Face Detection

P J Narayanan

CS5765. Computer Vision. Spring 2013

CVIT, IIIT, Hyderabad

Face Detection

The Face Detection problem:

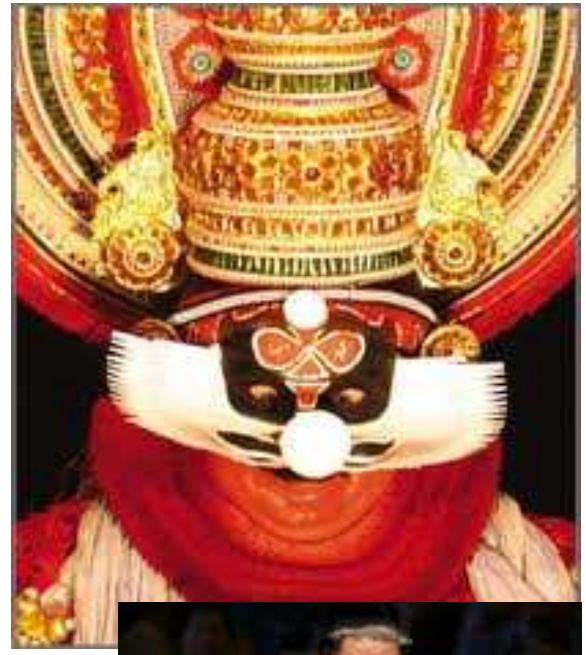
- Tell if a given image contains a human face
- If yes, draw a box around the face(s) to indicate the location

Why is this hard?

- Variation in appearance
- Variation in viewpoints
- Variation in scale
- Occlusion
- Intra-class variations

Different Cases

Different Cases



Why Face Detection

- Surveillance, Human-Computer Interaction, etc.
- First step towards face recognition, which has uses in biometrics, surveillance, etc.
- Intelligent cameras
- ....

Face detection and recognition have been among the most active research areas in Computer Vision for a long time.

Face Detection: How?

- Using skin colour
  - Identify regions with skin colour
  - Look for oval-like shape ...
  - Skin colour has narrow distribution in hue space: Learn a GMM from examples
- Using facial features
  - Detect eye, mouth, nose, eyebrows, etc.
  - Could be guided by skin colour
  - Relationship between them to confirm faces

Viola-Jones Face Detector

CVPR 2001, IJCV 2004

VJ Face Detector

- A very popular real-time face detector. A highly successful research work, appeared in CVPR 2001.
- Designed for frontal faces with decent appearance
- High training cost/time, but ultrafast detection
- Incorporated into digital cameras, etc.
- Framework proposed is generic and can be tuned to detect any object.
- OpenCV implementation by Lienhart is available

Overall Idea

- Look for a face in every window of every size (greater than a minimum)
- A really **HUGE** number of such windows!
  - For $w \times h$ windows with shift of δ_x and δ_y,

$$\sum_{w=m}^{M} \sum_{h=m}^{N} \frac{(M-w+1)(N-h+1)}{\delta_x \ \ \delta_y} = 14352998416$$

