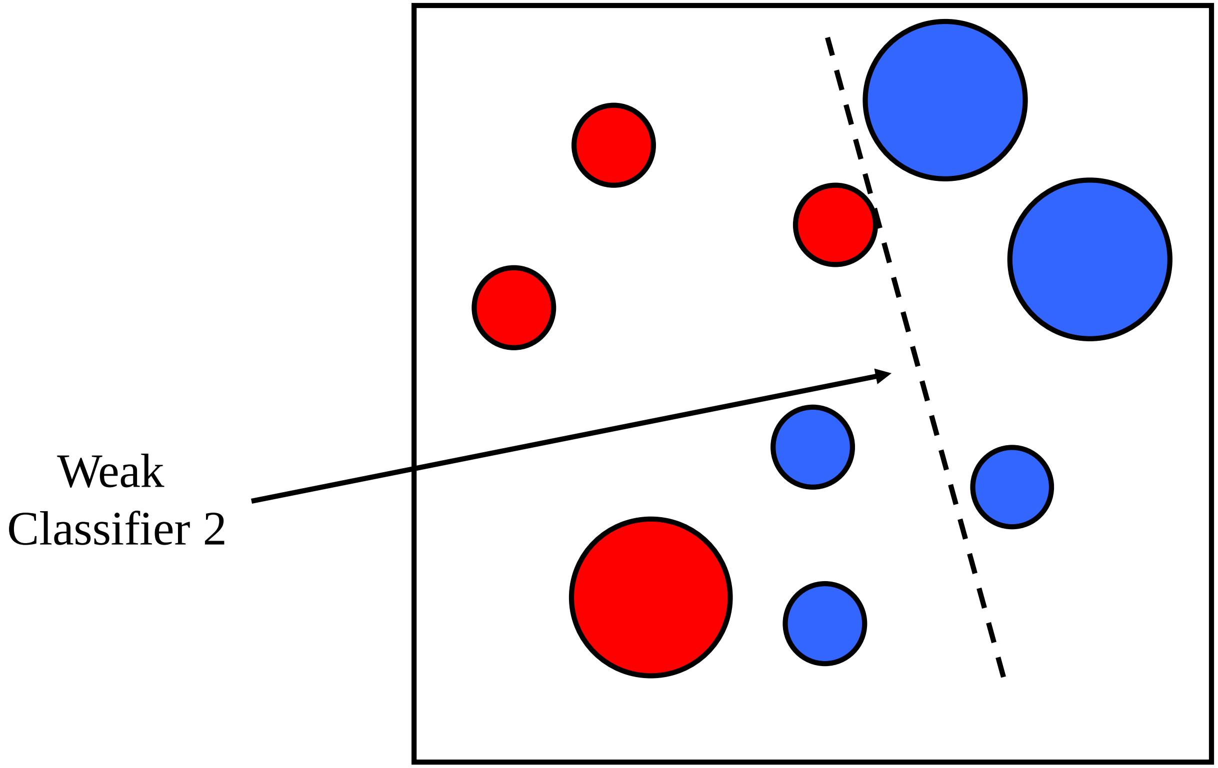
Boosting illustration



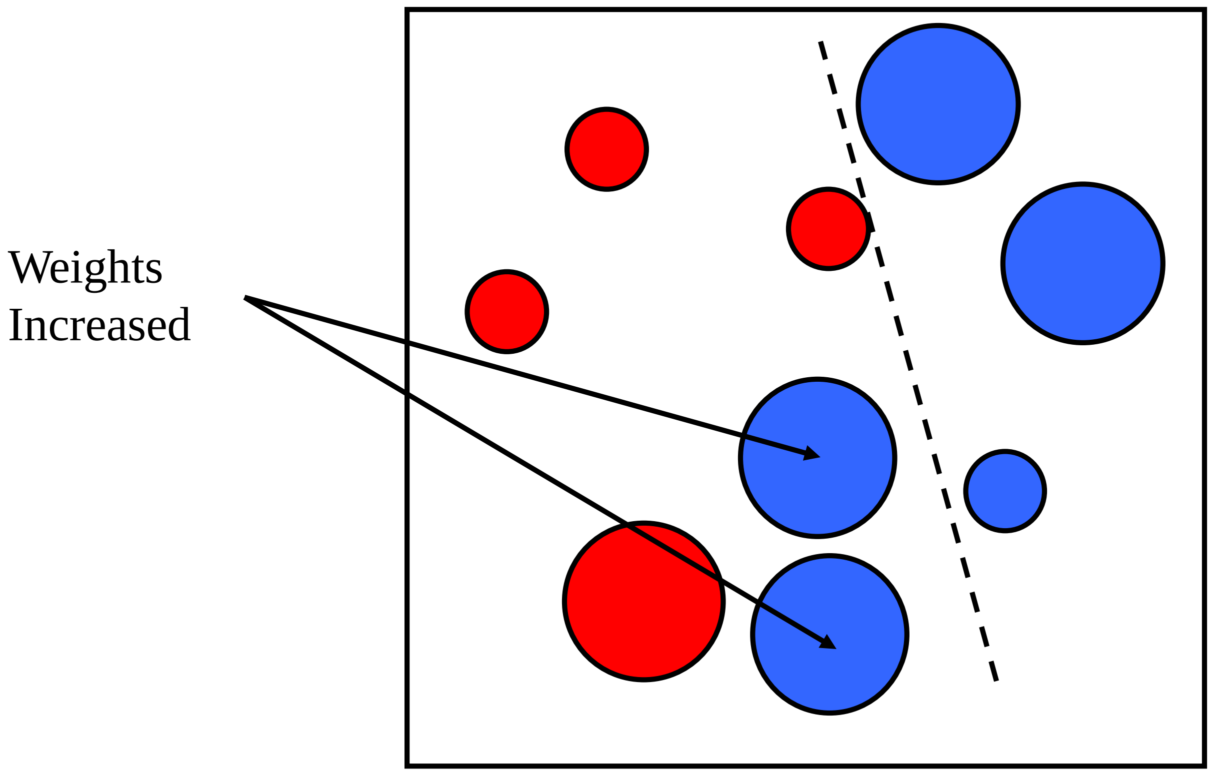
Boosting illustration



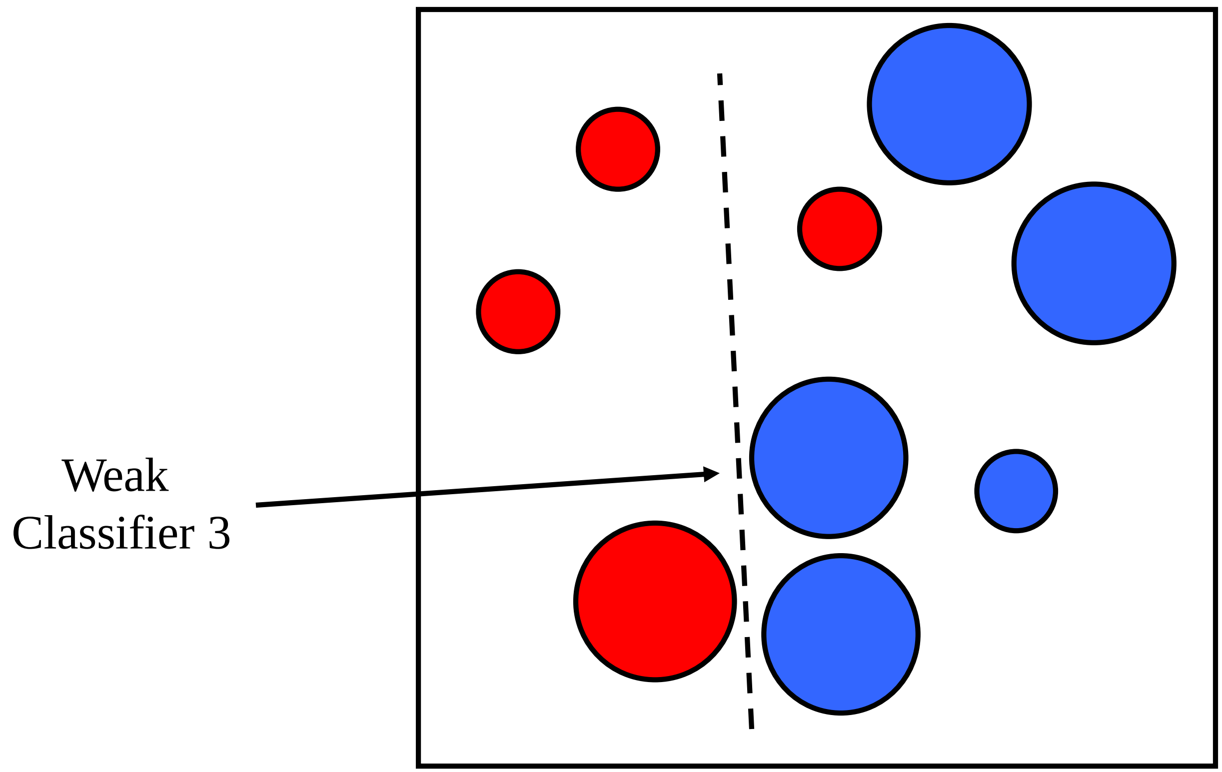
Boosting illustration



