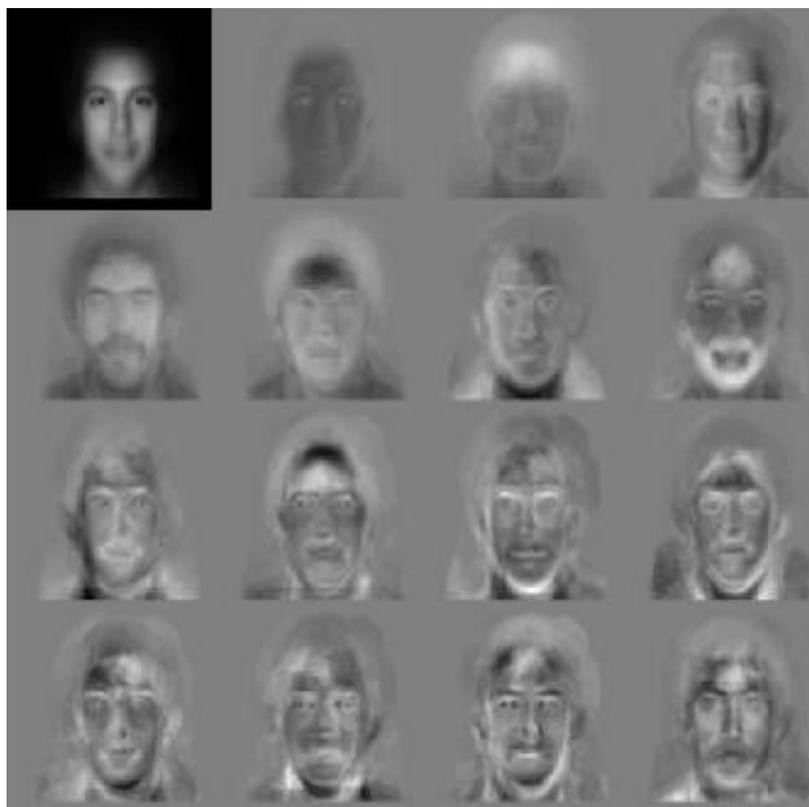


CS6670: Computer Vision

Noah Snavely

Lecture 15: Eigenfaces



Announcements

- Final project page up, at
 - <http://www.cs.cornell.edu/courses/cs6670/2009fa/projects/p4/>
 - One person from each team should submit a proposal (to CMS) by tomorrow at 11:59pm

- Project 3: Eigenfaces

Skin detection results

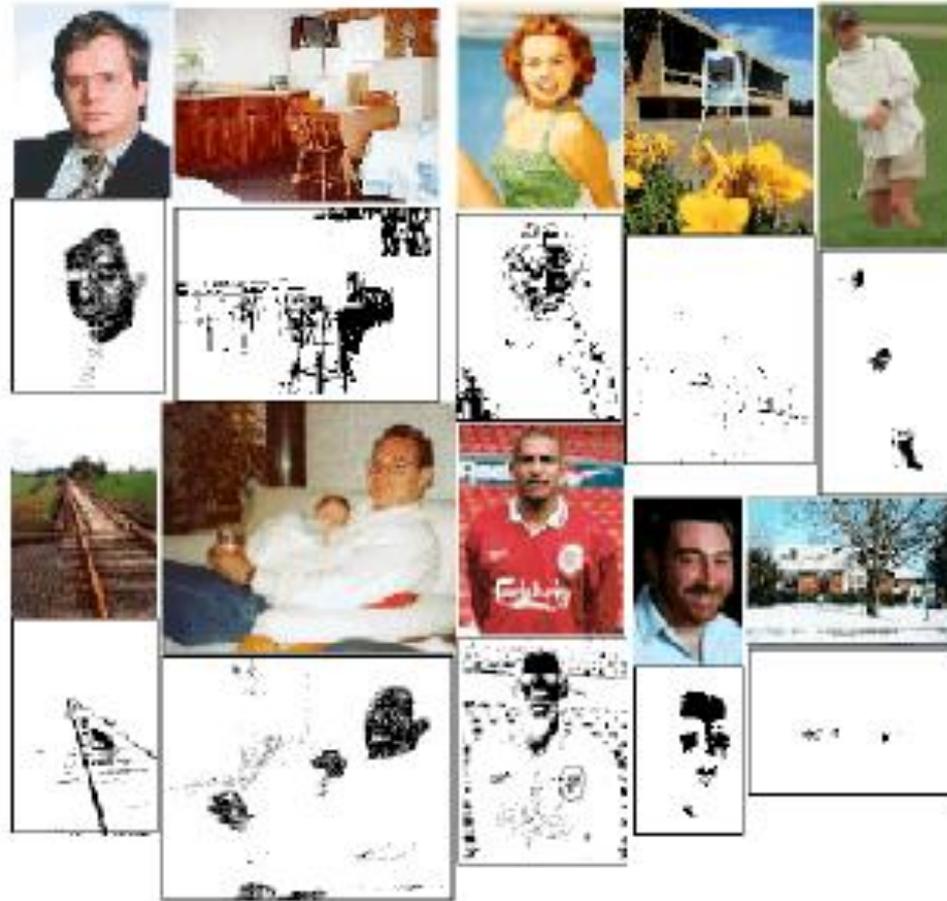
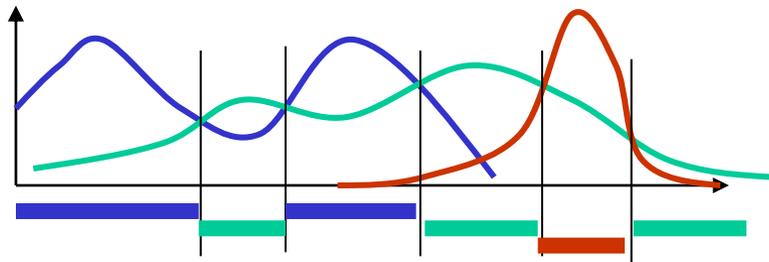


Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

General classification

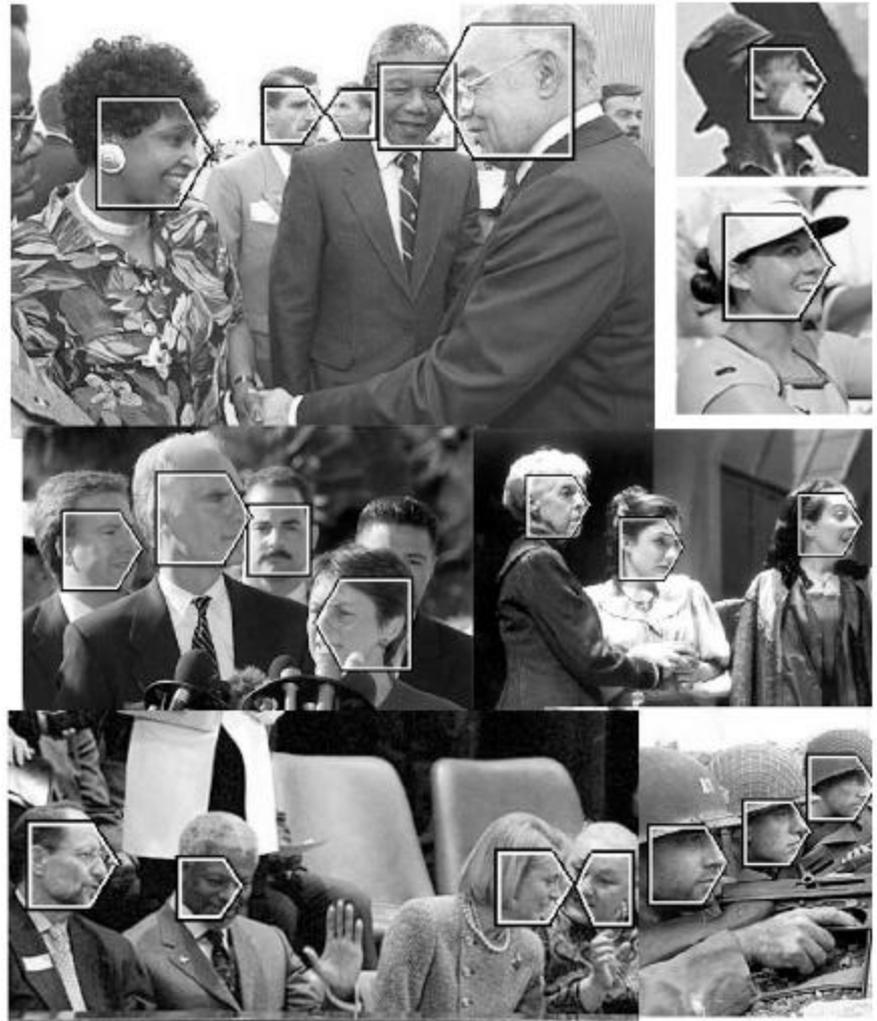
This same procedure applies in more general circumstances