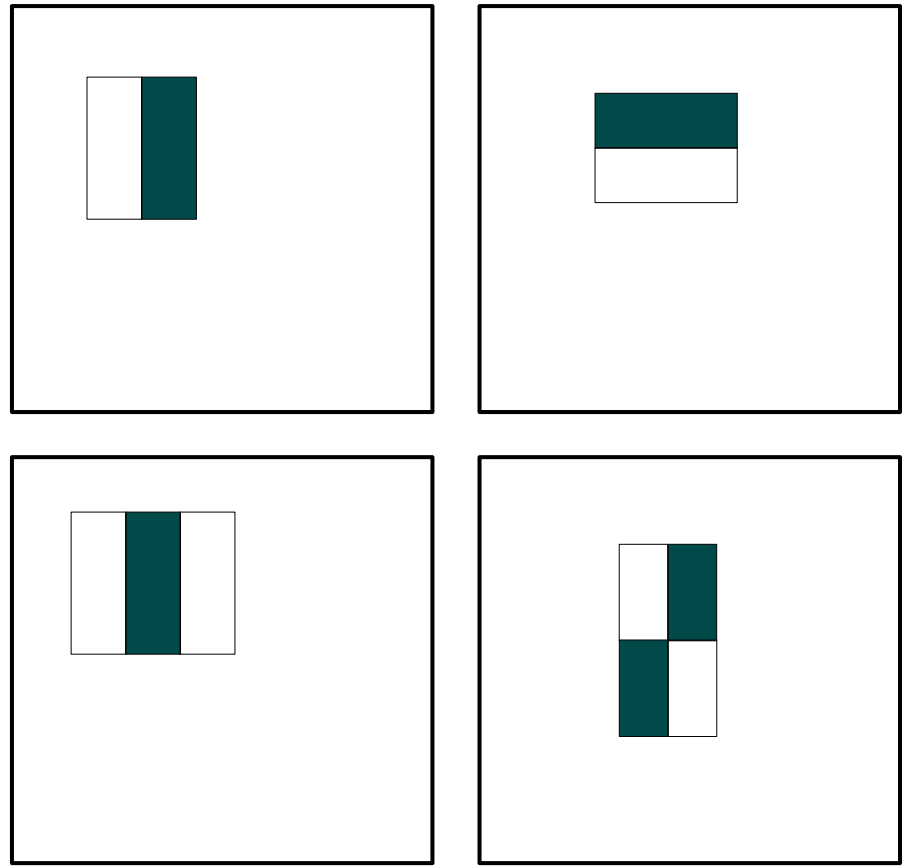
  - 14B for 512x512 image, 24x24 minimum, 1x1 shift
- Faces are **rare** (even in a group picture!)
- Find the faces and reject others rapidly
  - Use a small number of quick-to-compute features
  - Be smart: Spend very small time on windows without promise

Innovative Ideas

- Simple Haar-like features based on sums of rectangular regions
  - Integral images to compute them very quickly
  - Constant time to sum rectagular regions
- AdaBoost learning to design simple classifiers using a small number of features
- Cascade of classifiers to achieve good **true positive rate (TPR)** and acceptable **false positive rate (FPR)**
  - Each has high FPR, but very low false negative rate
  - Good TPR and low FPR by combining several such "weak" classifiers

Haar-Like Features

- Sum/differences of rectangular regions
- Only 4 features used
  - Specified by the type and opposite corners
  - Any location, size
- Sum of light regions subtracted from the sum of the dark regions
- Base window: 24×24
  - Over 1,60,000 features possible



Integral Image

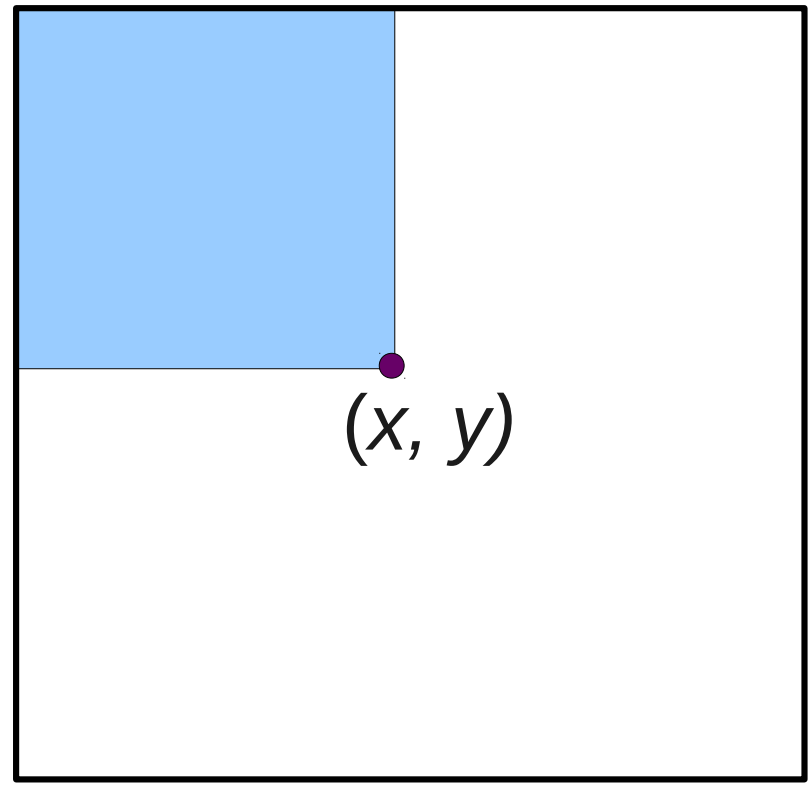
Each pixel position of the **integral image** is the sum of all pixels above and to the left of itself