Boosting illustration

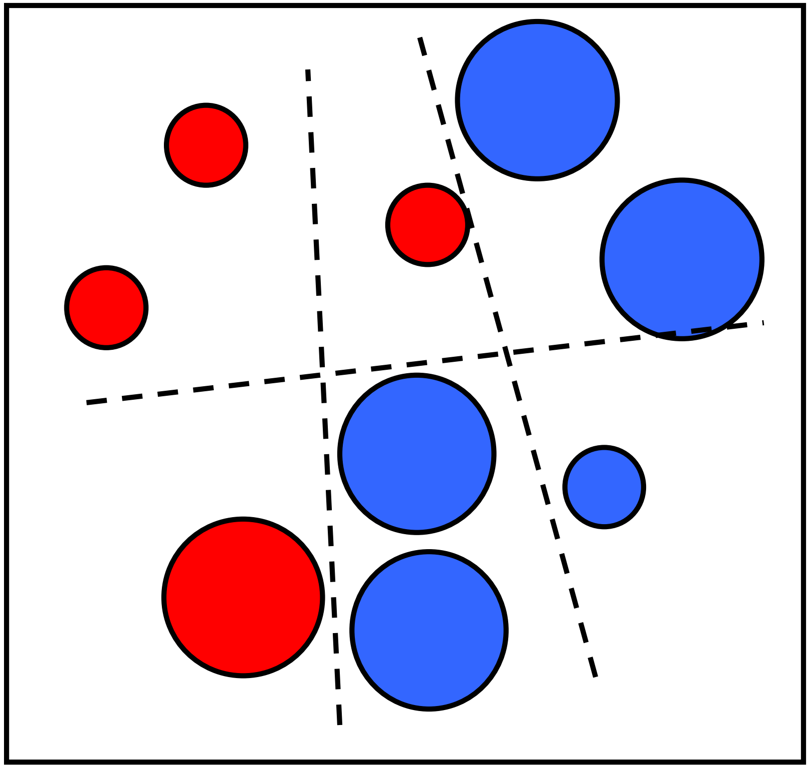


Boosting illustration



Boosting illustration

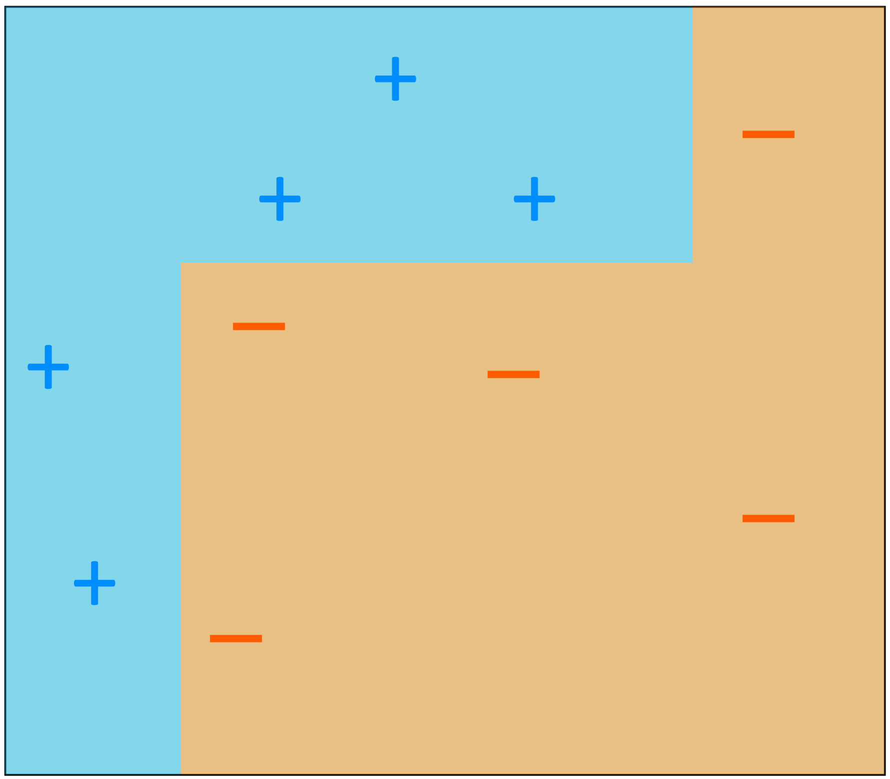
Final classifier is
a combination of weak
classifiers