- More than two classes
- More than one dimension

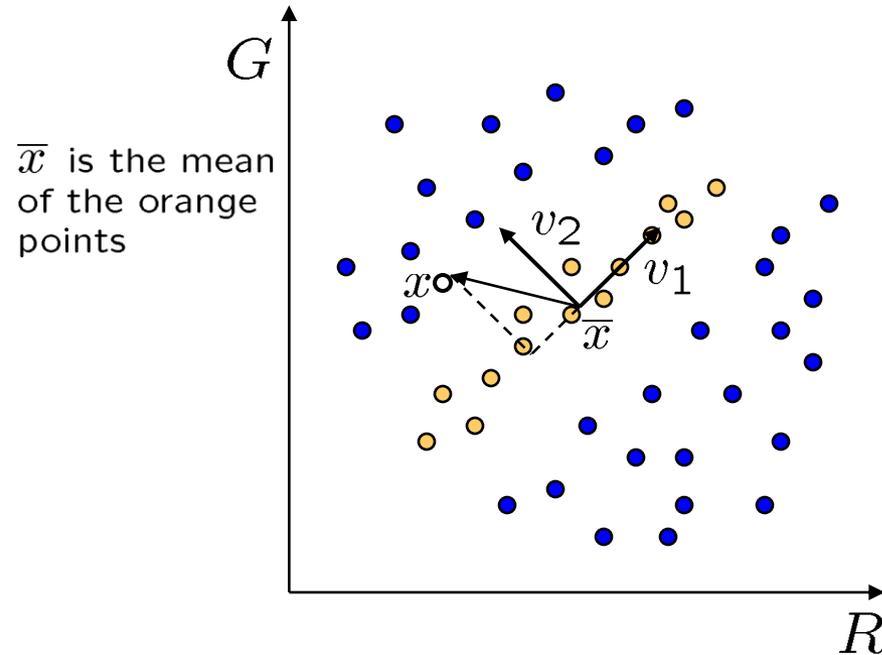


Example: face detection

- Here, X is an image region
 - dimension = # pixels
 - each face can be thought of as a point in a high dimensional space



Linear subspaces



convert \mathbf{x} into $\mathbf{v}_1, \mathbf{v}_2$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_1, (\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_2)$$

What does the \mathbf{v}_2 coordinate measure?

- distance to line
- use it for classification—near 0 for orange pts

What does the \mathbf{v}_1 coordinate measure?

- position along line
- use it to specify which orange point it is

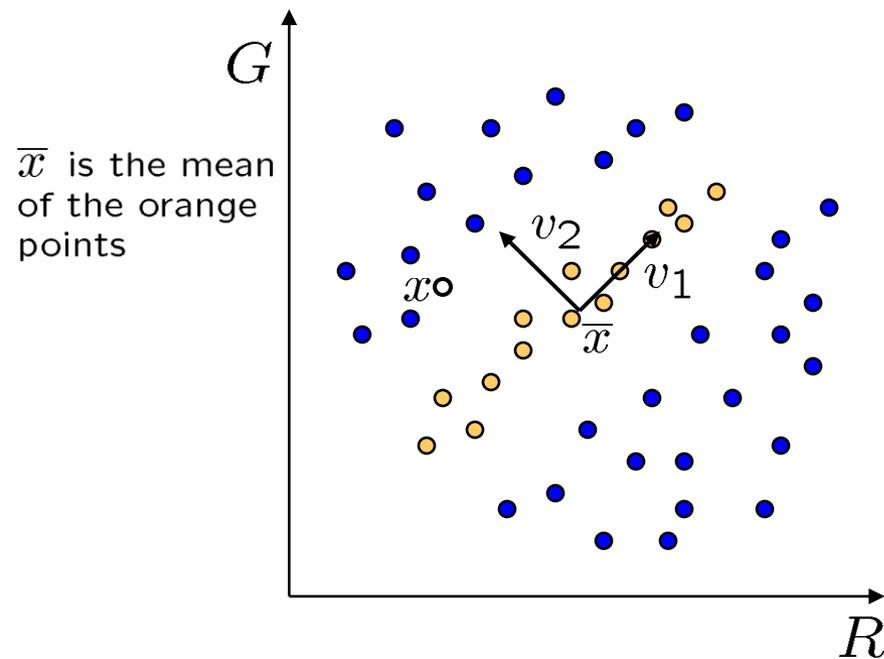
Classification can be expensive

- Must either search (e.g., nearest neighbors) or store large PDF's

Suppose the data points are arranged as above

- Idea—fit a line, classifier measures distance to line

Dimensionality reduction

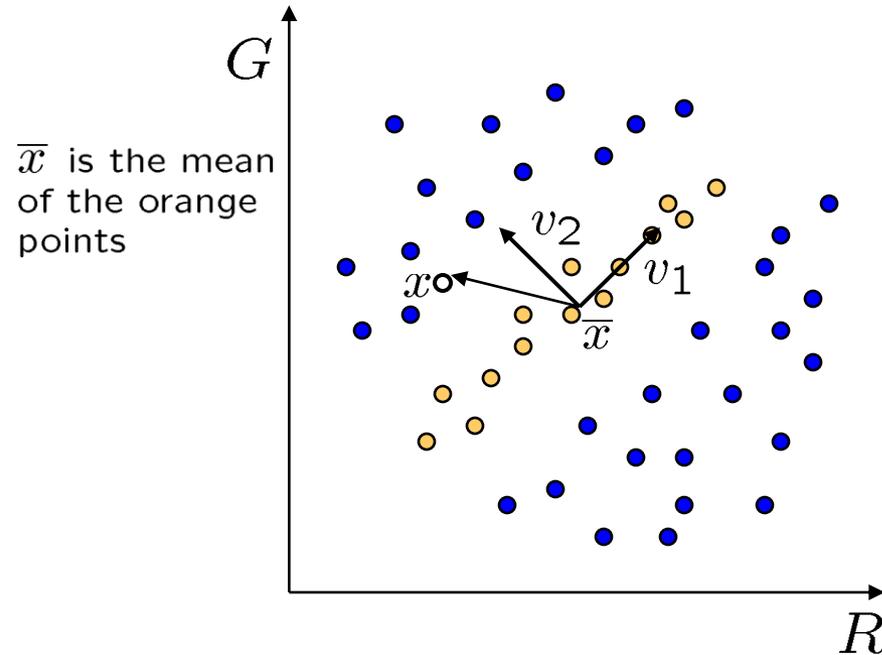


How to find \mathbf{v}_1 and \mathbf{v}_2 ?

Dimensionality reduction

- We can represent the orange points with *only* their \mathbf{v}_1 coordinates
 - since \mathbf{v}_2 coordinates are all essentially 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems

Linear subspaces



Consider the variation along direction \mathbf{v} among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \bar{\mathbf{x}})^T \cdot \mathbf{v}\|^2$$

What unit vector \mathbf{v} minimizes var ?

$$\mathbf{v}_2 = \min_{\mathbf{v}} \{var(\mathbf{v})\}$$

What unit vector \mathbf{v} maximizes var ?

$$\mathbf{v}_1 = \max_{\mathbf{v}} \{var(\mathbf{v})\}$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \bar{\mathbf{x}})^T \cdot \mathbf{v}\|^2 \\ &= \sum_{\mathbf{x}} \mathbf{v}^T (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{v} \\ &= \mathbf{v}^T \left[\sum_{\mathbf{x}} (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T \right] \mathbf{v} \\ &= \mathbf{v}^T \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^T \end{aligned}$$

Solution: \mathbf{v}_1 is eigenvector of \mathbf{A} with *largest* eigenvalue
 \mathbf{v}_2 is eigenvector of \mathbf{A} with *smallest* eigenvalue