- $\mathcal{I}(x,y) = \sum_{i \le x, j \le y} I(i,j)$

- Sum along the rows and then along the columns

$$s(x,y) = s(x,y-1) + I(x,y)$$
$$\mathcal{I}(x,y) = \mathcal{I}(x-1,y) + s(x,y)$$

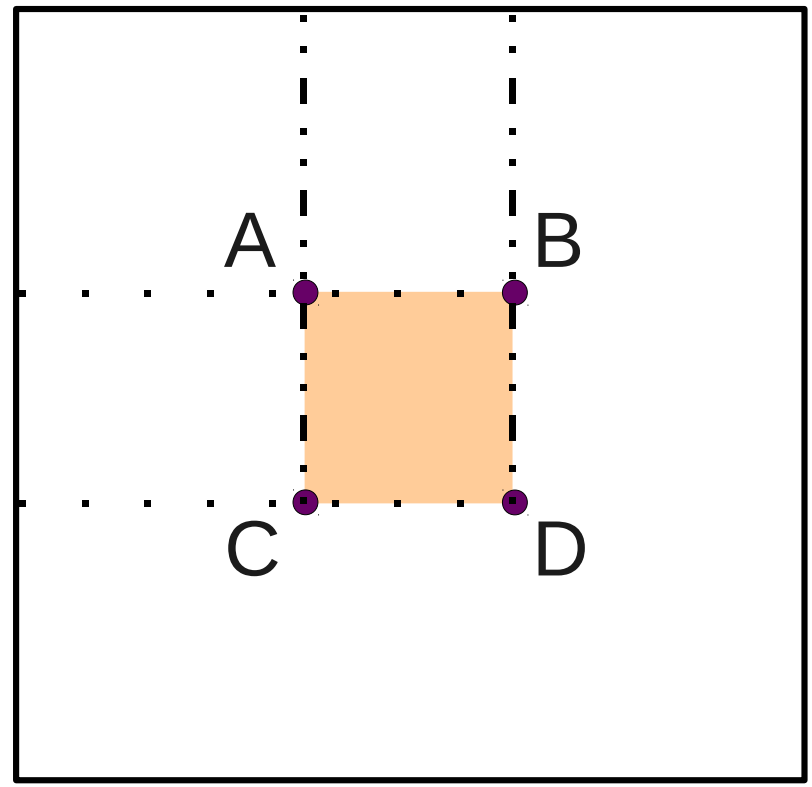
- Called **summed area tables** in graphics; used for mip-map texturing



Finding a Rectangle Sum

Sum of rectangular regions can be computed very efficiently using integral images.