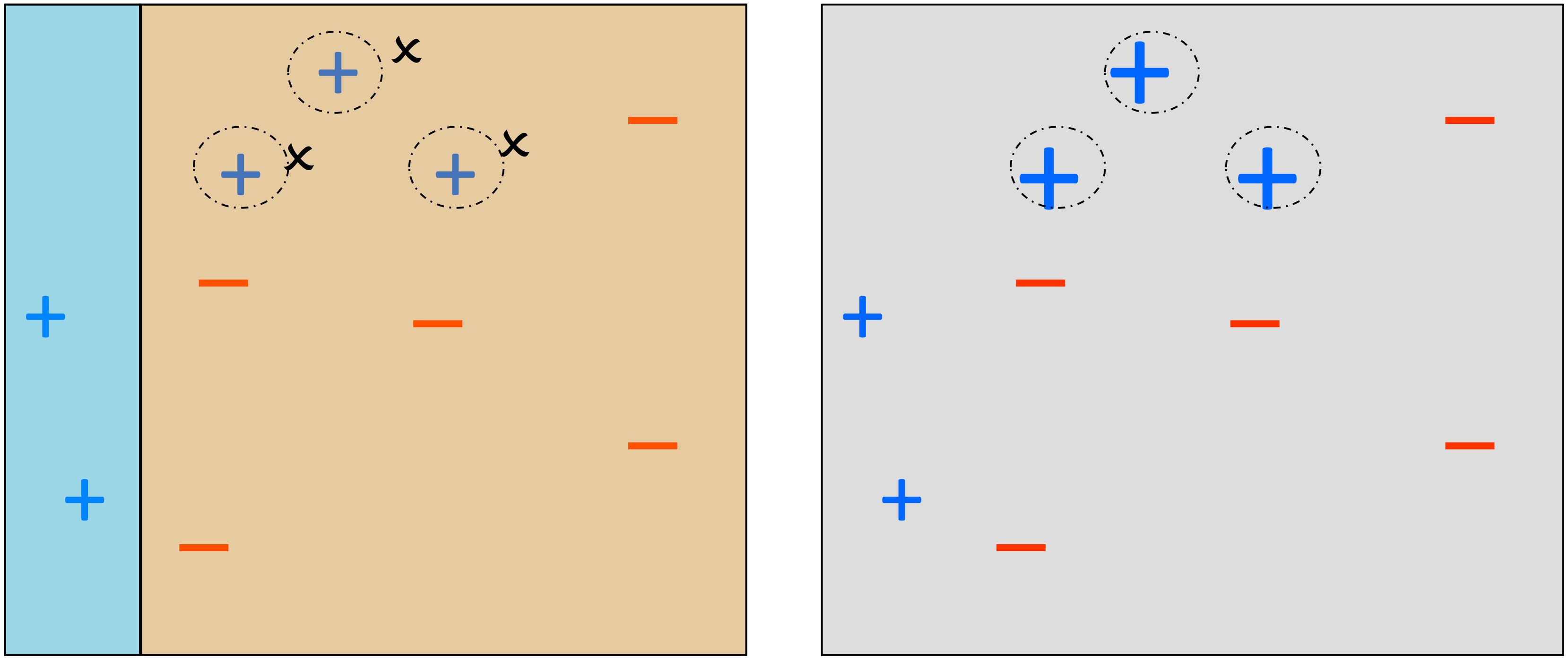
What is Boosting?

- A method for improving classifier accuracy
- Basic idea:
  - Perform iterative search to locate the regions/ examples that are more difficult to predict.
  - Through each iteration reward accurate predictions on those regions.
  - Combines the rules from each iteration.
- Only requires that the underlying learning algorithm be better than guessing.