Principal component analysis

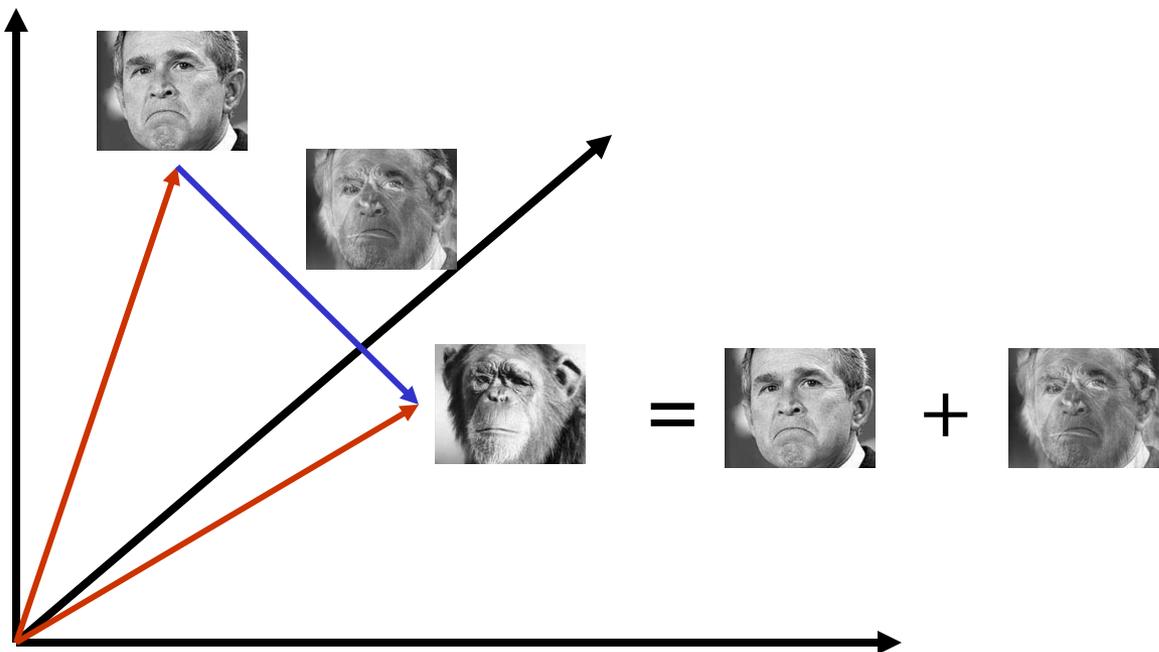
Suppose each data point is N-dimensional

- Same procedure applies:

$$\begin{aligned} \text{var}(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \bar{\mathbf{x}})^T \cdot \mathbf{v}\|^2 \\ &= \mathbf{v}^T \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T \end{aligned}$$

- The eigenvectors of \mathbf{A} define a new coordinate system
 - eigenvector with largest eigenvalue captures the most variation among training vectors \mathbf{x}
 - eigenvector with smallest eigenvalue has least variation
- We can compress the data by only using the top few eigenvectors
 - corresponds to choosing a “linear subspace”
 - » represent points on a line, plane, or “hyper-plane”
 - these eigenvectors are known as the ***principal components***

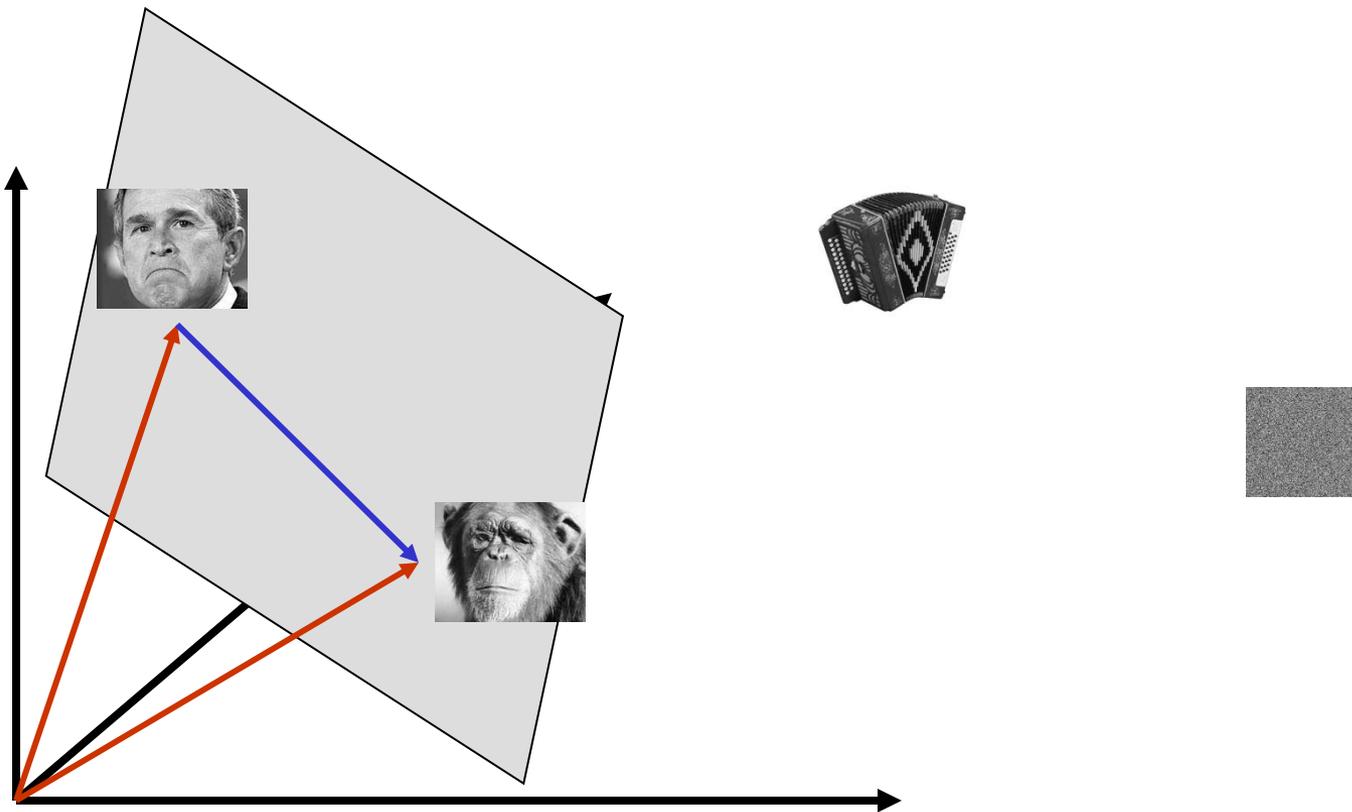
The space of faces



An image is a point in a high dimensional space

- An $N \times M$ intensity image is a point in \mathbb{R}^{NM}
- We can define vectors in this space as we did in the 2D case

Dimensionality reduction



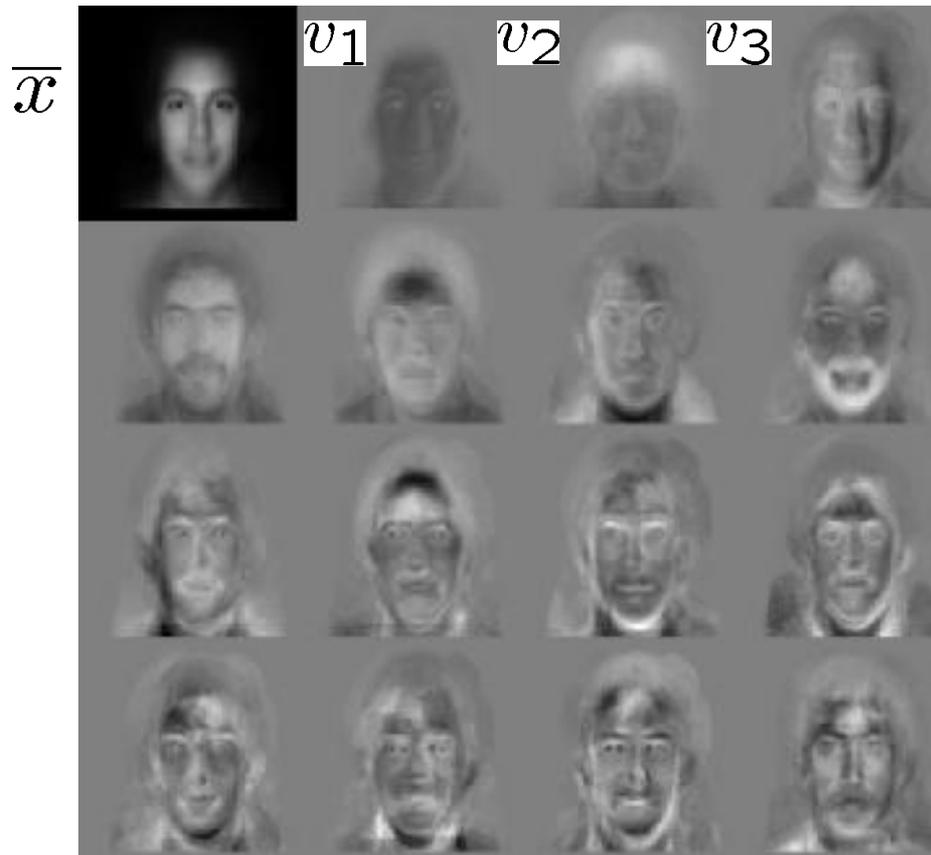
The set of faces is a “subspace” of the set of images

- Suppose it is K dimensional
- We can find the best subspace using PCA
- This is like fitting a “hyper-plane” to the set of faces
 - spanned by vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K$
 - any face $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \dots + a_k\mathbf{v}_k$

Eigenfaces

PCA extracts the eigenvectors of \mathbf{A}

- Gives a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots$
- Each one of these vectors is a direction in face space
 - what do these look like?