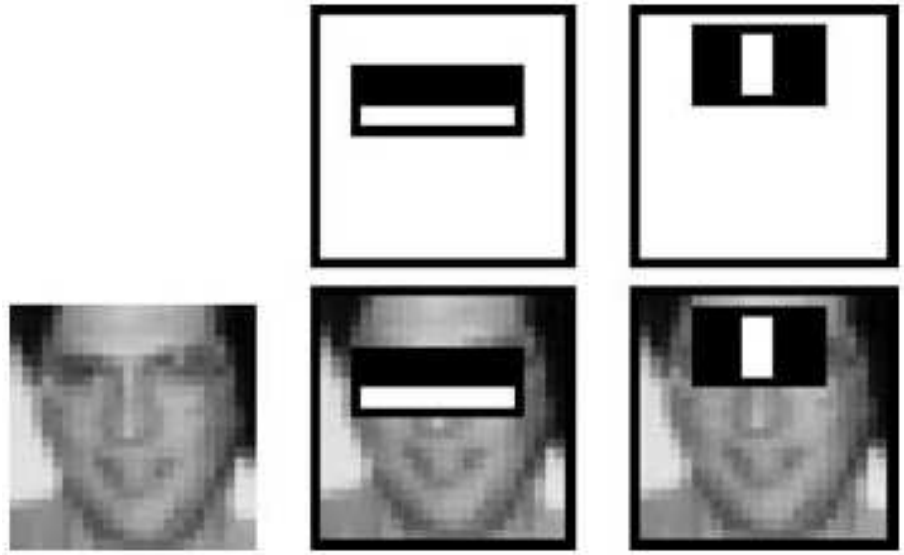
- $\mathit{RectSum}(A, B, C, D) = \mathcal{I}(D) - \mathcal{I}(C) - \mathcal{I}(B) + \mathcal{I}(A)$
- Exactly 4 accesses to the integral image
- Exactly 3 add/subtract operations
- Independent of the size of rectangle!



Integral image can be computed in $O(MN)$ steps!

What can such features do?

Two simple features and a perceptron classifier



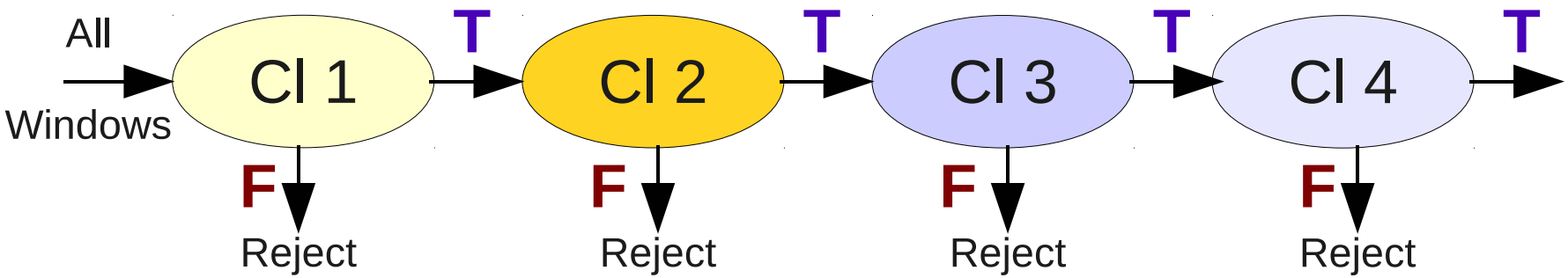
can get 50% false positive rate and 100% detection with a suitable threshold.

Looks for possible eye regions with darker and lighter areas

Half of non-promising regions eliminated while keeping **all** of the faces

Classification: Key Idea

- Use several such **weak** classifiers with very low FNR
- Windows rejected by any are discarded completely
- Rest are sent to the next classifier
- A *cascade* of such classifiers
- Early ones are simple; later ones more complicated
- Only a few reach the more expensive stages