Example of a Good Classifier



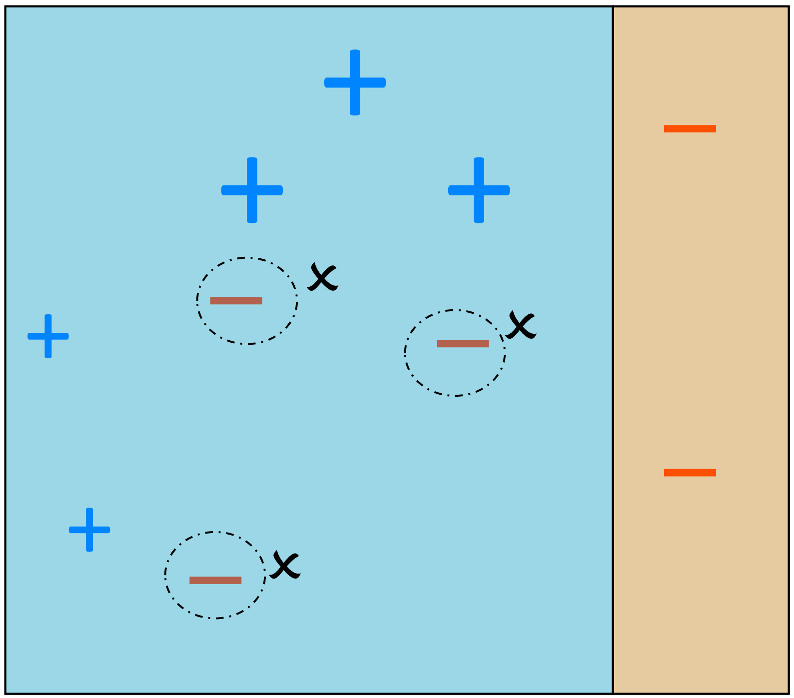
Round 1 of 3

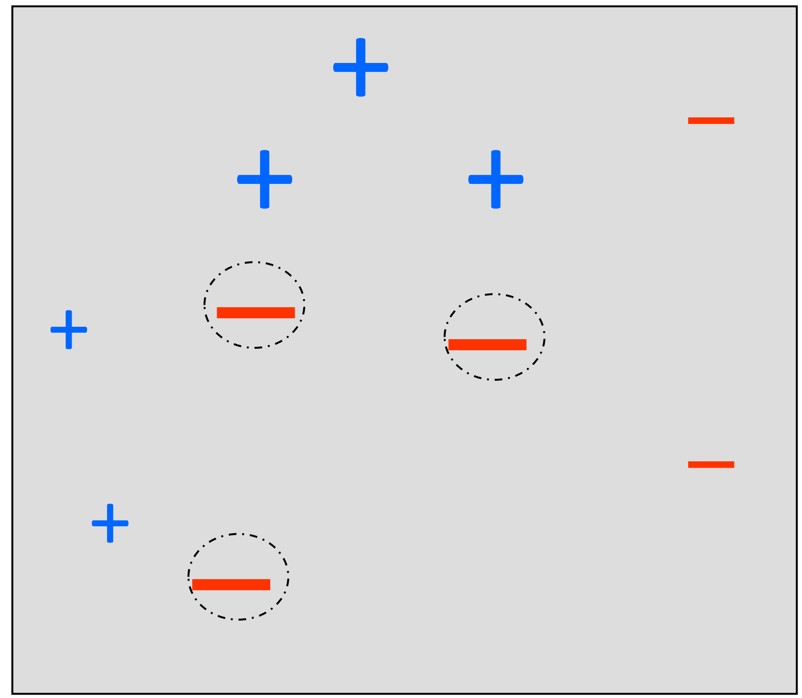


h_1 $\varepsilon_1 = 0.300$ D_2

$\alpha_1 = 0.424$

Round 2 of 3



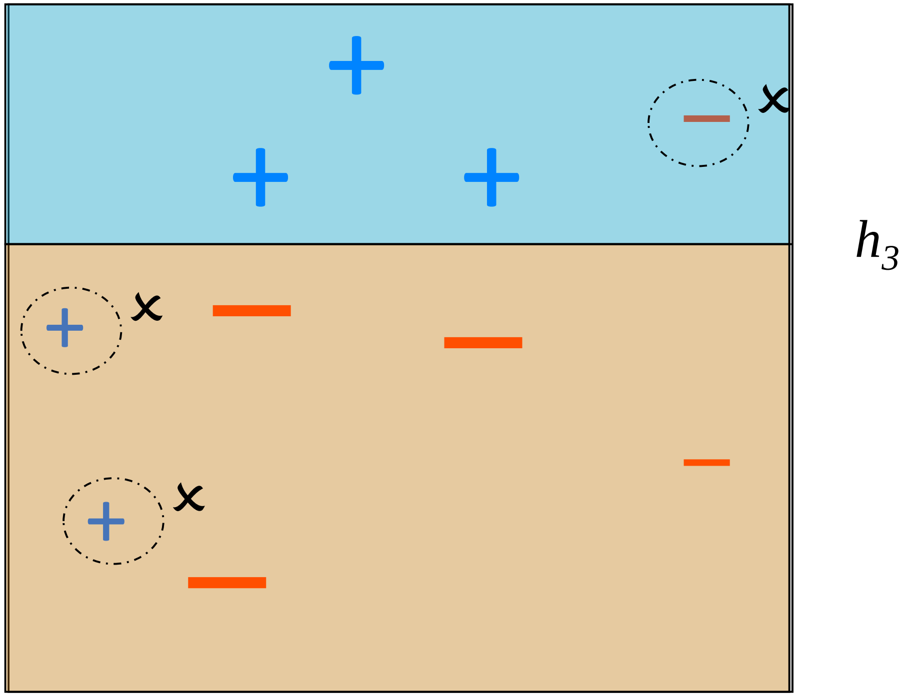


$\varepsilon_2 = 0.196$ h_2

$\alpha_2 = 0.704$

D_2

Round 3 of 3

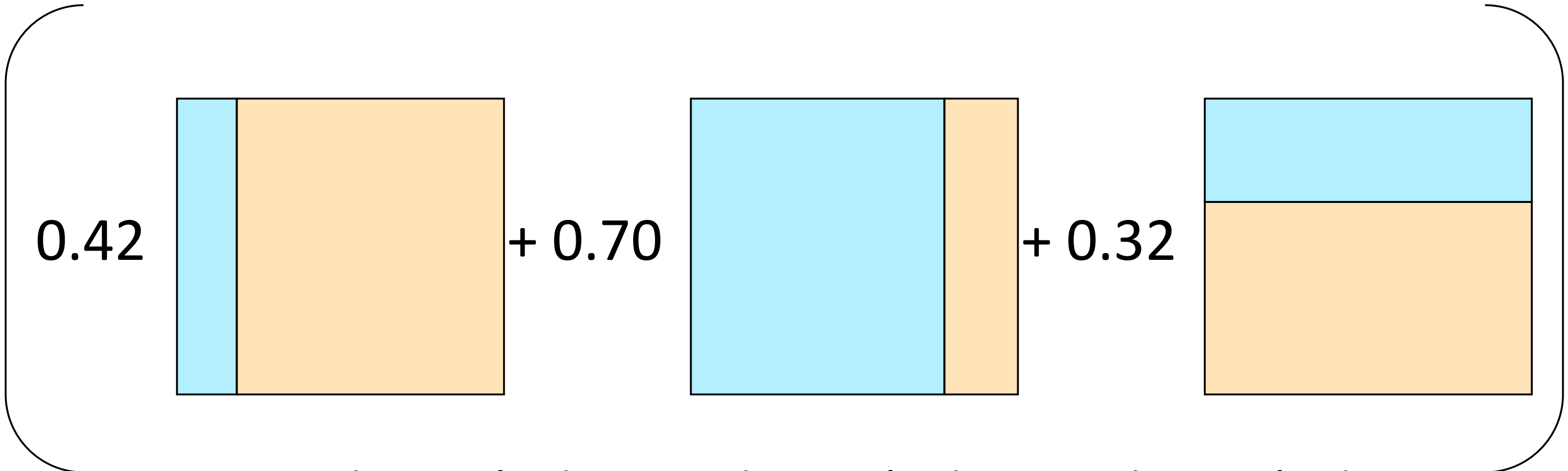


STOP

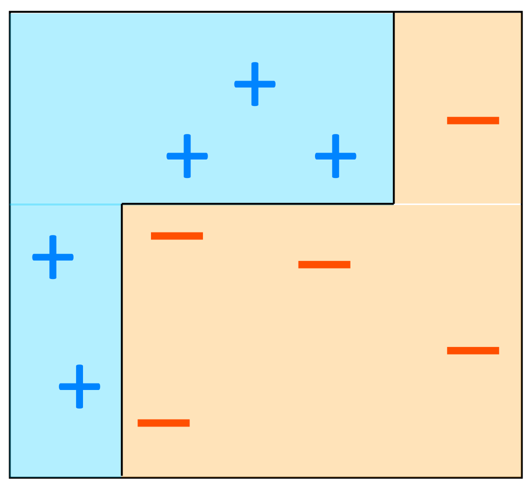
$\varepsilon_3 = 0.344$

$\alpha_2 = 0.323$

Final Hypothesis



$$H_{final} = \text{sign}[\ 0.42(h1?\ 1|-1) + 0.70(h2?\ 1|-1) + 0.32(h3?\ 1|-1)\ ]$$



History of Boosting

- "Kearns & Valiant (1989) proved that learners performing only slightly better than random, can be combined to form an arbitrarily good ensemble hypothesis."
- Schapire (1990) provided the first polynomial time Boosting algorithm.
- Freund (1995) “Boosting a weak learning algorithm by majority”
- Freund & Schapire (1995) AdaBoost. Solved many practical problems of boosting algorithms. “Ada” stands for adaptive.

AdaBoost

Given: m examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For t = 1 to T

1. Train learner $\boldsymbol{h_t}$ with min error $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$

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

2. Compute the hypothesis weight $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

The weight **<u>Ada</u>**pts. The bigger ε_t becomes the smaller α_t becomes.

3. For each example i = 1 to m

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

Boost example if incorrectly predicted.

Z_t is a normalization factor.

Output

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

Linear combination of models.