Projecting onto the eigenfaces

The eigenfaces $\mathbf{v}_1, \dots, \mathbf{v}_K$ span the space of faces

- A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow \left(\underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_1}_{a_1}, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_2}_{a_2}, \dots, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_K}_{a_K} \right)$$

$$\mathbf{x} \approx \bar{\mathbf{x}} + a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \dots + a_K \mathbf{v}_K$$



\mathbf{X}



$a_1 \mathbf{v}_1$ $a_2 \mathbf{v}_2$ $a_3 \mathbf{v}_3$ $a_4 \mathbf{v}_4$ $a_5 \mathbf{v}_5$ $a_6 \mathbf{v}_6$ $a_7 \mathbf{v}_7$ $a_8 \mathbf{v}_8$



Detection and recognition with eigenfaces

Algorithm

1. Process the image database (set of images with labels)
 - Run PCA—compute eigenfaces
 - Calculate the K coefficients for each image
2. Given a new image (to be recognized) \mathbf{x} , calculate K coefficients

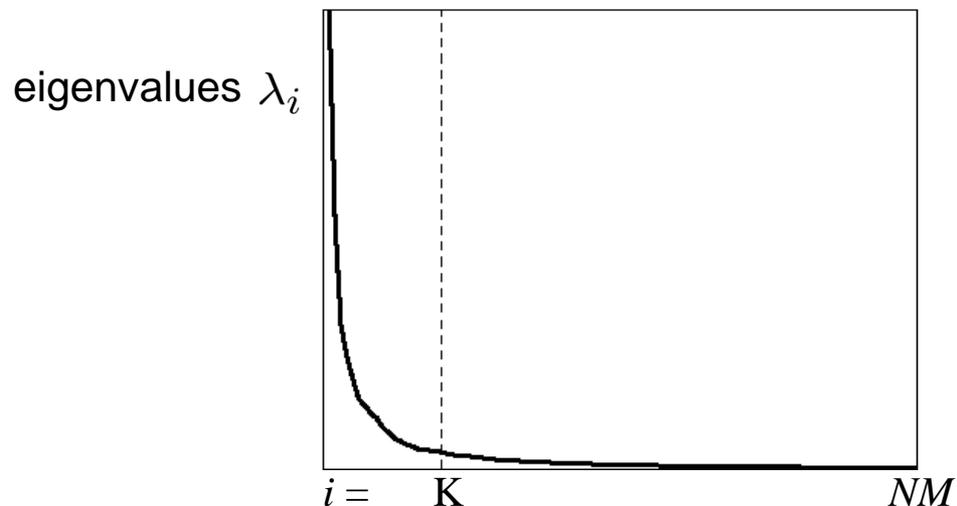
$$\mathbf{x} \rightarrow (a_1, a_2, \dots, a_K)$$

3. Detect if \mathbf{x} is a face

$$\|\mathbf{x} - (\bar{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \dots + a_K\mathbf{v}_K)\| < \text{threshold}$$

4. If it is a face, who is it?
 - Find closest labeled face in database
 - nearest-neighbor in K -dimensional space

Choosing the dimension K



How many eigenfaces to use?

Look at the decay of the eigenvalues

- the eigenvalue tells you the amount of variance “in the direction” of that eigenface
- ignore eigenfaces with low variance

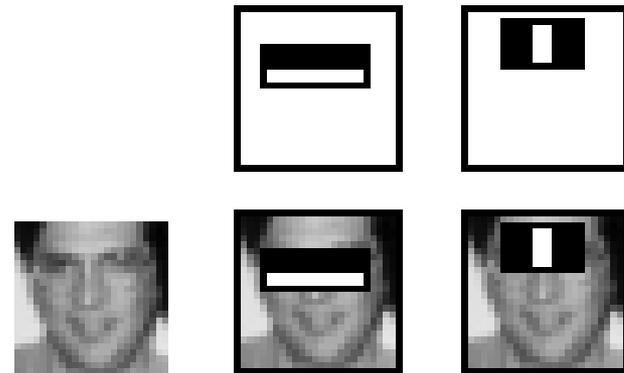
Issues: metrics

What's the best way to compare images?

- need to define appropriate features
- depends on goal of recognition task



exact matching
complex features work well
(SIFT, MOPS, etc.)



classification/detection
simple features work well
(Viola/Jones, etc.)

Metrics

Lots more feature types that we haven't mentioned

- moments, statistics
 - metrics: Earth mover's distance, ...
- edges, curves
 - metrics: Hausdorff, shape context, ...
- 3D: surfaces, spin images
 - metrics: chamfer (ICP)
- ...

Issues: feature selection



If all you have is one image:
non-maximum suppression, etc.



If you have a training set of images:
AdaBoost, etc.

Issues: data modeling

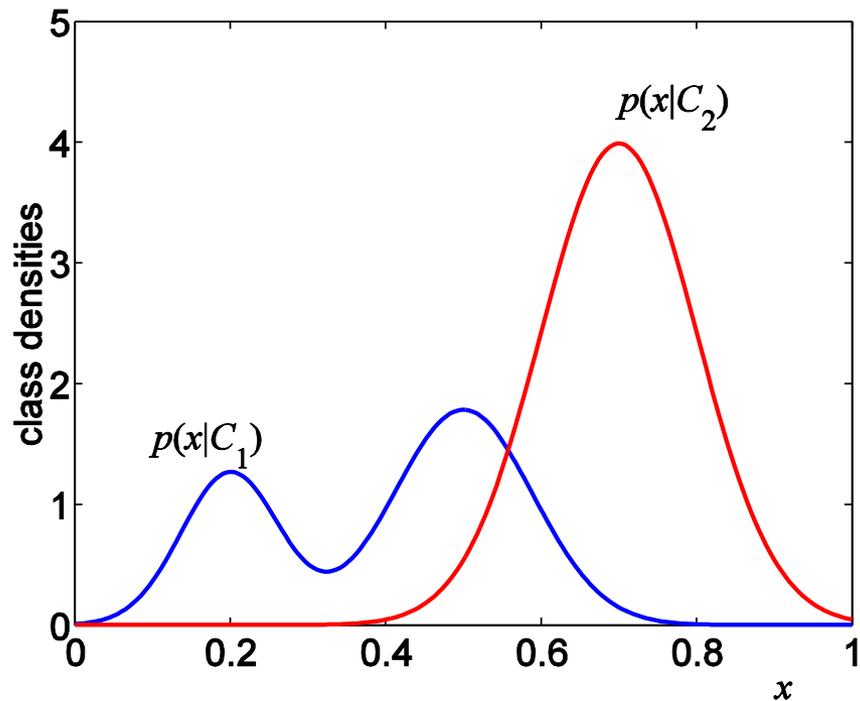
Generative methods

- model the “shape” of each class
 - histograms, PCA, mixtures of Gaussians
 - graphical models (HMM’s, belief networks, etc.)
 - ...