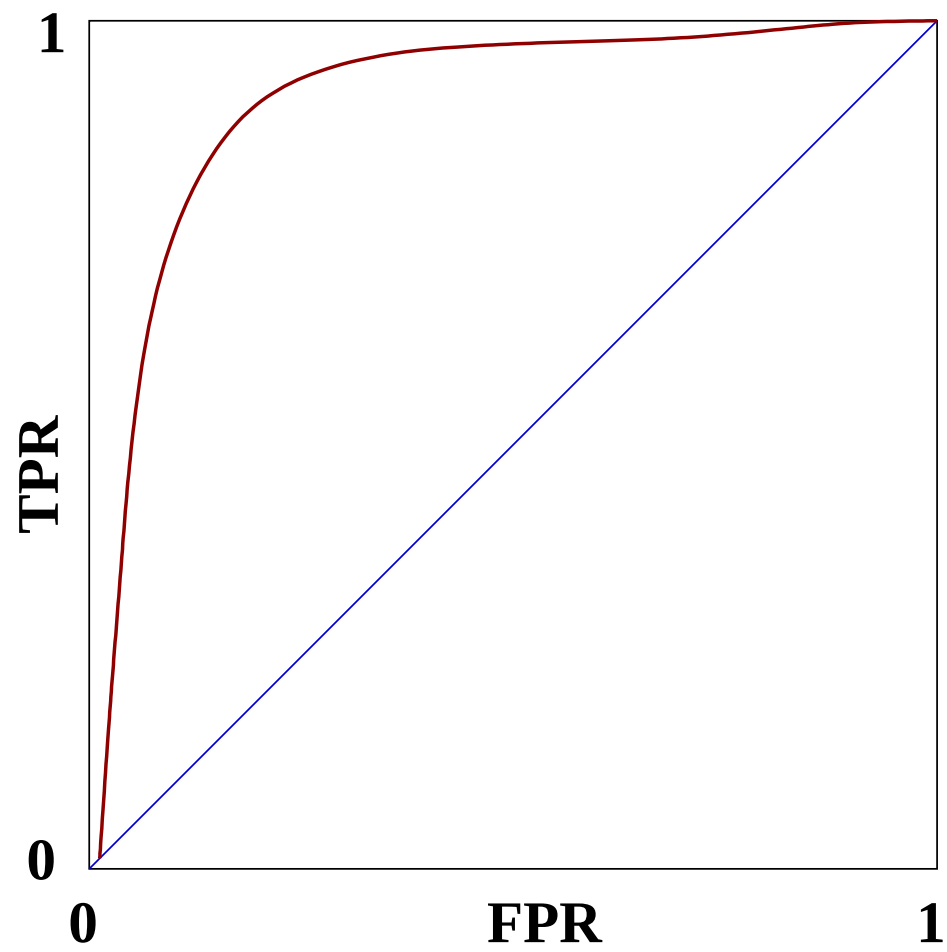
Windows that reach the end are faces!!

High Detection and FPR

High detection or true positive ratio if moderate false positives can be tolerated!

Cannot tolerate many *false negatives*. Windows rejected by any stage is rejected for ever.

Tune parameters (weights, threshold) to achieve high TPR with moderate FPR.