Boosting vs. SVM

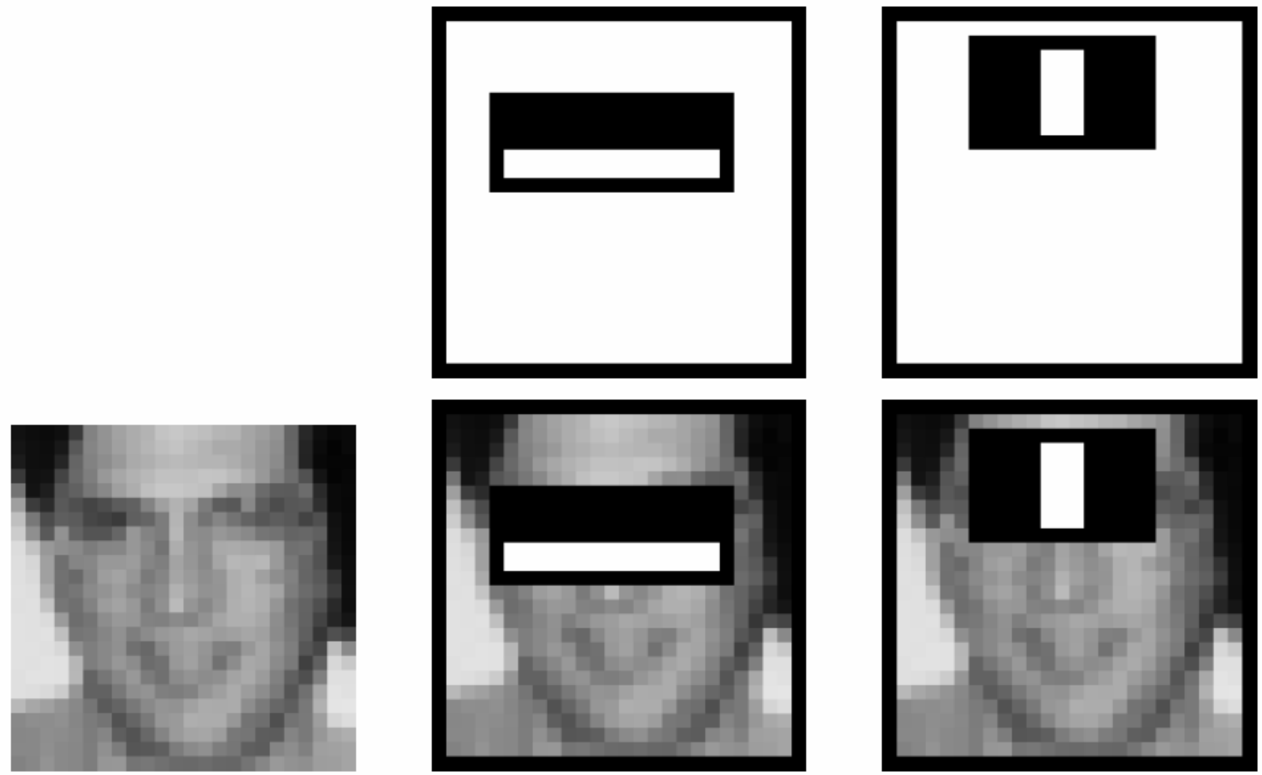
- Advantages of boosting
  - Integrates classification with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement
- Disadvantages
  - Needs many training examples
  - Often doesn’t work as well as SVM (especially for many-class problems)

Boosting for face detection

- Define weak learners based on rectangle features
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

Boosting for face detection

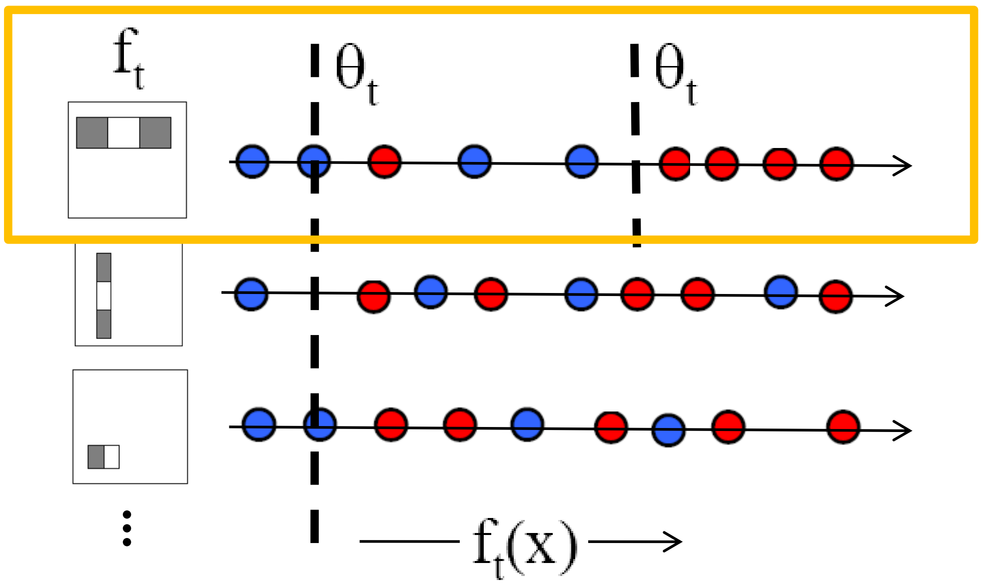
- First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

Viola-Jones detector: AdaBoost

- **Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.**



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

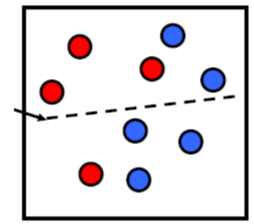
For next round, reweight the examples according to errors, choose another filter/threshold combo.

Kristen Grauman

AdaBoost Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

Evaluate *weighted* error for each feature, pick best.

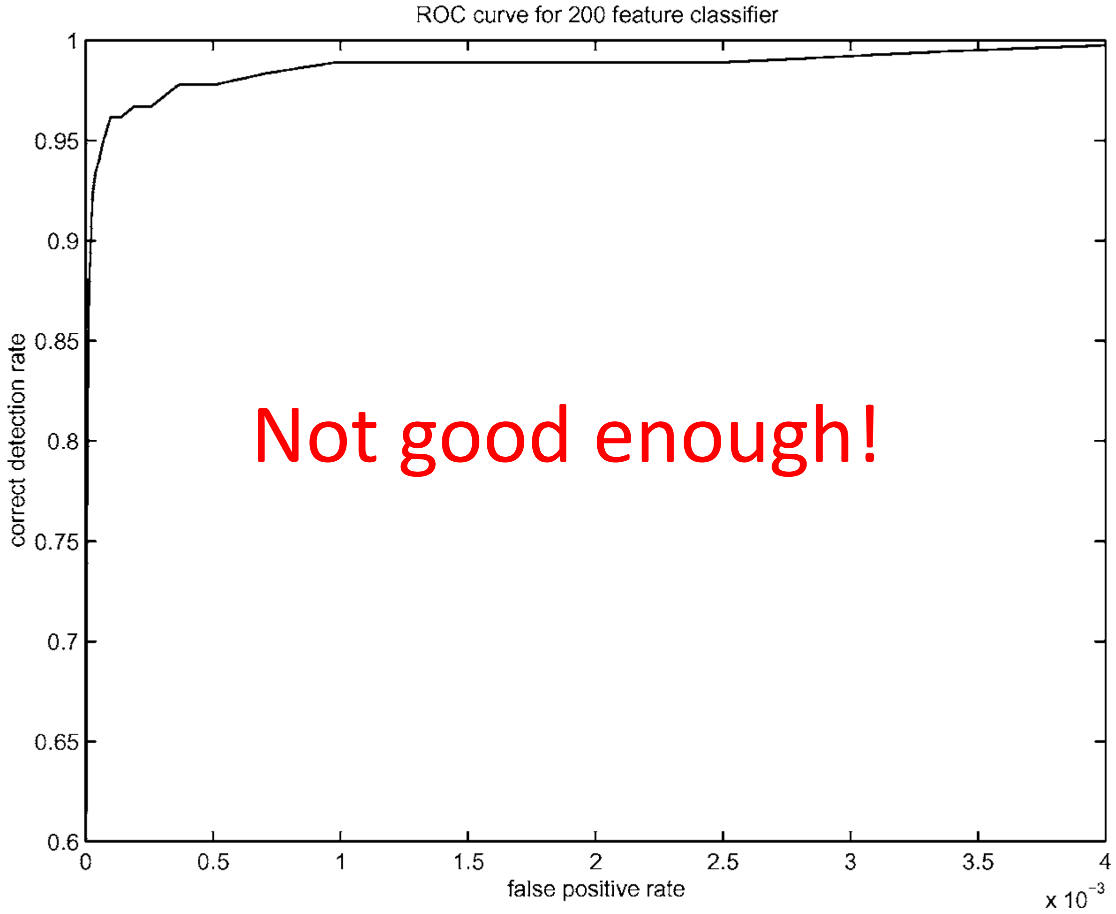
Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