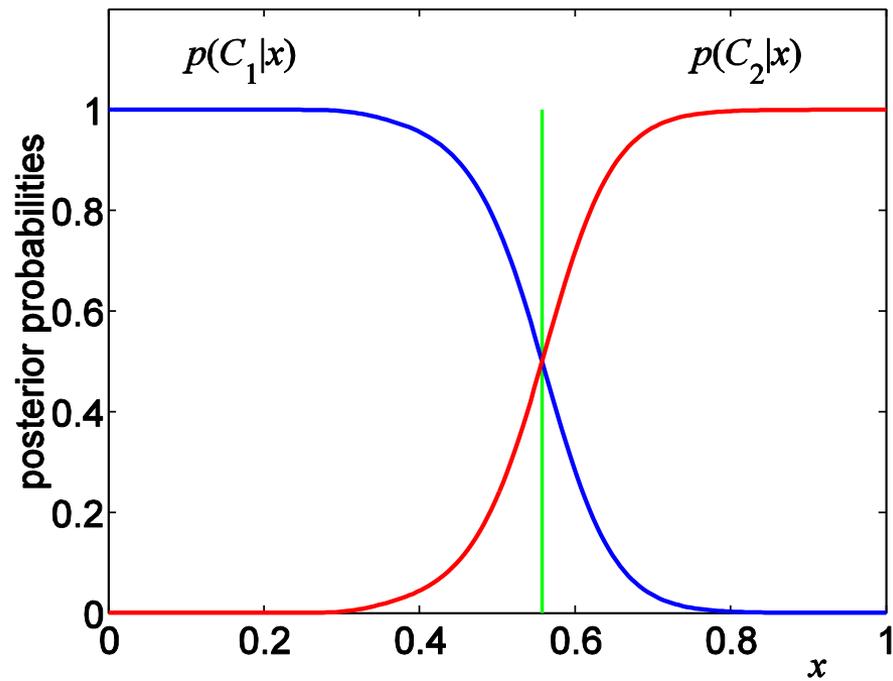
Discriminative methods

- model boundaries between classes
 - perceptrons, neural networks
 - support vector machines (SVM’s)

Generative vs. Discriminative



Generative Approach
model individual classes, priors

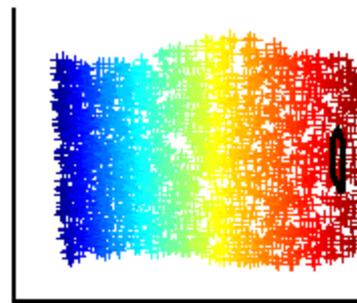
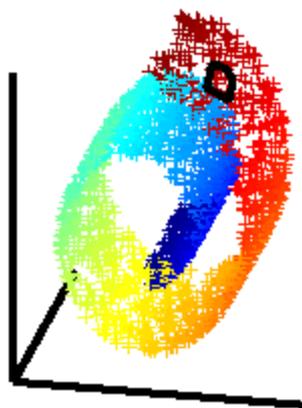
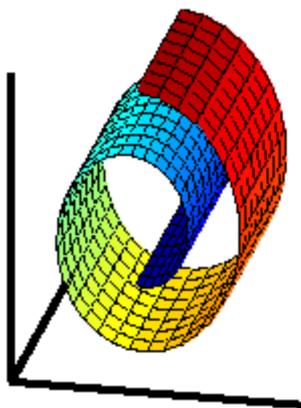


Discriminative Approach
model posterior directly

Issues: dimensionality

What if your space isn't *flat*?

- PCA may not help



Nonlinear methods

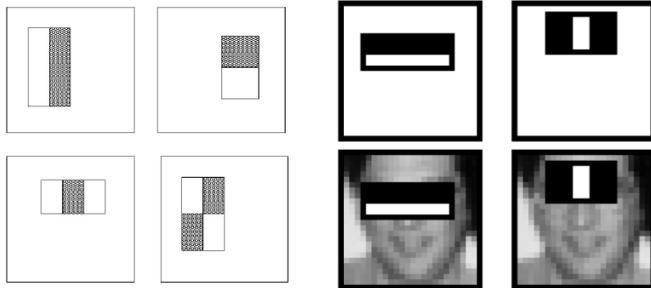
LLE, MDS, etc.

Issues: speed

- Case study: Viola Jones face detector
- Exploits two key strategies:
 - simple, super-efficient features
 - pruning (cascaded classifiers)
- Next few slides adapted Grauman & Liebe's tutorial
 - <http://www.vision.ee.ethz.ch/~bleibe/teaching/tutorial-aaai08/>
- Also see Paul Viola's talk (video)
 - <http://www.cs.washington.edu/education/courses/577/04sp/contents.html#DM>

Feature extraction

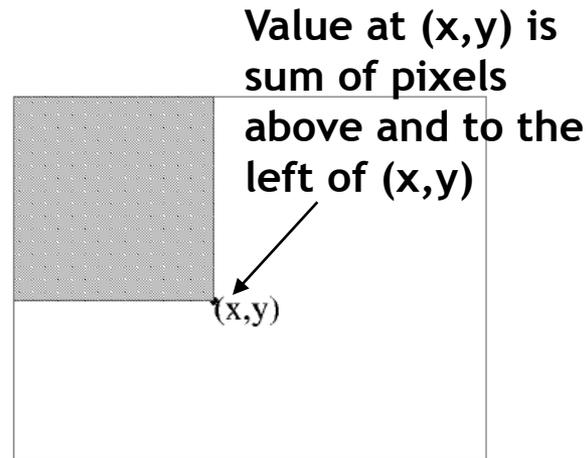
“Rectangular” filters



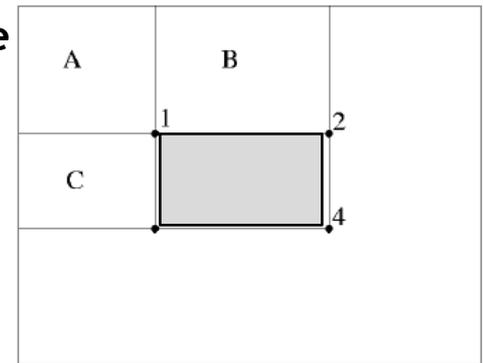
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost

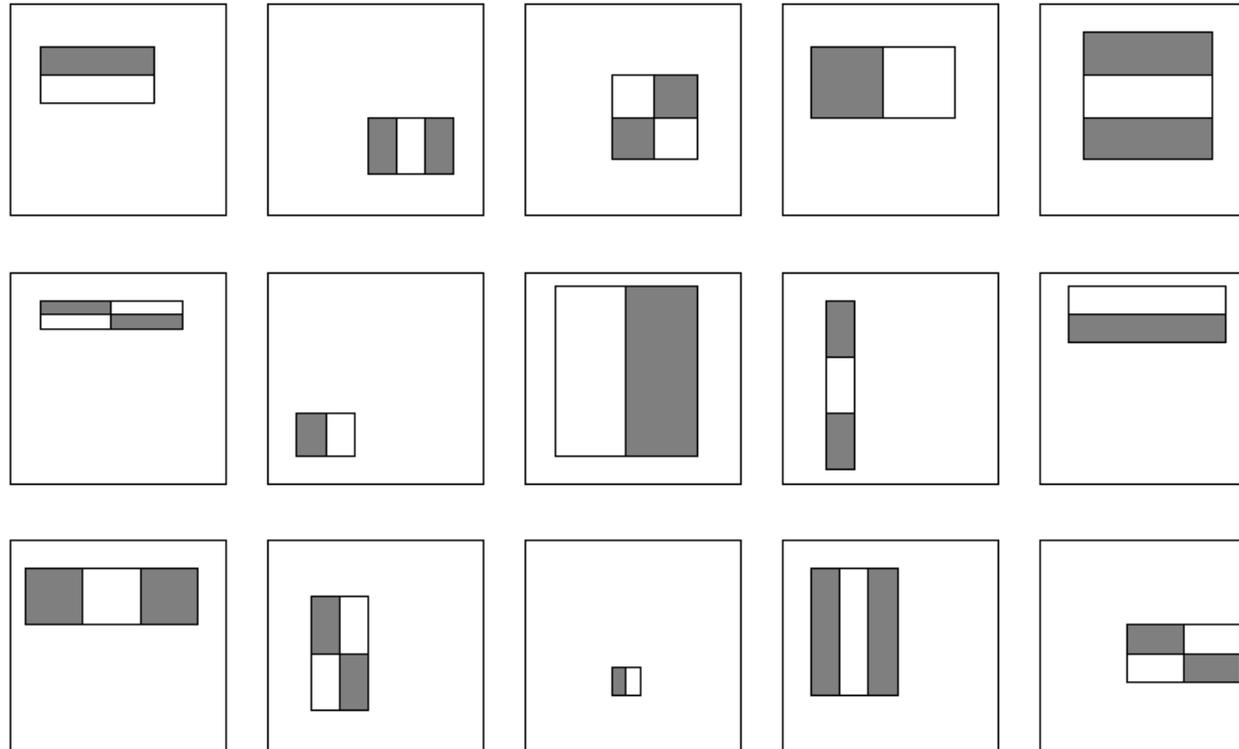


Integral image



$$\begin{aligned} D &= 1+4 - (2+3) \\ &= A + (A+B+C+D) - (A+C + A+B) \\ &= D \end{aligned}$$