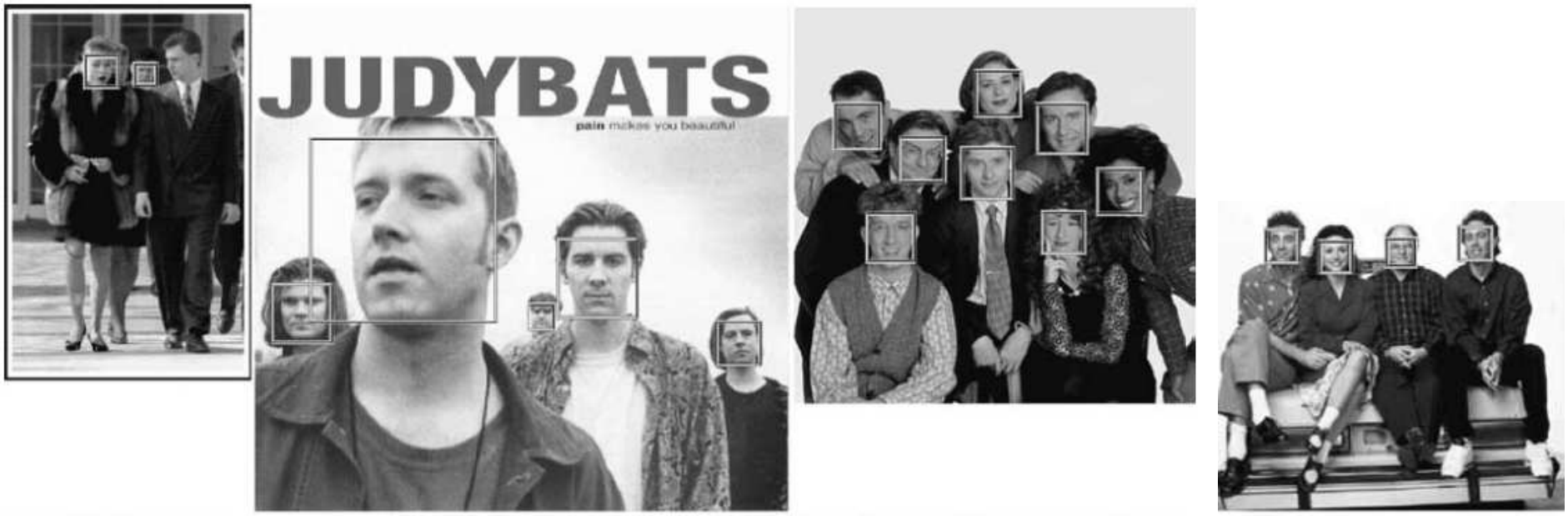
How does it work?

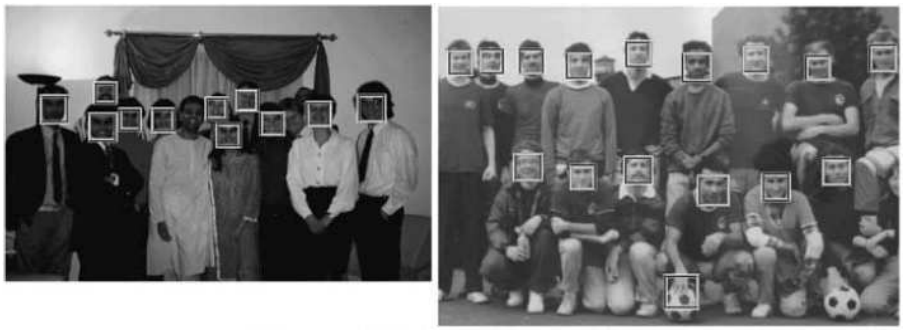
- Assume an image with 1 million sub-windows and 100 faces. Classifier has an FPR of 0.5 and a FNR of 10^{-4} in each stage.
  - Stage 1: 500K windows remain. 10^{-2} faces rejected
  - Stage 2: 250K windows remain. Some rejected
  - Stage 3: 125K windows remain. Some rejected
- Assume k stages with f_i and d_i the false positive and detection rate of stage i
- Overall detection and false positive rates:

$$D = \prod_{i=1}^{k} d_i \quad \text{and} \quad F = \prod_{i=1}^{k} f_i$$

High D needs fewer stages. Low F needs many stages!

Some Results





VJ System: Details

- Minimum face window: 24×24
- 38 stage cascade.
- Features in first 5 stages: 1, 10, 25, 25, 50 (CVPR01)
  IJCV 2004: 2 (50% FPR), 10 (80%), 25, 25, 50
- Overall detection rate: 95% and FPR: 0.02%
- A total of 6060 features used (out of 160K)
- Average of 10 [8] features evaluated per sub-window
- Face detection speed: 15 fps
  (67ms for 384x288 images on 700MHz Pentium III)
- Training: 4916 annotated faces and 10000 non-faces
- Time to train the cascade: **weeks!**