Boosting for face detection

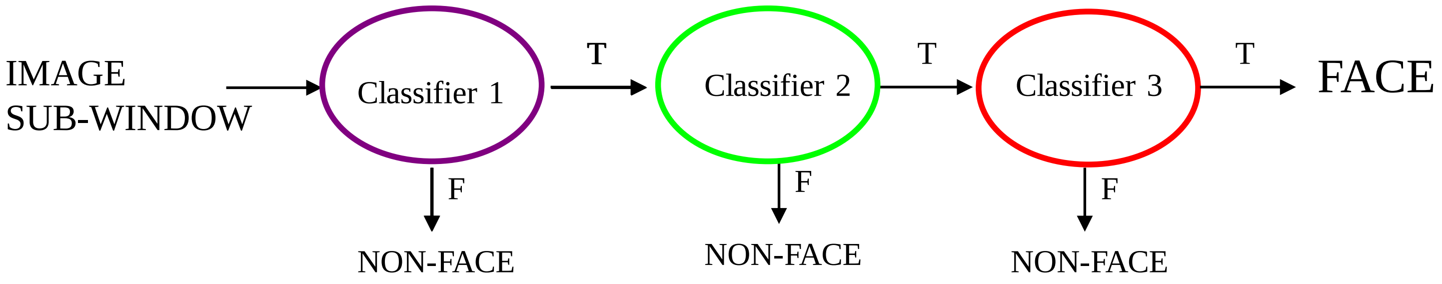
- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084



Receiver operating characteristic (ROC) curve

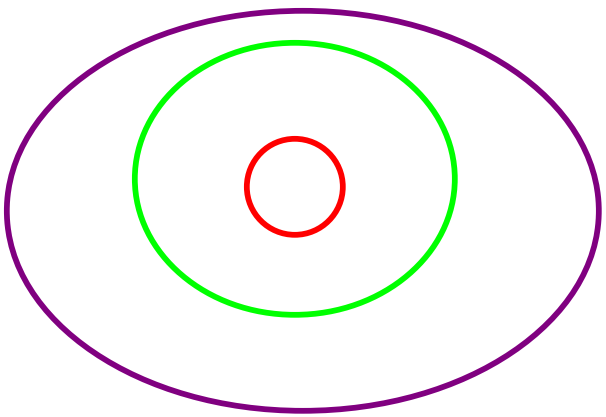
Attentional cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

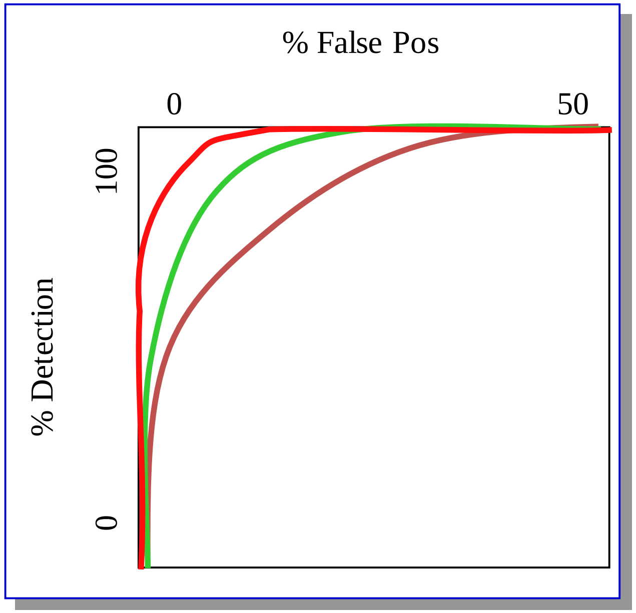


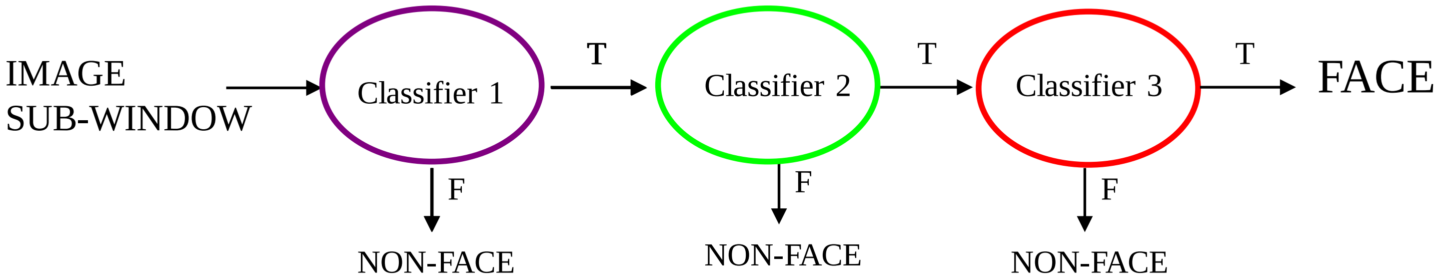
Attentional cascade

- Chain classifiers that are progressively more complex and have lower false positive rates:



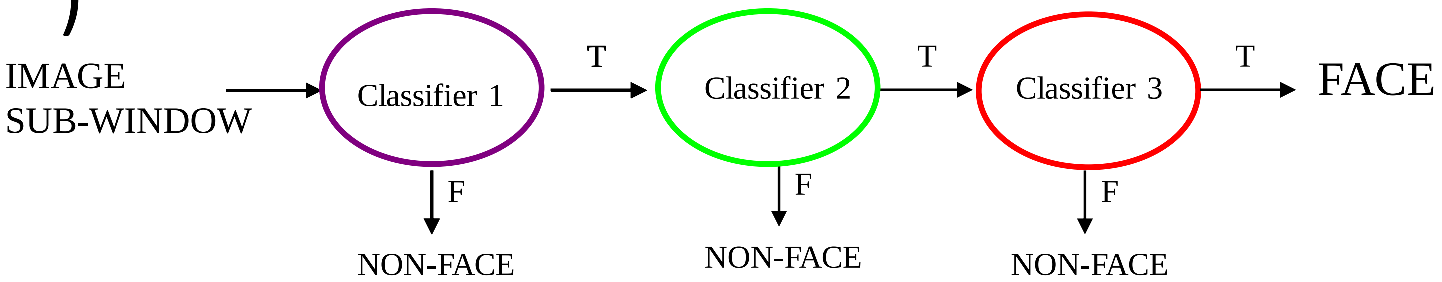
Receiver operating characteristic





Attentional cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6\times10^{-6}$)



Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

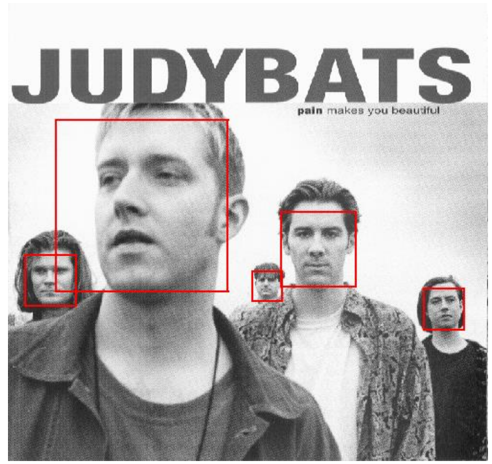
The implemented system

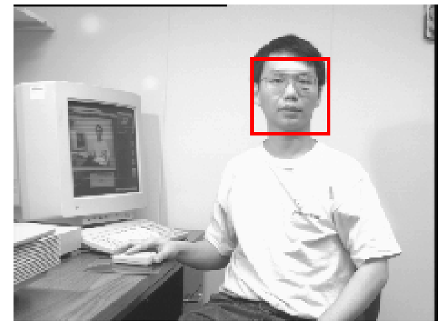
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

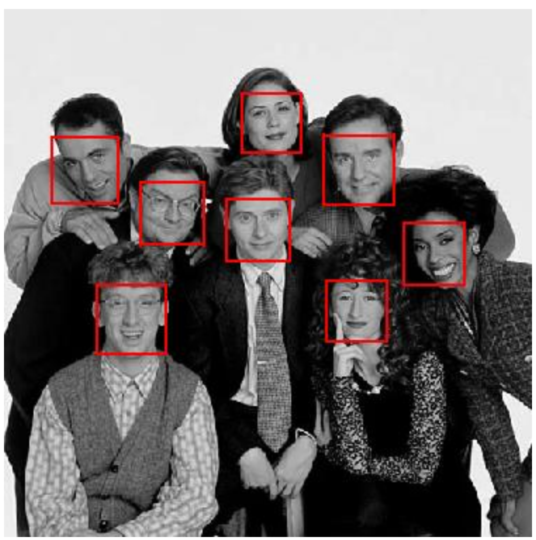
System performance

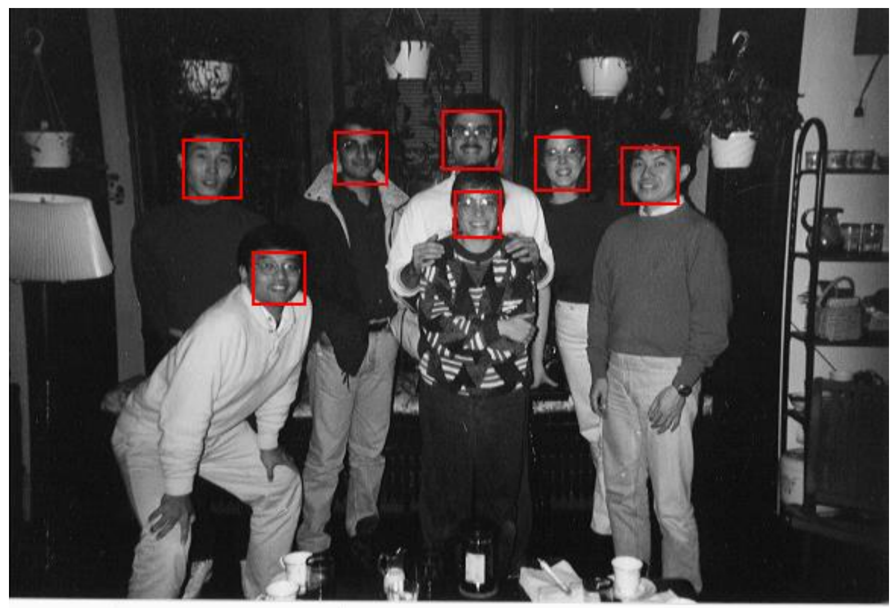
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Output of Face Detector on Test Images