Large library of filters

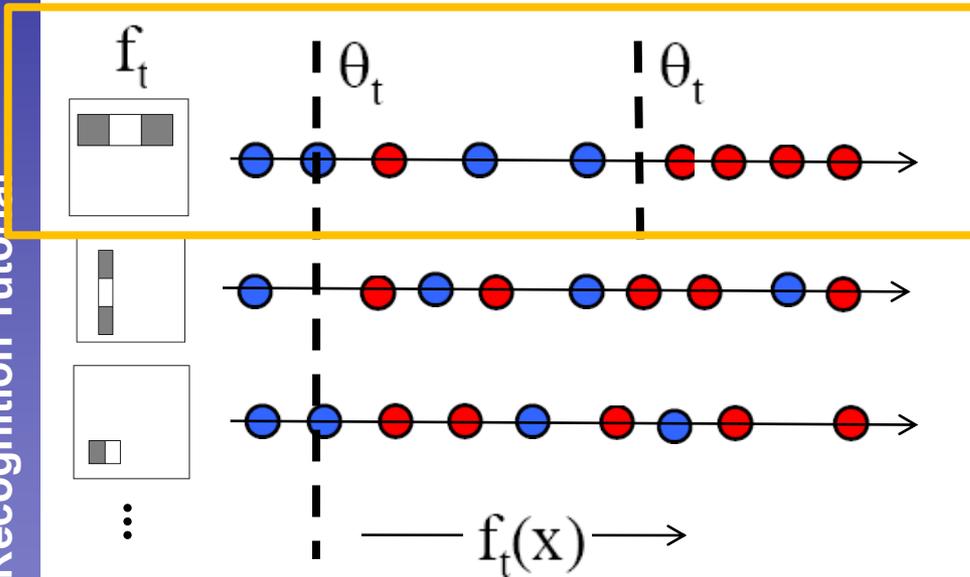


Considering all possible filter parameters:
position, scale, and type:
180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

AdaBoost for feature+classifier selection

- 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.



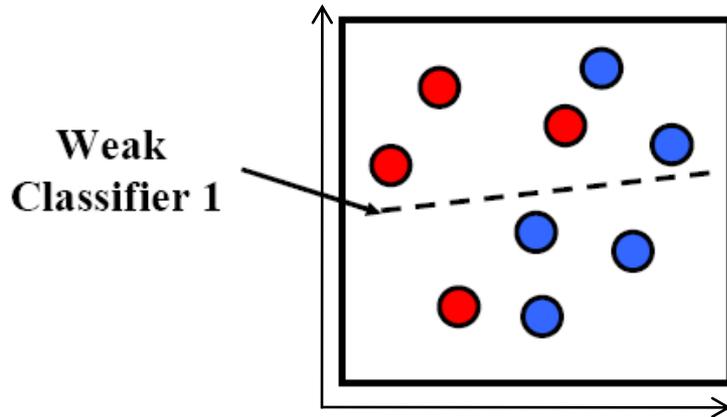
Outputs of a possible rectangle feature on faces and non-faces.

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.

AdaBoost: Intuition

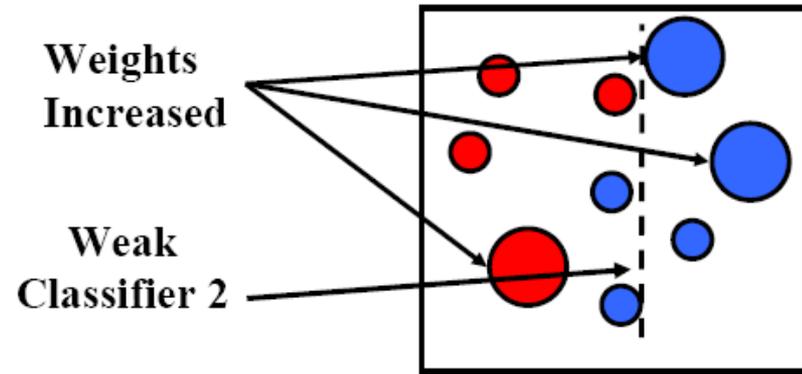
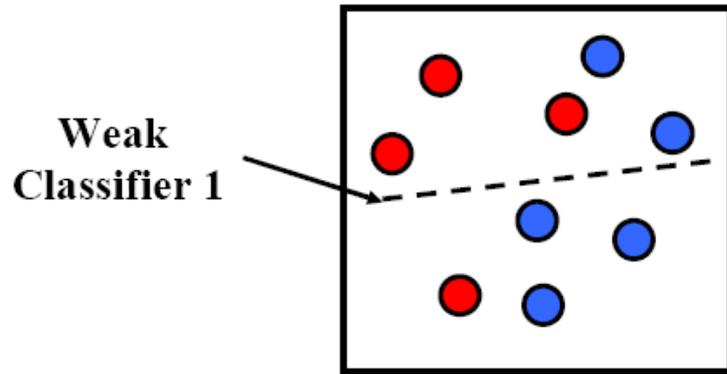


Consider a 2-d feature space with **positive** and **negative** examples.

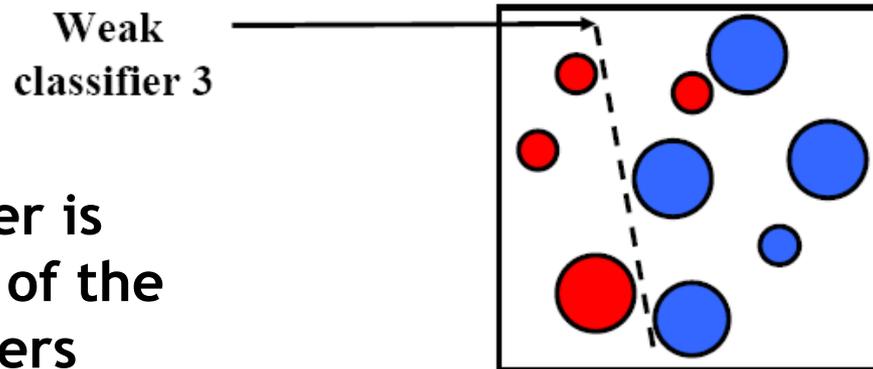
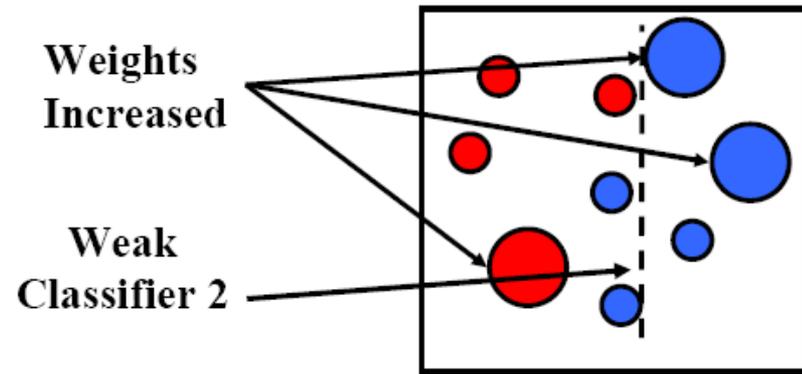
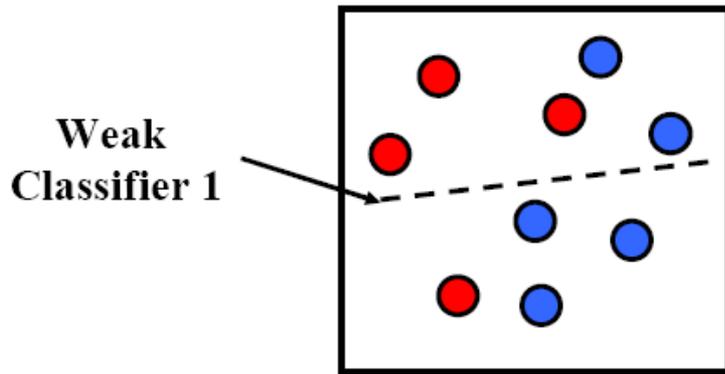
Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

AdaBoost: Intuition



AdaBoost: Intuition



Final classifier is combination of the weak classifiers

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ 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 $t = 1, \dots, T$:

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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

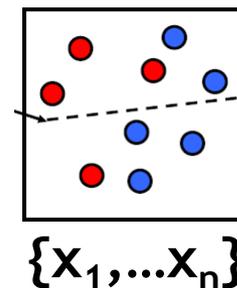
- The final strong classifier is:

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

where $\alpha_t = \log \frac{1}{\beta_t}$

AdaBoost Algorithm

Start with
uniform weights
on training
examples



For T rounds

← Evaluate
weighted error
for each feature,
pick best.

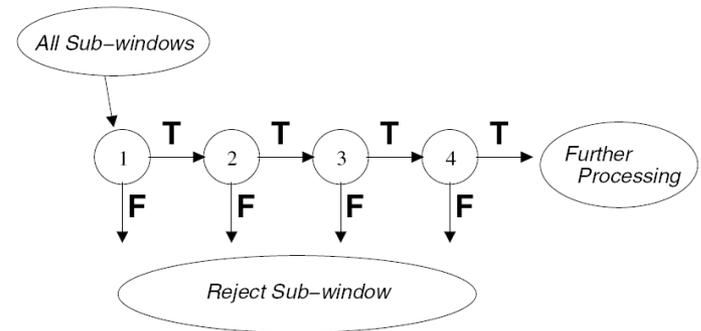
Re-weight the examples:
← Incorrectly classified -> more weight
Correctly classified -> less weight

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

Cascading classifiers for detection

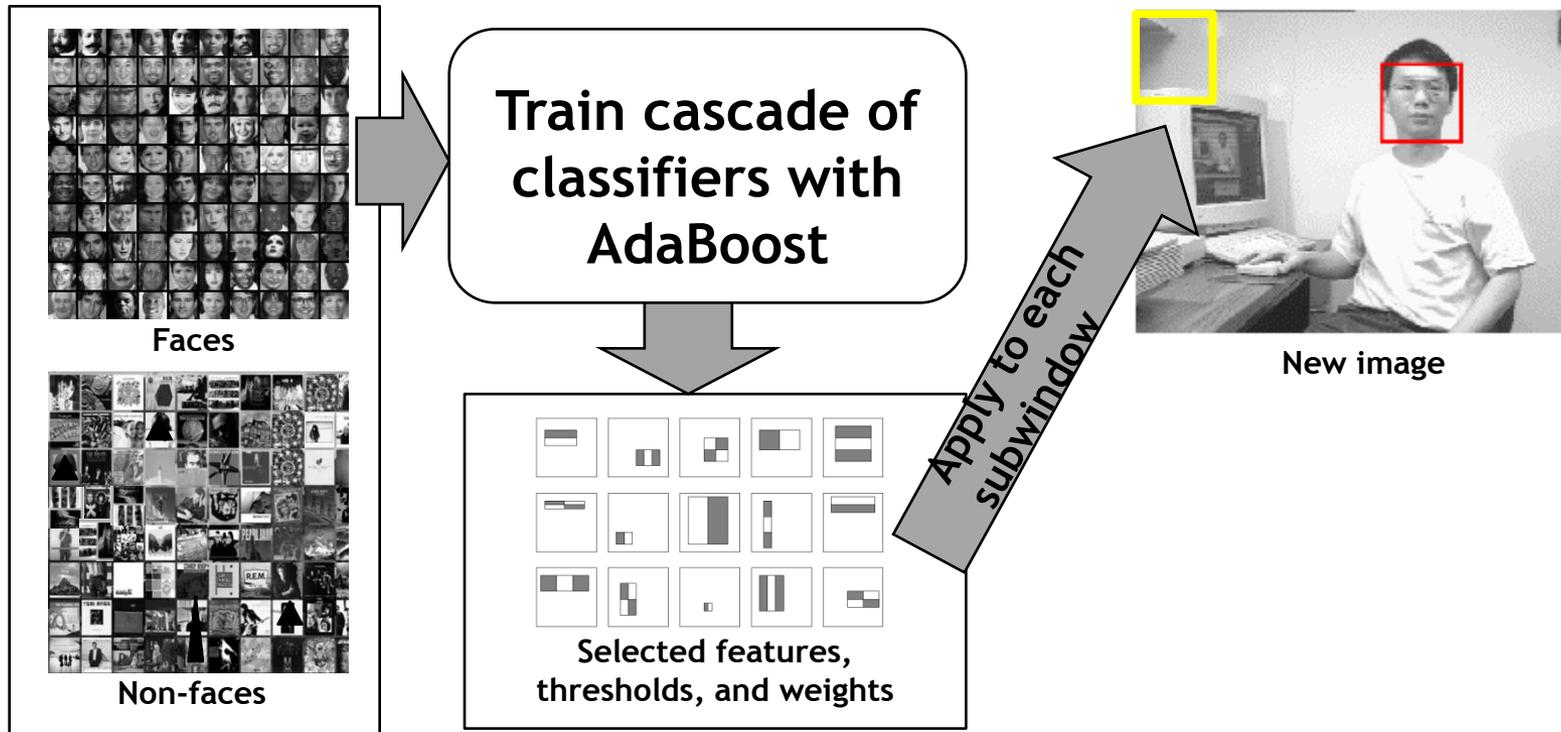
For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

- Filter for promising regions with an initial inexpensive classifier
- Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain



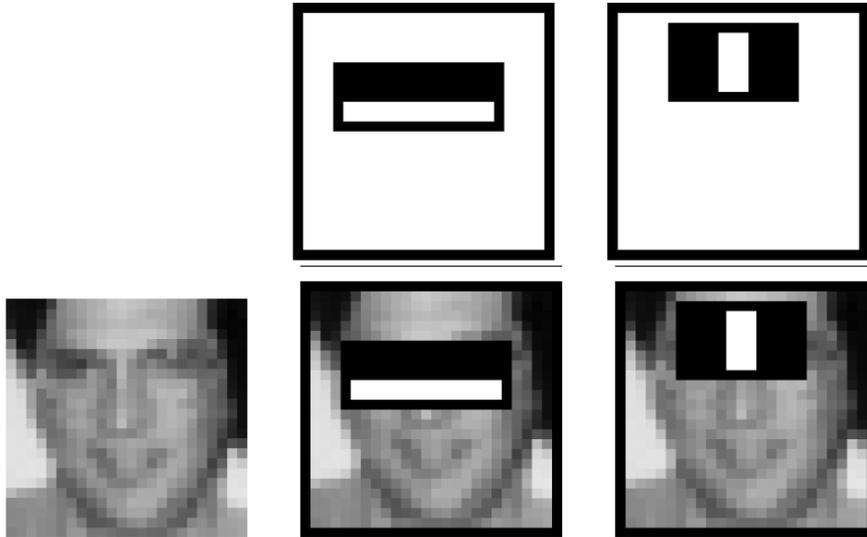
Fleuret & Geman, IJCV 2001
Rowley et al., PAMI 1998
Viola & Jones, CVPR 2001

Viola-Jones Face Detector: Summary



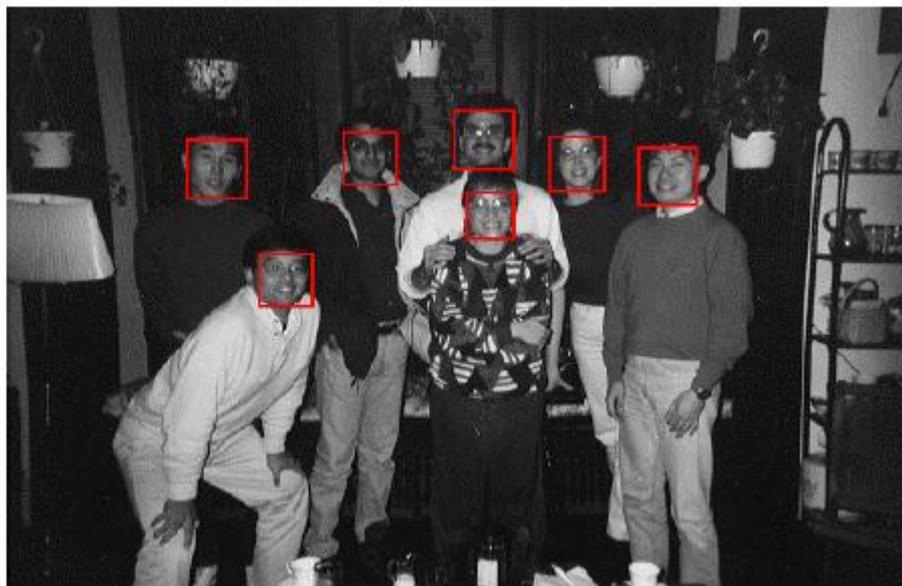
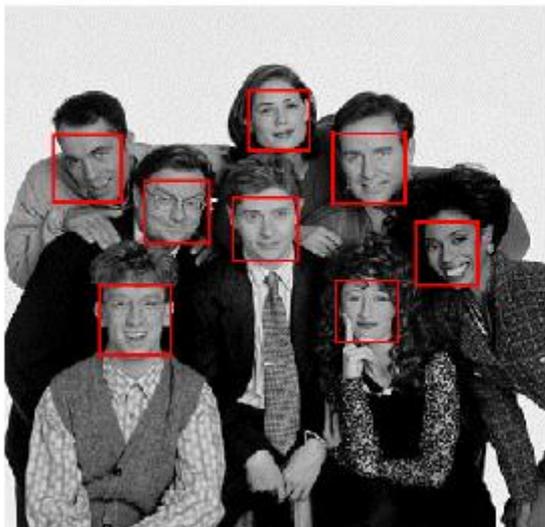
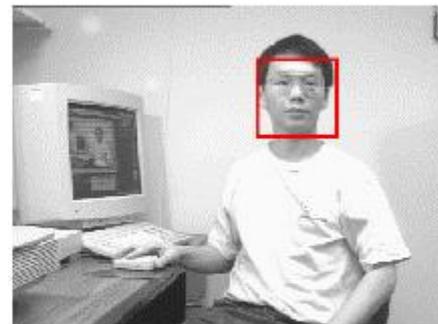
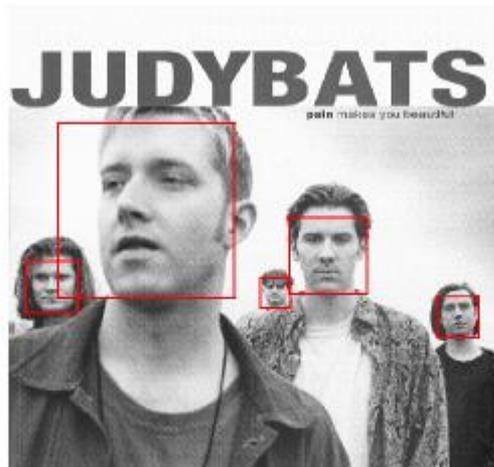
- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV:
<http://www.intel.com/technology/computing/opencv/>]

Viola-Jones Face Detector: Results

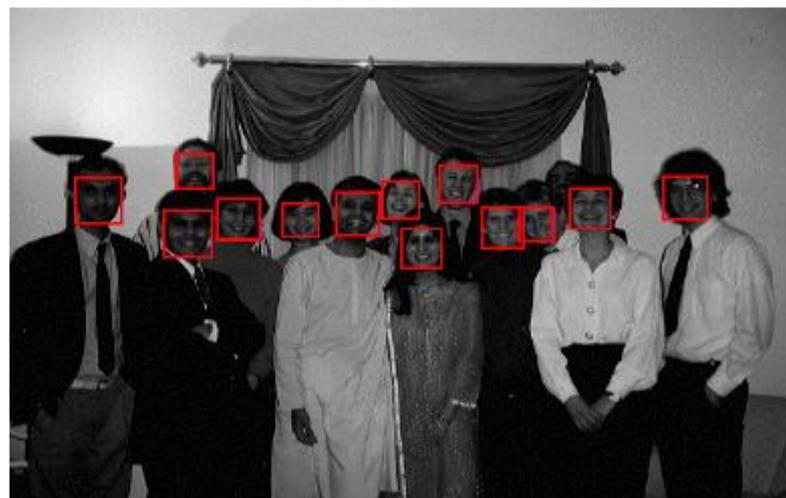
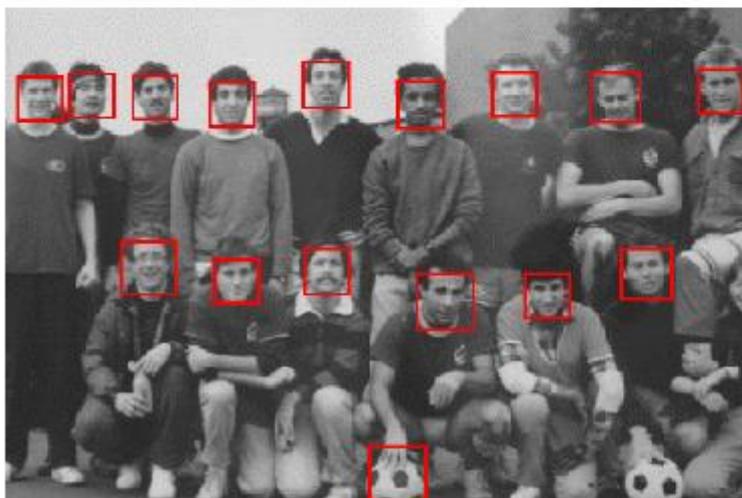
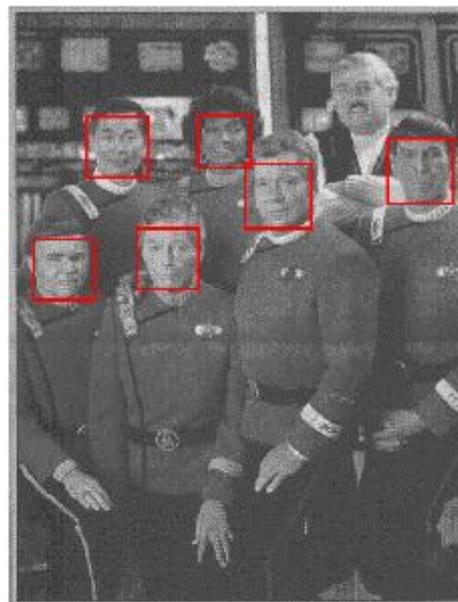
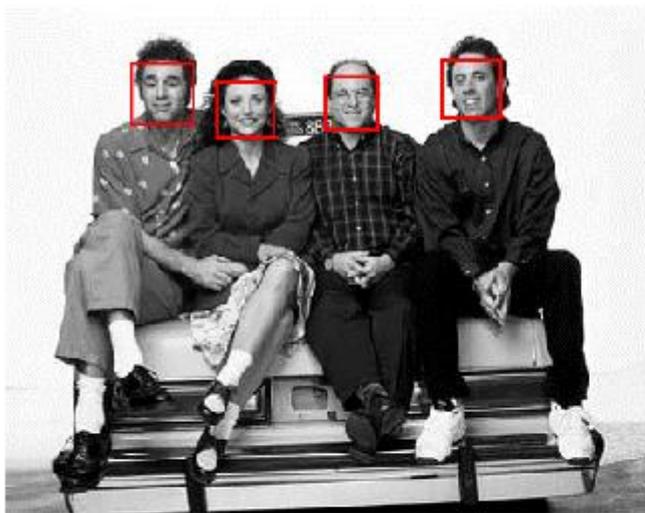


First two features
selected

Viola-Jones Face Detector: Results



Viola-Jones Face Detector: Results

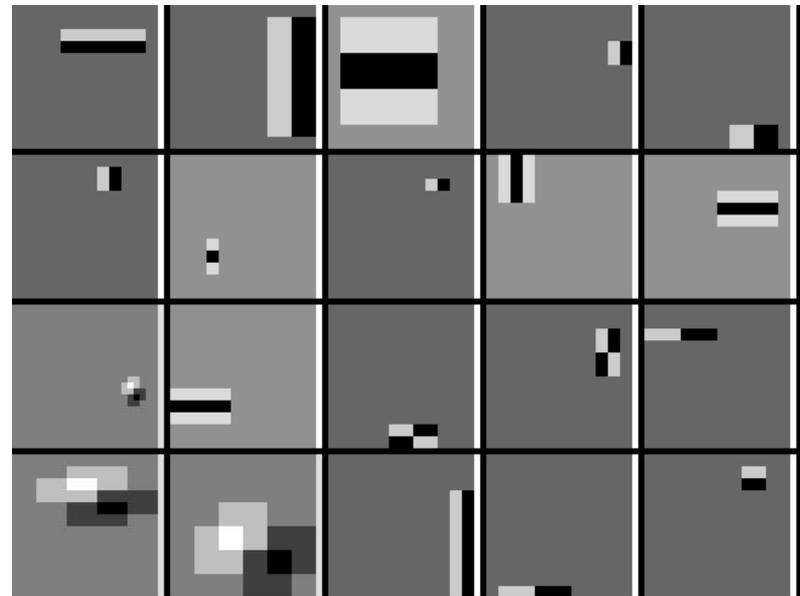


Viola-Jones Face Detector: Results



Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.



Viola-Jones Face Detector: Results