Positive Training Examples

Viola-Jones at Work

Courtesy: Gary Bishop, ECCV04

Viola-Jones at Work

Courtesy: Gary Bishop, ECCV04

Viola-Jones at Work

Courtesy: Gary Bishop, ECCV04

Viola-Jones at Work

Courtesy: Gary Bishop, ECCV04

Viola-Jones at Work

Courtesy: Gary Bishop, ECCV04

Classifiers and AdaBoost Training

Viola-Jones Face Detector

Classifier Nodes

- Simple perceptrons that combine weighted feature values
- Stage i decision: $h_i = \delta\, f_i(x) < \delta\, \theta$, where $\delta = \pm 1$ controls the comparison, x is the 24x24 window.
- $f_i(x)$ is the perceptron function involving features derived from image window.
- Use AdaBoost to learn each $f_i(x)$ until a target detection and false positive rates are met.

Training the Cascade

- User sets an overall target false positive rate F_t. And for each stage, the maximum acceptable FPR f, and the minimum acceptable detection rate d.
- If the FPR of cascade is above F_t, add another stage ot the cascade.
- The negative examples for the new stage are the negative examples of the cascade so far on the non-face images. Train later stages on weaknesses of previous ones!
- Train the stage by selecting more features using the AdaBoost algorithm.
- Adjust the threshold to meet the detection rate.

Examples: positive (faces) P, negative (non-faces) N.

Cascade Training Algorithm

Face examples P, non-faces N, $i \leftarrow 0,\ F_0 \leftarrow 1,\ D_0 \leftarrow 1$

while $(F_i > F_t)$ // Global FPR not yet met

1. $i \leftarrow i + 1,\ \ n_i \leftarrow 0,\ \ F_i \leftarrow F_{i-i}$
2. while $(F_i > f * F_{i-1})$ // Stage f too high
   (a) $n_i \leftarrow n_i + 1$ // Increase #features
   (b) Use positive set P and negative set N to train a classifier with n_i features using AdaBoost
   (c) Compute F_i and D_i by applying the current cascade on a validation set.
   (d) Reduce threshold of current stage until the overall detection rate is at least $d * D_{i-1}$
3. if $(F_i > F_t)$ form a new negative set N as the detections of the cascade on non-faces images.

Training each Stage

- Consider multiple "weak" classifiers involving a single feature each.
- Find the best weak classifier that produces the least error on the samples.
- Increase the weights of previously misclassified examples for later weak classifiers.
- A classifier: $h(x, f, p, \theta) = [pf(x) < p\theta]$, where $f(x)$ is the perceptron function involving features from image x, θ the threshold, and p the polarity of comparison.

Given labelled samples (x_i, y_i) where $y_i = \{0, 1\}$ with m 0-examples and l 1-examples.

$w_{1i} \leftarrow 1/2m$ for 0-examples and $w_{1i} \leftarrow 1/2l$ for 1-examples.

AdaBoost Algorithm

for $t = 1, 2, \cdots, T$ // T features used

1. $w_{ti} \leftarrow w_{ti} / \sum_j w_{tj}$ // Normalize weights across samples
2. Select best weak classifier that minimizes the error

   // Exhaustive search across features/weights for best perceptron!

$$\epsilon_t = \min_{f,p,\theta} \sum_i w_{ti} \, |h(x_i, f, p, \theta) - y_i|$$

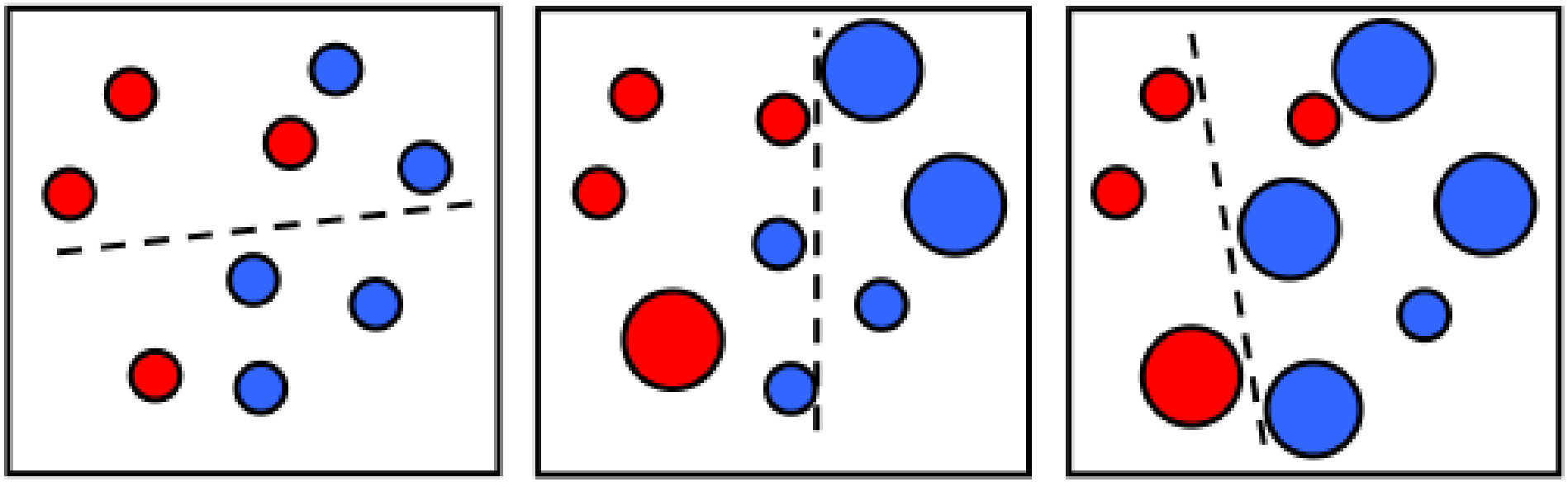
3. Set $h_t(x) = h(x, f_t, p_t, \theta_t)$ from the above step
4. Update weights: $w_{(t+1)i} = w_{ti} \, \beta_t^{1-e_i}$ where $e_i = 1$ for incorrectly classfied examples and $\beta_t = \epsilon_t / (1 - \epsilon_t)$

Final classifier with $\alpha_t = \log(1/\beta_t)$:

$$h(x) \quad = \quad \left(\sum_{t=1}^{T} \alpha_t h_t(x) \quad \geq \quad \frac{1}{2} \sum_{t=1}^{T} \alpha_t \right)$$

AdaBoost: Discussion

- $\epsilon_i < 1$ as only misclassified samples contribute
- Weights for next round is same for incorrect samples and lower for others.



Viola-Jones: Other Details

Image processing: Variance normalize each window to eliminate changes in illumination

- $\sigma^2 = \sum (I_i - \mu)^2 = \left(\sum I_i^2\right) - N\left(\sum I_i\right)$
- Evaluate this efficiently using integral image of I_i^2 also
- $I_i' = (I_i - \mu)/\sigma$. Transform $\sum I'$ values or the features!!

Different Scales: Scale of 1.25 works best. For computation, *scale the features up by 1.25* instead of scaling the image down!

Scanning: The window is translated to other positions. High Δ can be faster, but can miss faces. Step size of 1 is used ($s\Delta$ at scale s). Results in multiple detections for the same face. **Non-Maxima Suppression** necessary for all sliding window methods.

Viola-Jones: Conclusions

- Previous face detectors (Rowley 1996, Schneiderman 2000) worked quite well for detection
- Viola-Jones made it real time, by speeding it up more than 10 times!
- Provided a framework to construct several fast detectors. They built a person detector in ICCV 2003
- They had introduced AdaBoost to computer vision for image retrieval in 1999.
- An influential work and a useful framework!

Thank You!

Thanks to several public web sources for many figures!!