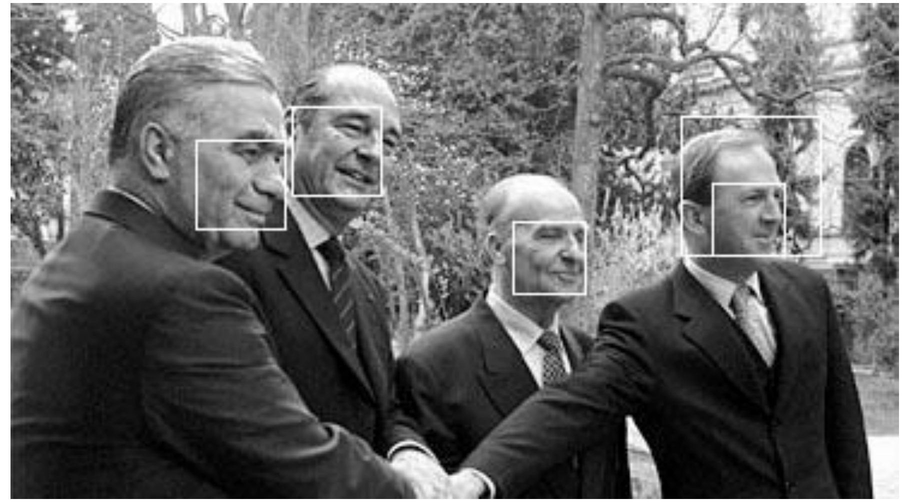






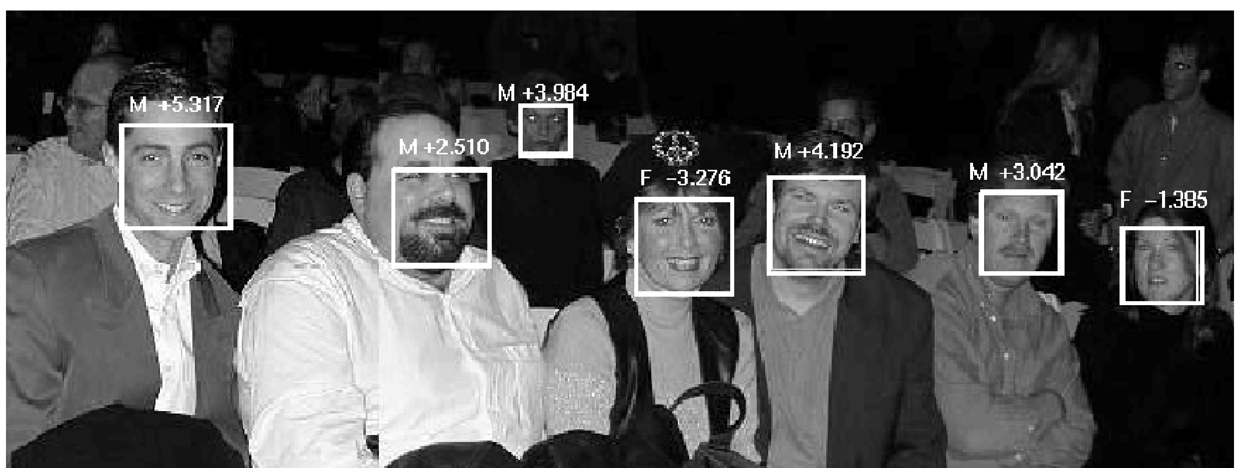
Other detection tasks

Facial Feature Localization

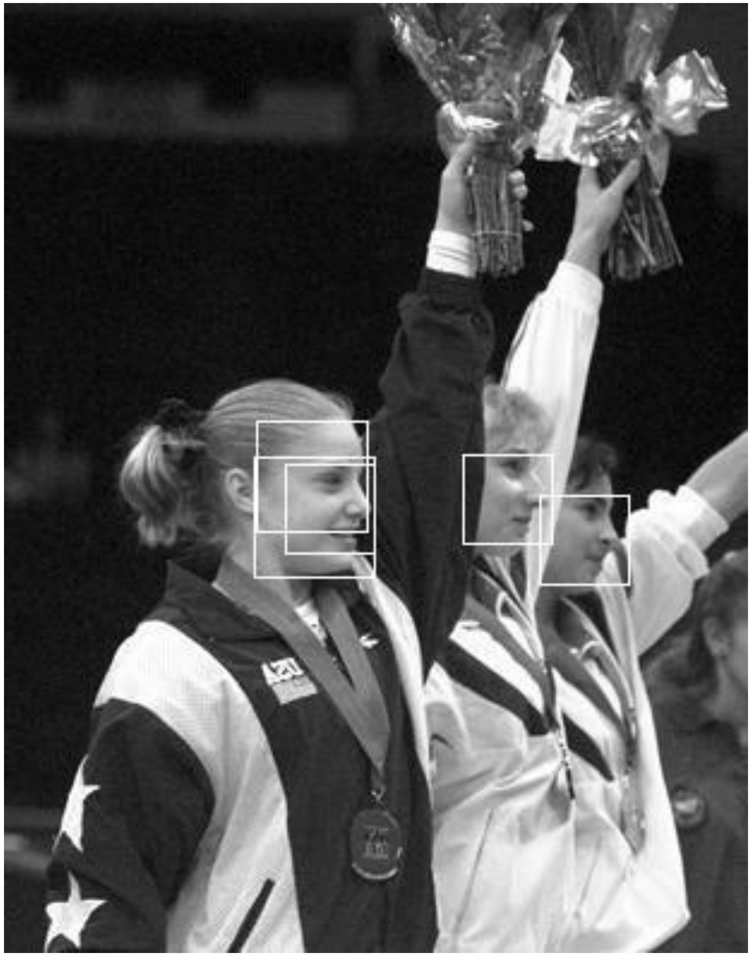


Profile Detection

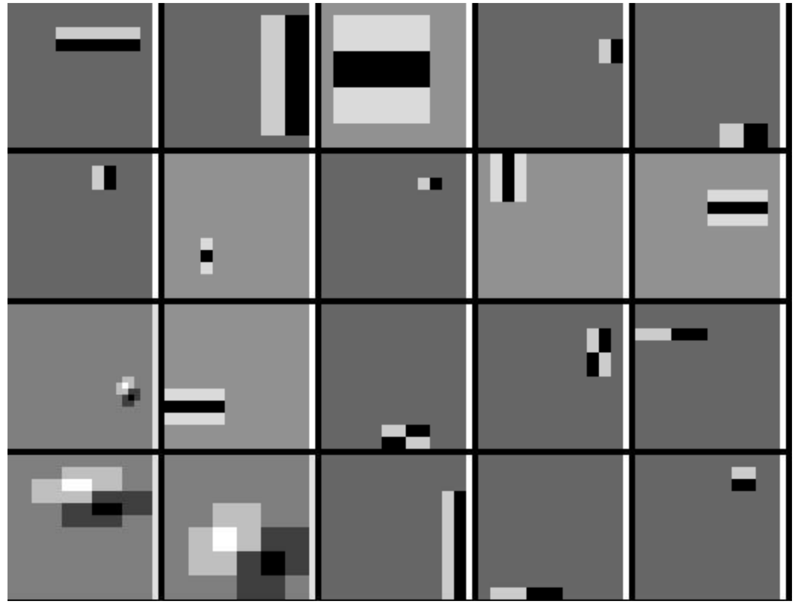
Male vs. female



Profile Detection



Profile Features

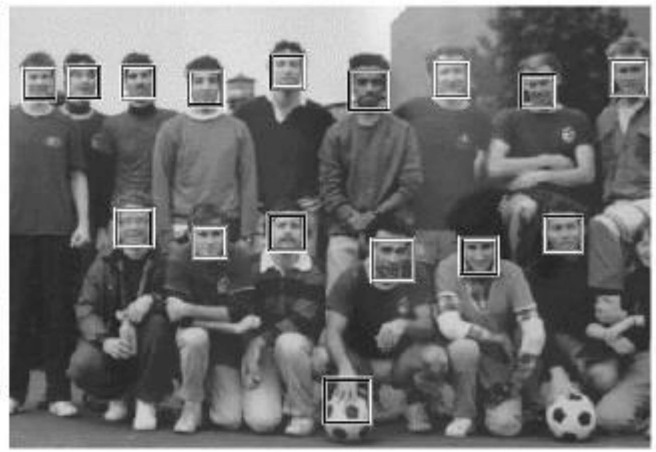


Viola-Jones details

- 38 stages with 1, 10, 25, 50 … features
  - 6061 total used out of 180K candidates
  - 10 features evaluated on average
- Training Examples
  - 4916 positive examples
  - 10000 negative examples collected after each stage
- Scanning
  - Scale detector rather than image
  - Scale steps = 1.25 (factor between two consecutive scales)
  - Translation 1*scale (# pixels between two consecutive windows)
- Non-max suppression: average coordinates of overlapping boxes
- Train 3 classifiers and take vote

Viola Jones Results

Speed = 15 FPS (in 2001)


Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset

Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows