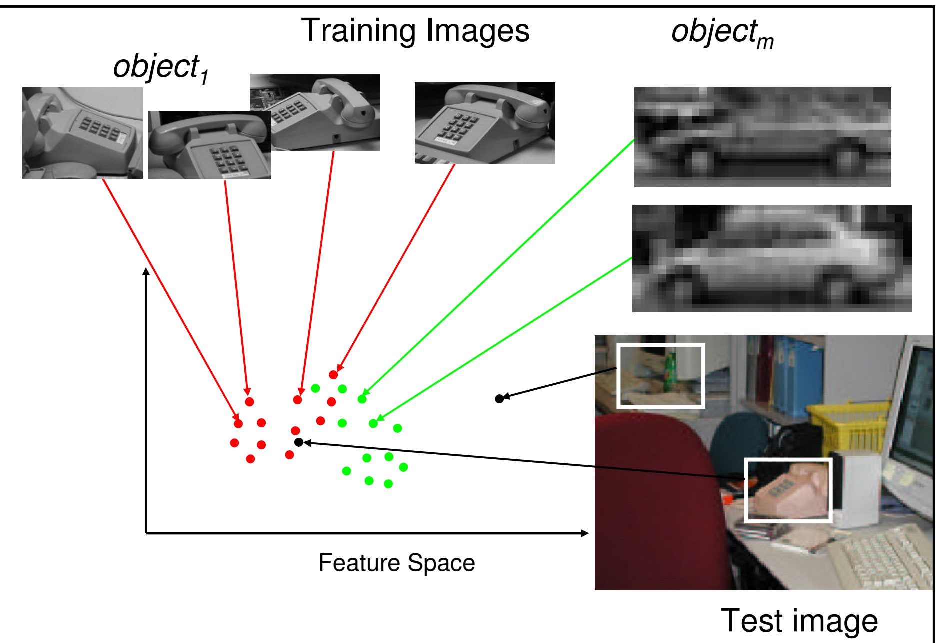
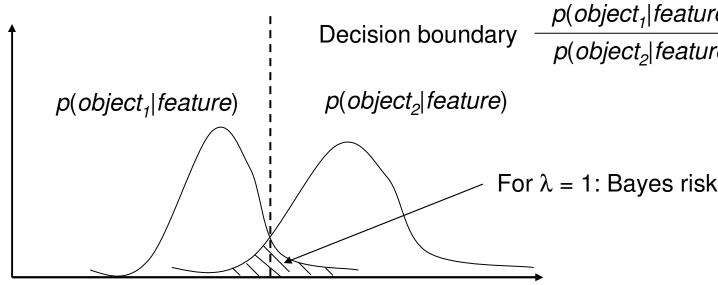
Recognition

Recognition by Templates
Classifiers



Probabilistic Formulation

Decision boundary $\dfrac{p(object_1|feature)}{p(object_2|feature)} > \lambda$



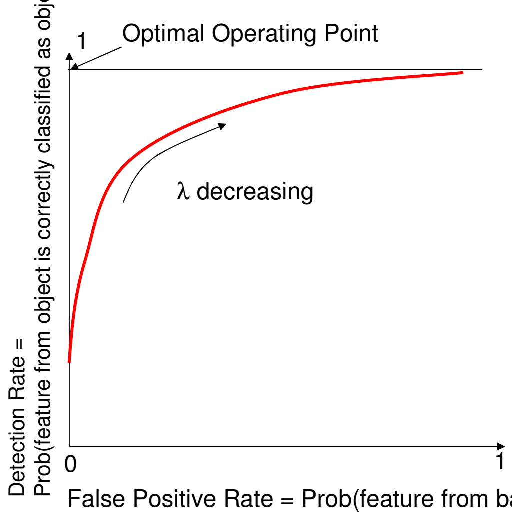
Learned from training data

$p(object_j|feature) \sim p(feature|object_j)\ p(object_j)$

How to represent and learn $p(feature|object_j)$ or decision boundary?
How to approach Bayes risk given small number of samples?
What features to use?
How to reduce the feature space?

How to evaluate classifier performance?

Receiver Operating Characteristic (ROC)



$\dfrac{p(object|feature)}{p(background|feature)} > \lambda$

Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ……………

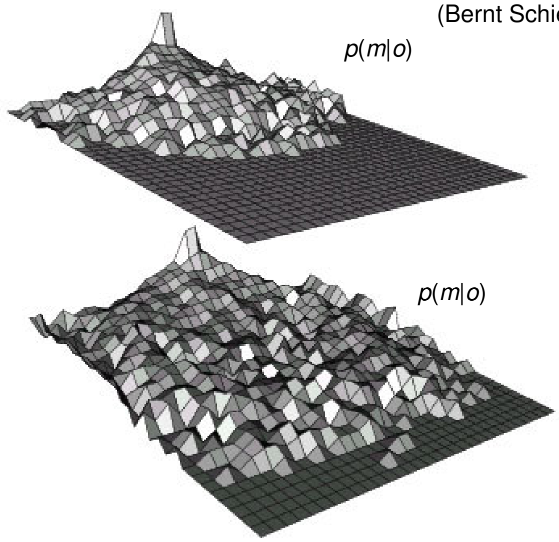
Your favorite buzzword goes here.

Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ……………

Histogram Representation

(Bernt Schiele)



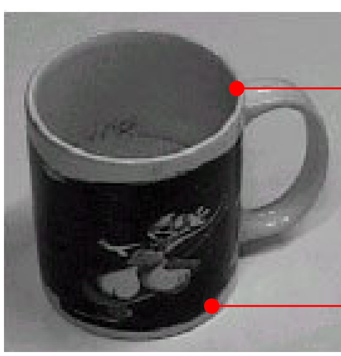
Features: m = magnitude of 1st derivatives of Gaussian + Laplacian at 3 different scales (6-component feature)

Representation: $p(m|o)$ = histogram of features from training data (24 levels per axis)

---

m_i = 6-vector at location i

Scan over all possible locations in the window



$$P(m_i | object_n)$$

Computed by looking up the histogram tables computed at training time

$$P(m_j | object_n)$$

Return object that maximizes:

$$P(object_n | image) \propto \prod_i P(m_i | object_n) P(object_n)$$

[Example from Bernt Schiele]

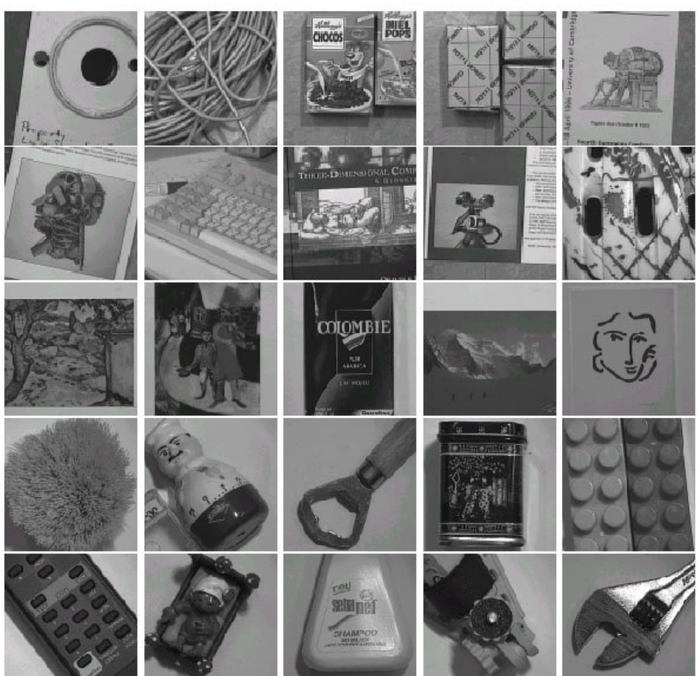
For complex scenes: Scan the image and evaluate "probability" at scanned window locations

$$P(object_n|W_k) \propto \prod_{i \in W_k} P(m_i|object_n)P(object_n)$$

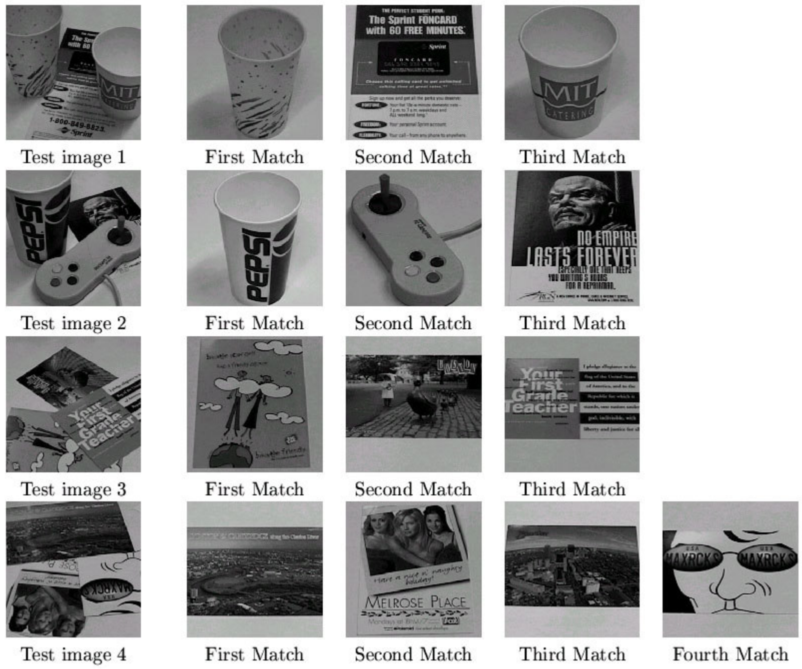
$$P(object_n|W_l) \propto \prod_{i \in W_l} P(m_i|object_n)P(object_n)$$

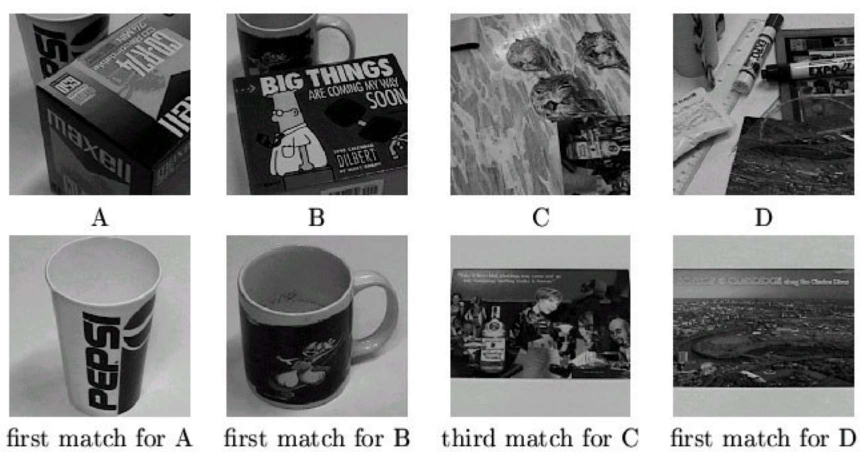
Object *n* is in the image if many windows "vote" for the image, e.g., by evaluating $vote(object_n) = \sum_k P(object_n|W_k)$

Note: This is not necessarily the best way to do this….see later



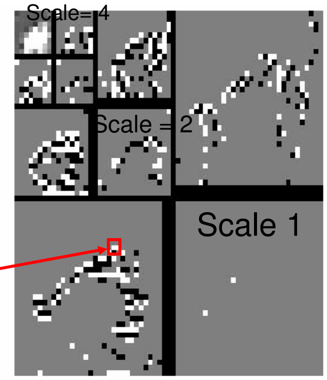
[Example from Bernt Schiele]





[Example from Bernt Schiele]

More Complicated Features



$C(x,y,s)$ = Wavelet coefficient at position x,y at scale s

- Feature = Set of coefficients $S = (C_1,..,C_N)$

Example from Henry Schneiderman

- Given features $\boldsymbol{S}_1,..,\boldsymbol{S}_r$ computed from a window, threshold the likelihood ratio

Assume independence (Naïve Bayes)

$$\log \frac{P(S_1 \cdots S_r \mid \omega_1)}{P(S_1 \cdots S_r \mid \omega_2)} =$$

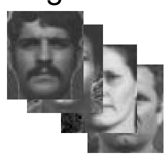
$$\log \frac{P(S_1 \mid \omega_1)}{P(S_1 \mid \omega_2)} + \log \frac{P(S_2 \mid \omega_1)}{P(S_2 \mid \omega_2)} + \ldots + \log \frac{P(S_r \mid \omega_1)}{P(S_r \mid \omega_2)} > \lambda\,?$$

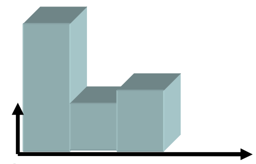
How can we compute these probabilities?

Example from Henry Schneiderman

Estimating the Probabilities

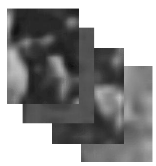
- Collect the values of the features for training data in histograms that approximate the probabilities

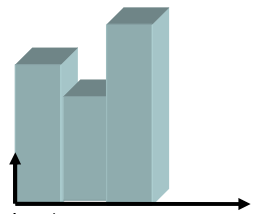




50-2,000 original images
~1,000 synthetic variations per original image

$P(S_i \mid \omega_1)$ $\quad S_i$





~10,000,000 examples

$P(S_i \mid \omega_2)$ $\quad S_i$

Example from Henry Schneiderman

Compute the values of all the features in the window
For each feature, compute the probabilities of coming from the object or non-object class
Aggregate into likelihood ratio

$f_1(0, 0) = \#5710 \rightarrow$ $P_1(\#5710, 0, 0 \mid obj) = 0.53$

$P_1(\#5710, 0, 0 \mid non\text{-}obj) = 0.56$

$f_1(0, 1) = \#3214 \rightarrow$ $P_1(\#3214, 0, 1 \mid obj) = 0.57$

$P_1(\#3214, 0, 1 \mid non\text{-}obj) = 0.48$

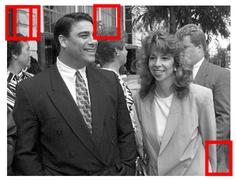
$$\frac{0.53 * 0.57 * \ldots * 0.83}{0.56 * 0.48 * \ldots * 0.19} > \lambda$$

-
-
-

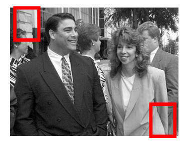
$f_N(n, m) = \#723 \rightarrow$ $P_N(\#723, n, m \mid obj) = 0.83$

$P_N(\#723, n, m \mid non\text{-}obj) = 0.19$

From Windows to Images



Search in position



Search in scale

• Move a window to all possible positions and all possible scales
• At each (position,scale) evaluate the classifier
• Return detection if above threshold

Example from Henry Schneiderman

Feature Selection Problem

- Each feature is a set of variables (wavelet coefficients) $S = \{C_1,..,C_N\}$
- Problem:
  - If N is large, the feature is very discriminative (S is equivalent to the entire window if N is the total number of variables) but representing the corresponding distribution is very expensive
  - If N is small, the feature is not discriminative but classification is very fast

Example from Henry Schneiderman

Solution: Classifier Cascade

- Standard problem:
  - We can have either discriminative or efficient features but not both!
  - Cannot do classification in one shot
- Standard solution: *Classifier Cascade*
  - Apply first a classifier with simple features → Fast and will eliminate the most obvious non-object locations
  - Then apply a classifier with more complex features → More expensive but applied *only* to these locations that survived the previous stage

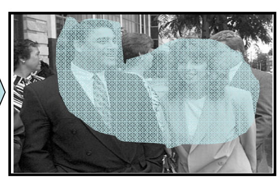
Example from Henry Schneiderman

Cascade Example

Cascade Stage 1

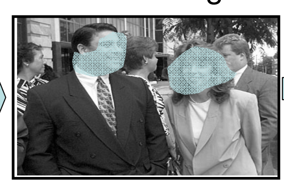
Apply classifier with very simple (and fast) features → Eliminates most of the image

Cascade Stage 2



Apply classifier with more complex features on what is left

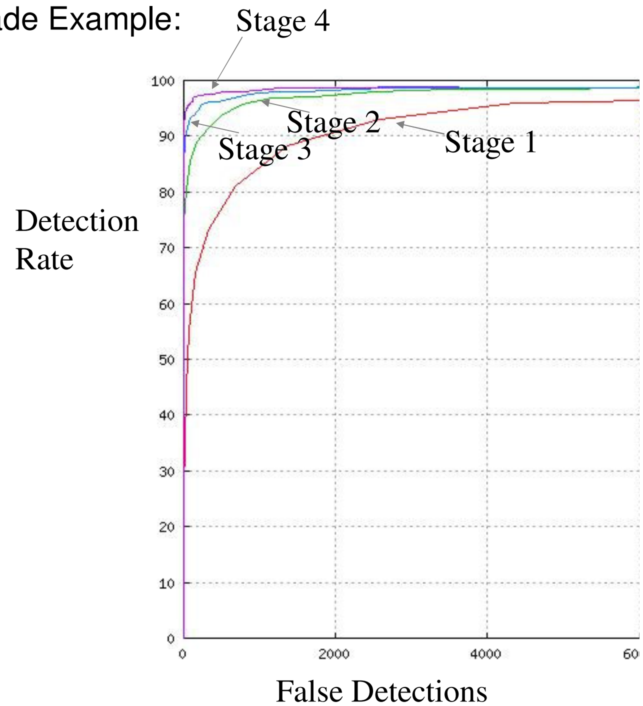
Cascade Stage 3



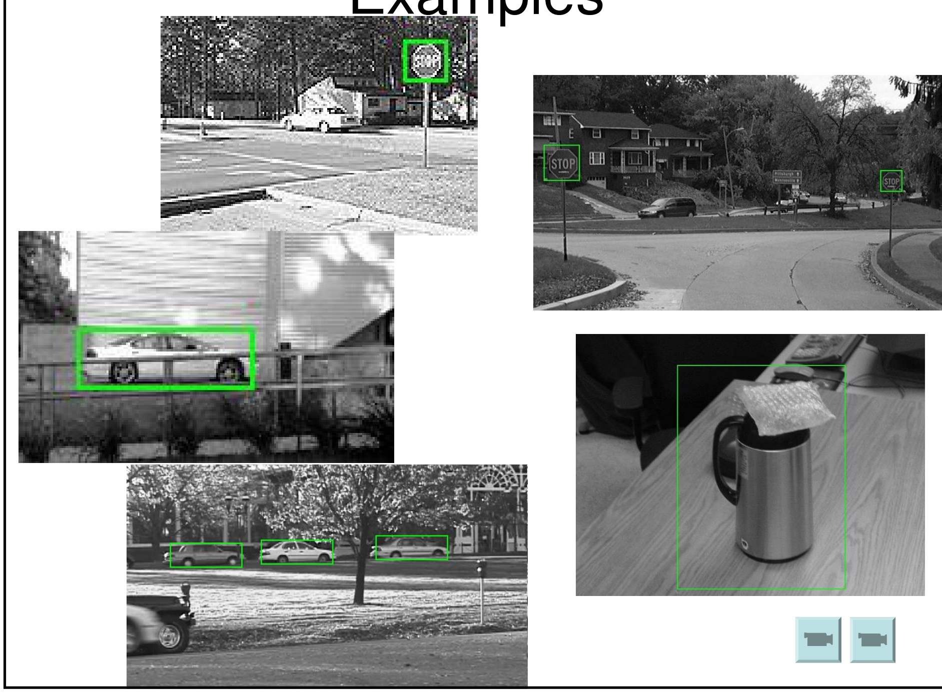
Apply classifier with more complex features on what is left

Example from Henry Schneiderman

Cascade Example:



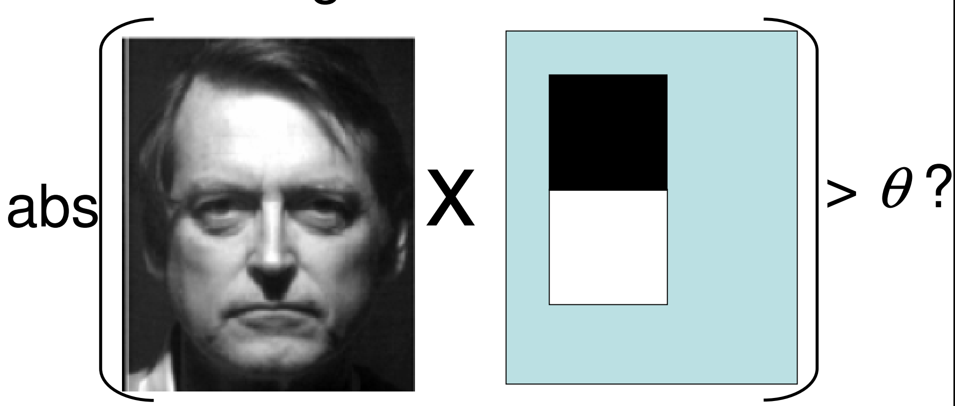
Examples



Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ....

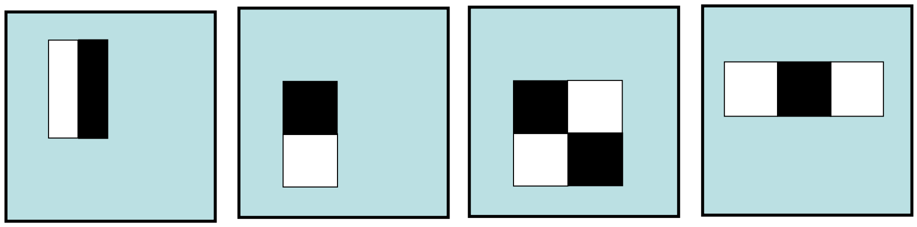
Using Weak Features



- Don’t try to design strong features from the beginning, just use really stupid but really fast features (and a lot of them)
- *Weak learner* = Very fast (but very inaccurate) classifier
- Example: Multiply input window by a very simple box operator and threshold output

(Example from Paul Viola, Distributed by Intel as part of the OpenCV library)

Feature Selection



- Operators defined over all possible shapes and positions within the window
- For a 24x24 window → 45,396 combinations!!
- How to select the "useful" features?
- How to combine them into classifiers?

(Example from Paul Viola)

- Input: Training examples $\{x_i\}$ with labels ("face" or "non-face" = +/-1) $\{y_i\}$ + weights w_i (initially w_i = 1)

Repeat T times

- Choose the feature (weak classifier h_t) with minimum error: $\boxed{\varepsilon_t = \sum_i w_i [h_t(x_i) \neq y_i]}$
- Update the weights such that
  - w_i is increased if x_i is misclassified
  - w_i is decreased if x_i is correctly classified
- Compute a weight α_t for classifier h_t
  - α_t large if ε_t is small
- Final classifier: $\boxed{H(x) = \mathrm{sgn}\left(\sum_t \alpha_t h_t(x)\right)}$

This is a general description of a *boosting* algorithm. Well-defined rules for updating w and for computing α guarantee convergence and "good" classification performance.

... $\{x_i\}$ with labels ... + weights w_i

Repeat T times

- Choose the feature (weak classifier h_t) with minimum error: ε_t ...

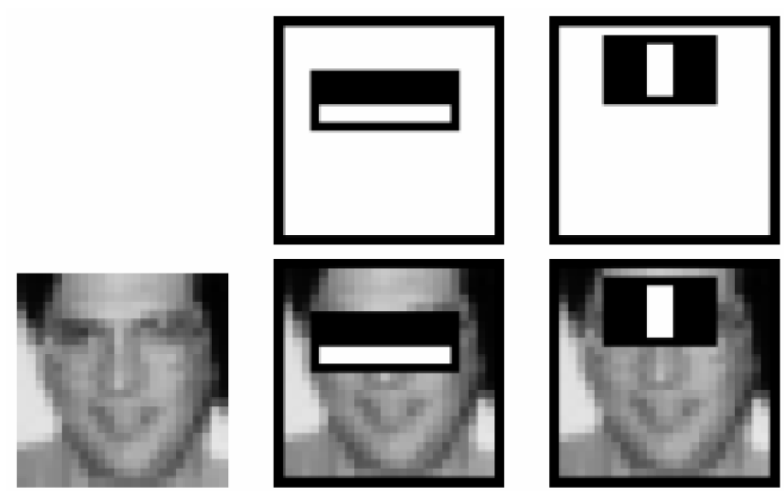
The training examples that are not correctly classified contribute more through higher weights

- Update the weights such that
  - w_i is increased if x_i is misclassified
  - w_i is decreased if x_i ...

Features that yield good classification performance receive higher weights

- Compute a weight α_t for classifier h_t
  - α_t large if ε_t is small
- Final classifier:

$$\boldsymbol{H}(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_t \alpha_t \boldsymbol{h}_t(\boldsymbol{x})\right)$$

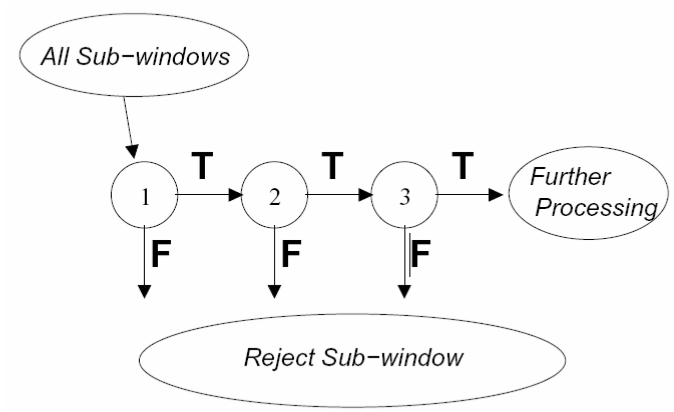


1st feature selected 2nd feature selected

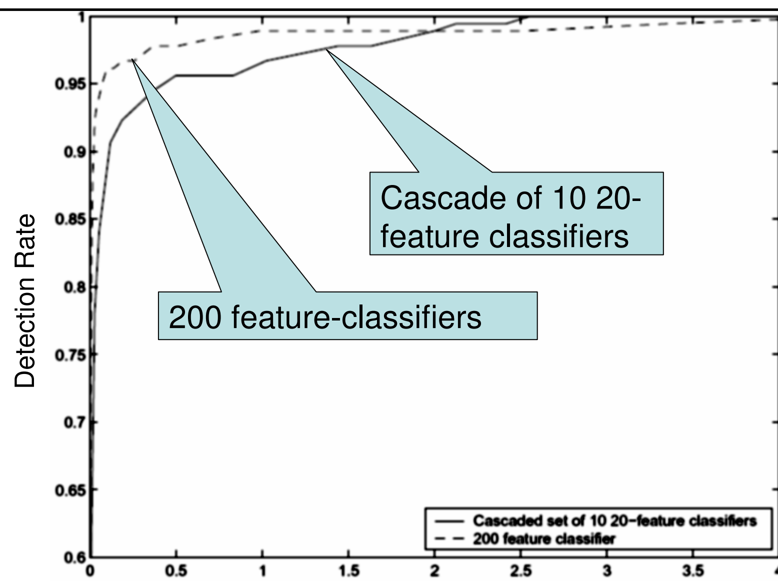
The automatic selection process selects "natural" features

(Example from Paul Viola)

Using a Cascade (Again)



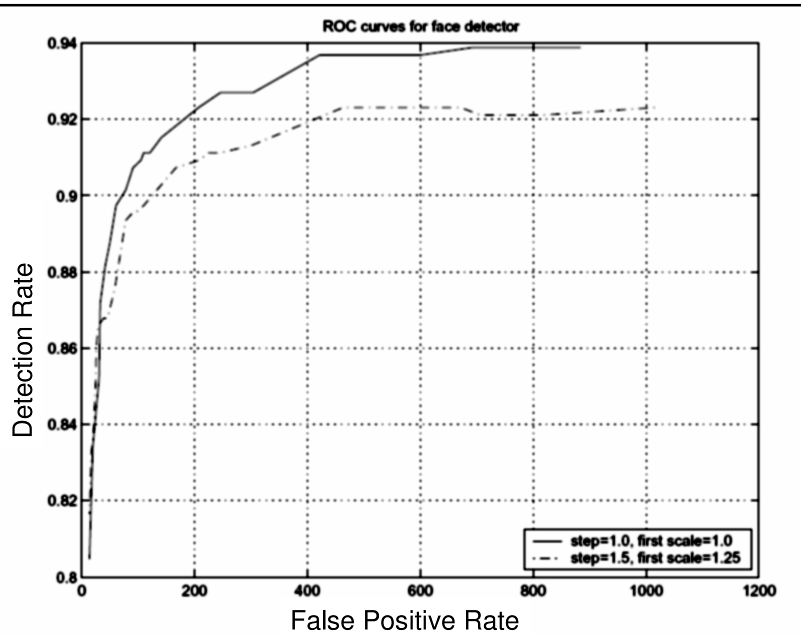
- Same problem as before: It is too hard (or impossible) to build a single accurate classifier
- Key reason: An image containing one face may have 10^5 possible locations but only 1 "correct" location → *rare event detection* → Would require an enormous number of features
- Solution: Use a cascade of classifiers. Each classifier eliminates more of the non-object locations while retaining the "object" locations.



- In this example, the cascade and the single classifier have similar accuracy, but the cascade is 10 times faster

Training: ~5000 face images
+ ~10000 non-face windows

Run-Time: Apply the cascade of classifiers over all positions and scales and return those positions/scales that survive the sequence of classifiers

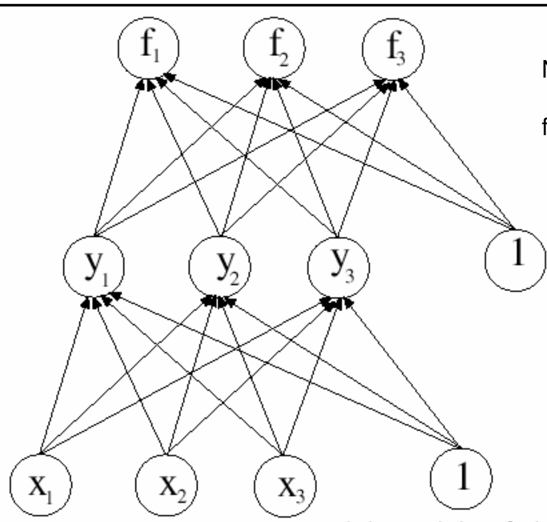


75,081,800 windows scanned over
130 images (507 faces).

(Example from Paul Viola)

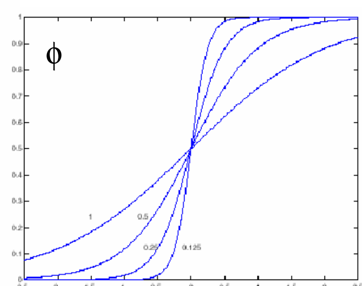
Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ……………



Neural Networks:

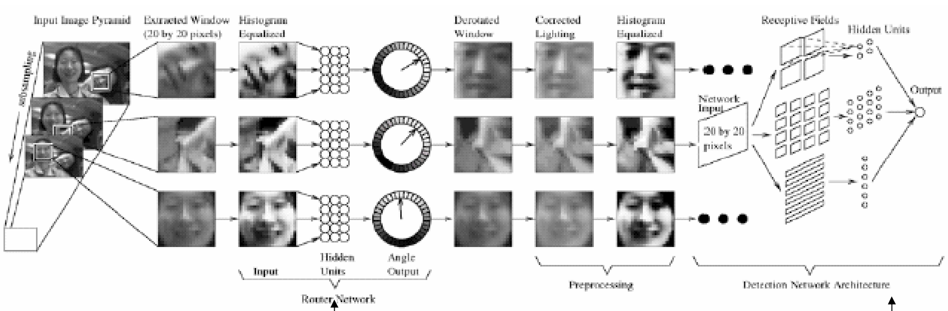
f(x) ~ g(x) (e.g., g(x) = p(o|x))



$$\boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}) = [\phi(\boldsymbol{w}_{21} \cdot \boldsymbol{y}), \phi(\boldsymbol{w}_{22} \cdot \boldsymbol{y}), \dots \phi(\boldsymbol{w}_{2n} \cdot \boldsymbol{y})]$$

$$\boldsymbol{y}(\boldsymbol{z}) = [\phi(\boldsymbol{w}_{11} \cdot \boldsymbol{z}), \phi(\boldsymbol{w}_{12} \cdot \boldsymbol{z}), \dots \phi(\boldsymbol{w}_{1m} \cdot \boldsymbol{z}), 1]$$

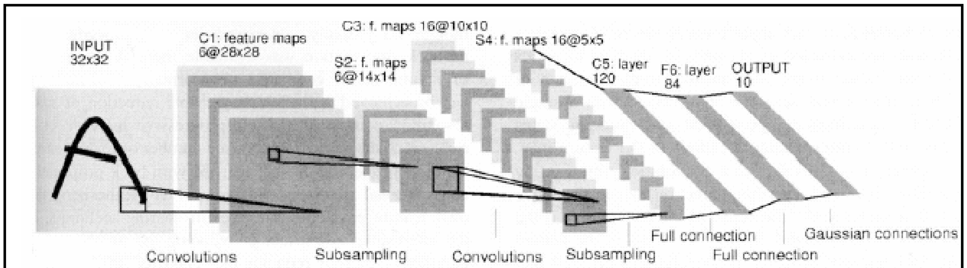
$$\boldsymbol{z}(\boldsymbol{x}) = [x_1, x_2, \dots, x_p, 1]$$

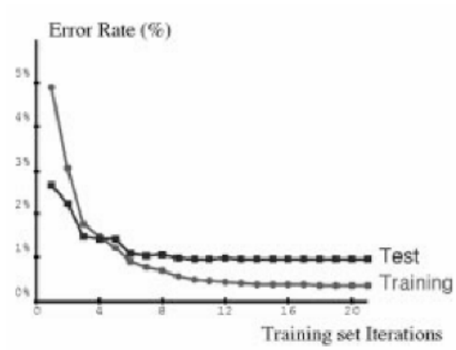


f(x) = orientation of template sampled at 10º intervals

f(x) = face detection posterior

Example from Rowley et al.





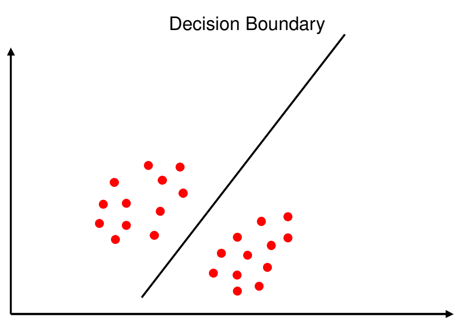
Convolutional Networks: Lecun et al.
http://yann.lecun.com/exdb/lenet/

Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ....

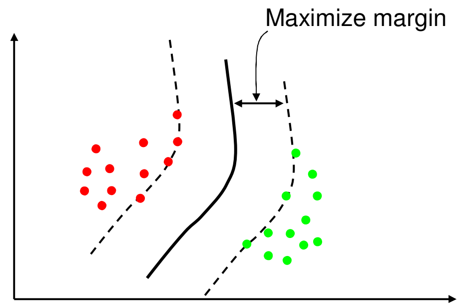
Discriminative Approaches

Linear Discriminant



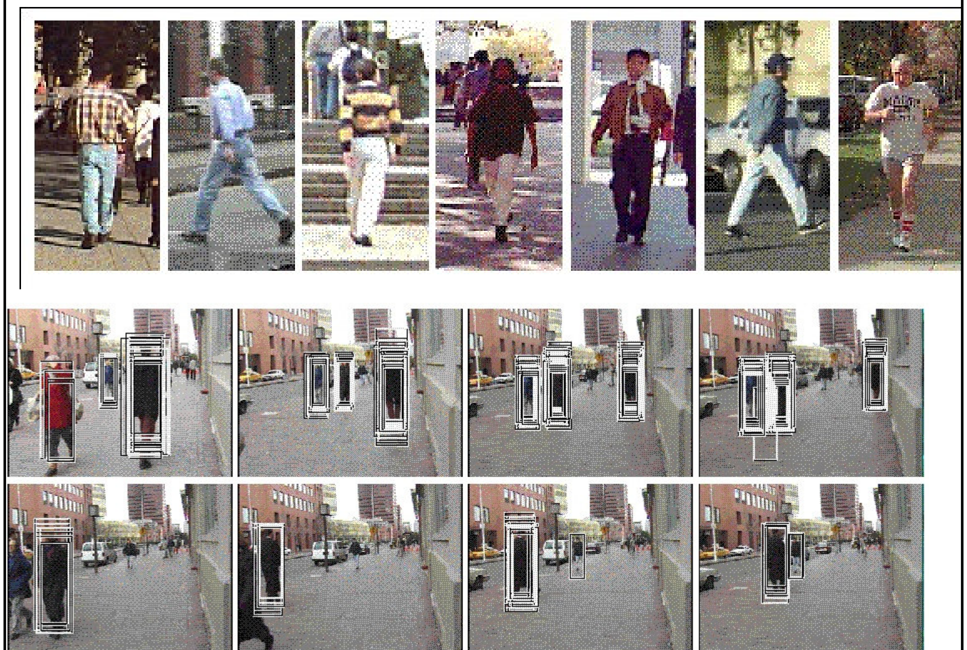
Feature Space

General: Large-Margin Classifiers, SVMs



Feature Space

- Difficult to represent the distribution in high-dimensional feature spaces → Find decision boundary directly
- General idea: Much less training data is needed to construct the decision boundary than the distributions
- Maximize separation between the classes for better generalization
- Fewer parameters: 2 Gaussians with equal covariance = 7 params, line = 2 params

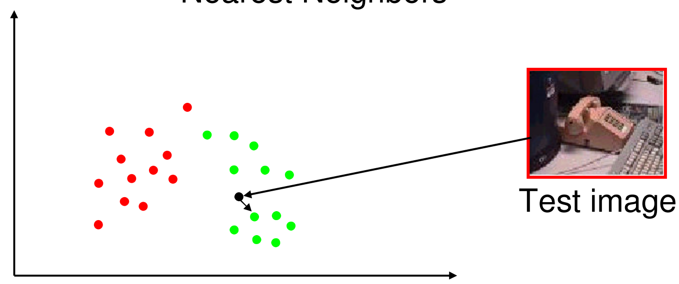


Example from Poggio et al.

Approaches

- Every single pattern classification/learning approach has been applied to this problem
- Pick your favorite:
  - Naïve Bayes
  - Boosting
  - Neural networks
  - SVMs
  - NNs
  - PCA/LDA/ICA dimensionality reduction
  - ……………

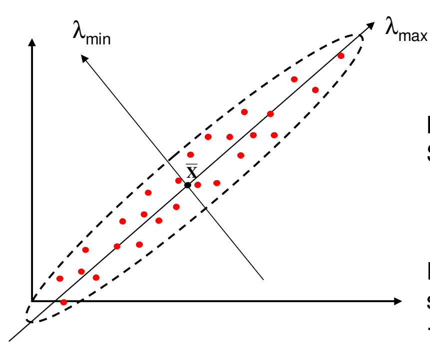
Nearest Neighbors



- Does not require recovery of distributions or decision surfaces
- Asymptotically twice Bayes risk at most
- Choice of distance metric critical
- Indexing may be difficult

Large Feature Spaces: PCA

X = feature vector of high dimension
→ Difficult indexing in high-dimensional space
→ Most of the dimensions are probably not useful



Principal Component: Dominant eigenvectors of scatter matrix

$$\tilde{\mathbf{X}} = \mathbf{X} - \overline{\mathbf{X}}$$

$$\sum_i \tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i^T = \begin{bmatrix} \sum_i x_{i1}^2 & \cdots & \sum_i x_{i1}x_{in} \\ \vdots & \ddots & \vdots \\ \sum_i x_{i1}x_{in} & \cdots & \sum_i x_{in}^2 \end{bmatrix}$$

Most of the information is contained in the Space spanned by ($\mathbf{V}_1$,..,$\mathbf{V}_k$)

$$\tilde{\mathbf{X}} \approx \lambda_1 \mathbf{V}_1 \bullet \tilde{\mathbf{X}} + \cdots + \lambda_k \mathbf{V}_k \bullet \tilde{\mathbf{X}}$$

PCA: Project first in the lower-dimensional space spanned by the principal component
→ Indexing in much lower dimensional space
→ Feature selection

PCA for Recognition

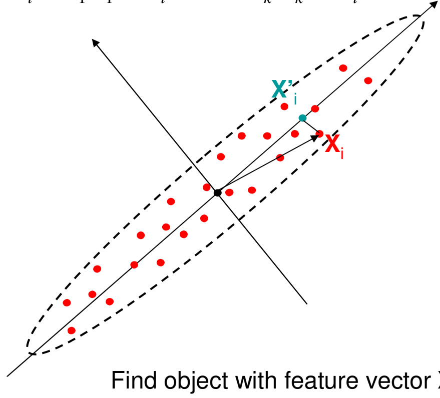
(assume centered features)

Training features: $\mathbf{X}_1,..,\mathbf{X}_m$

Compute principal directions: $\mathbf{V}_1,..,\mathbf{V}_k$

Project training features onto principal directions

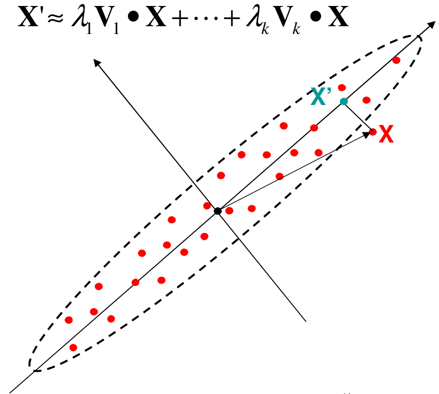
$$\mathbf{X}'_i \approx \lambda_1 \mathbf{V}_1 \bullet \mathbf{X}_i + \cdots + \lambda_k \mathbf{V}_k \bullet \mathbf{X}_i$$



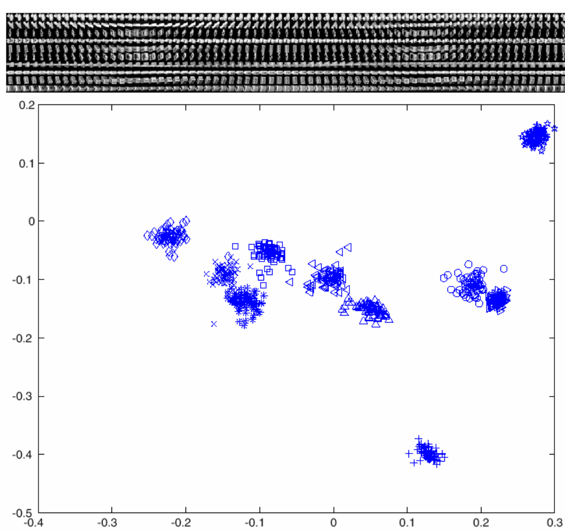
Input: Feature vector **X**

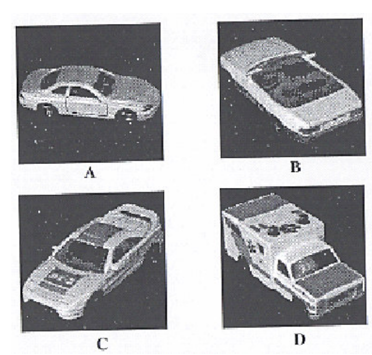
Project **X** onto principal component space:

$$\mathbf{X}' \approx \lambda_1 \mathbf{V}_1 \bullet \mathbf{X} + \cdots + \lambda_k \mathbf{V}_k \bullet \mathbf{X}$$



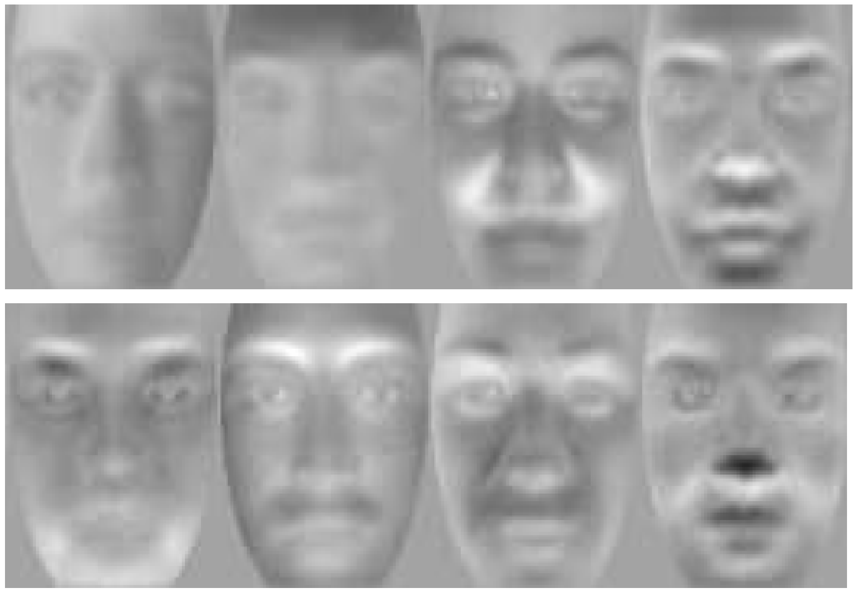
Find object with feature vector $\mathbf{X}'_{io}$ closest to **X**: $i_o = \arg\max_i \left\| \mathbf{X}' - \mathbf{X}'_i \right\|$





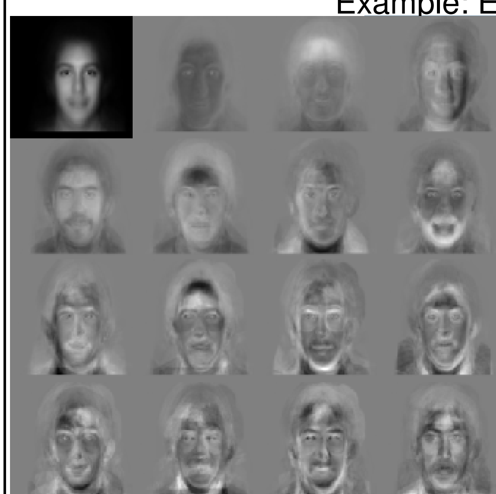
From S. Nayar, Columbia University

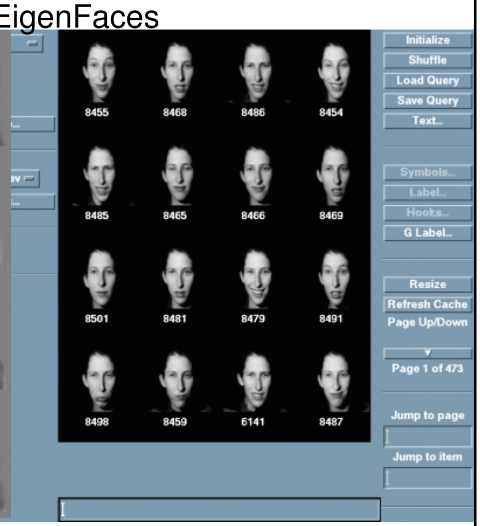
Example Eigenvectors:



[Example from Draper et al.]

Extreme case: X = image itself
Example: EigenFaces



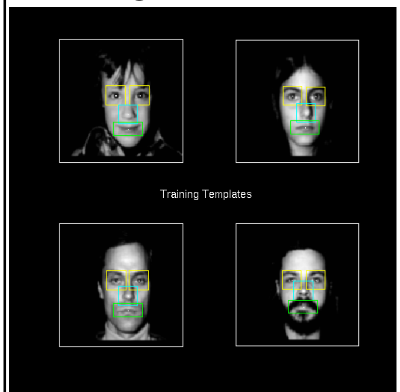


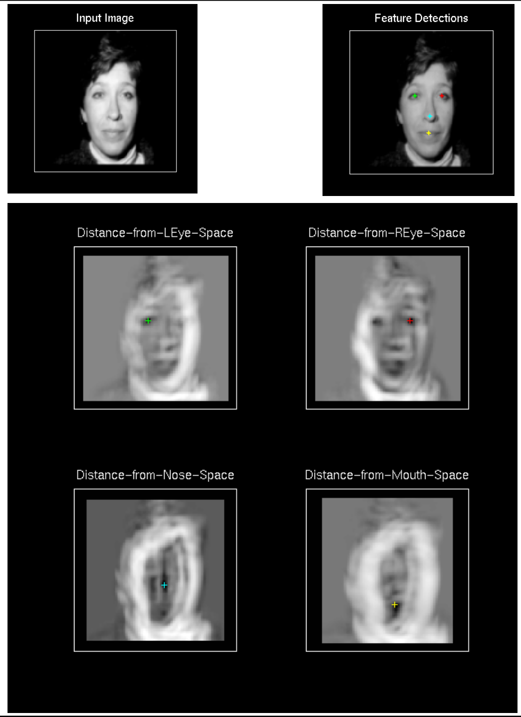
Eigenfaces: Projection on the first 20 eigenvectors from 128 face images

15 most similar faces among 7,562 faces (3000 subjects)

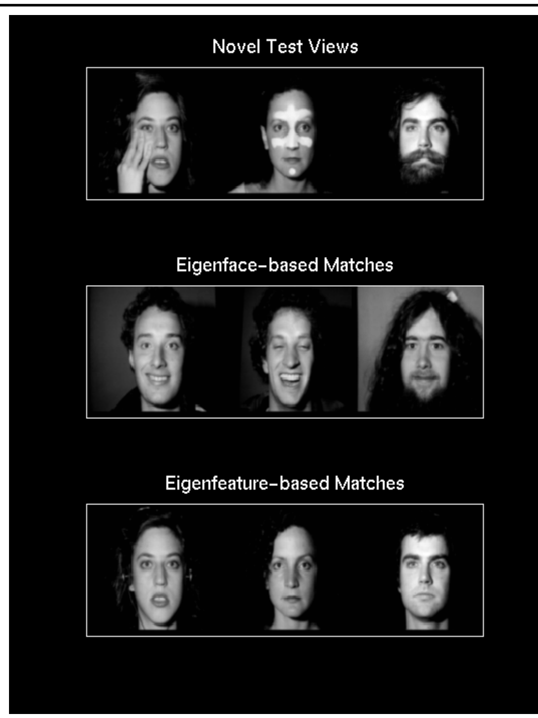
http://www-white.media.mit.edu/vismod/demos/facerec/basic.html

EigenFeatures

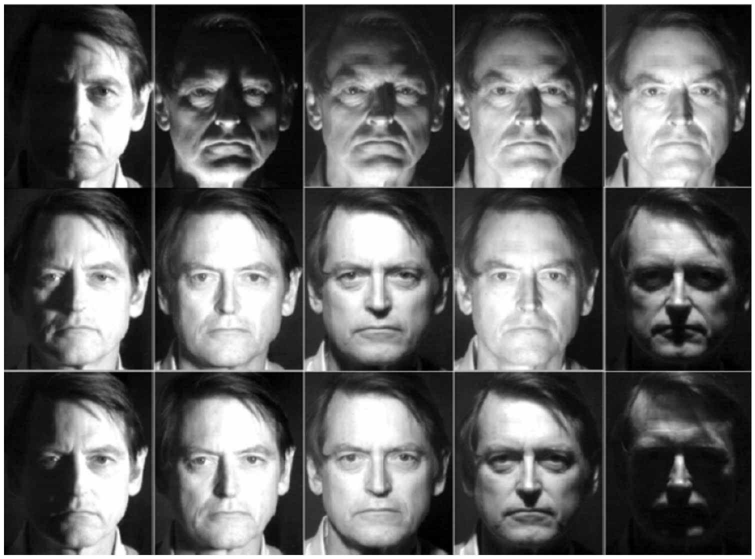




Features trained on 128 face images, retaining the first 10 eigenvectors



Problems

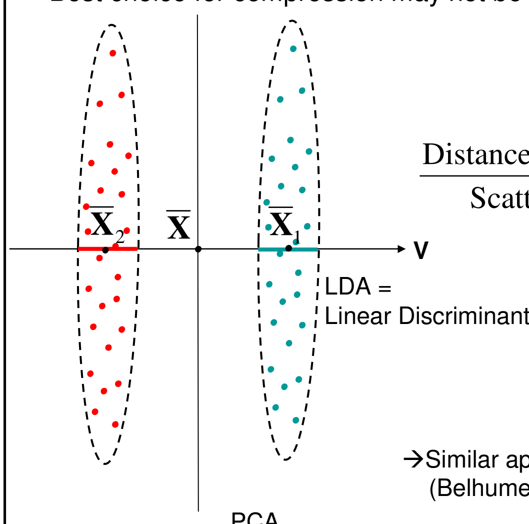


Variation in appearance due to illumination and expression >>> Variation in appearance due to identity

Example from the Yale Face DataBase

Problems

- Assume "linear" distribution of features
- Best choice for compression may not be the best choice for discrimination



LDA → Find projection direction **V** that separates the 2 classes best

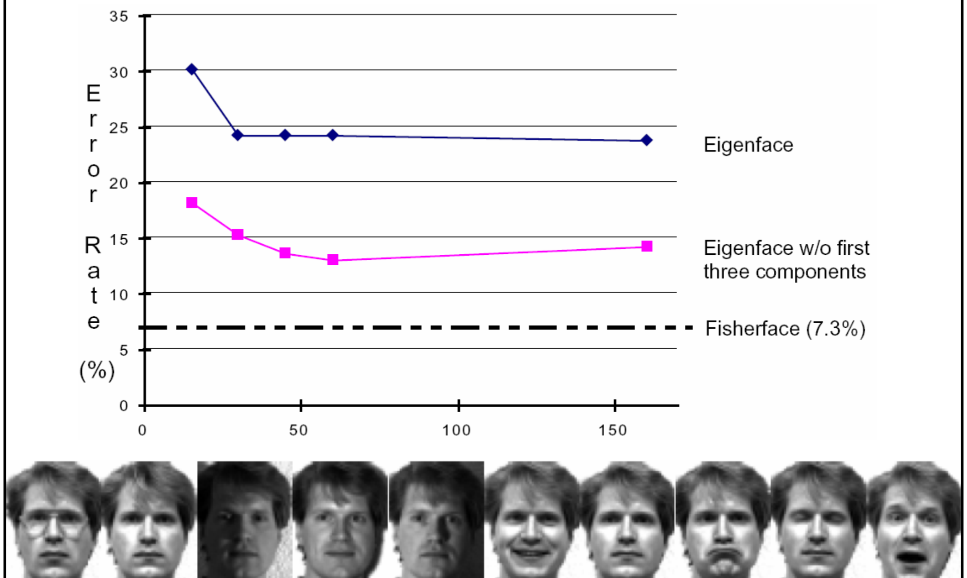
Maximize:

$$\frac{\text{Distance between classes after projection}}{\text{Scatter of classes after projection}} = \frac{(\mathbf{V} \bullet (\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2))^2}{\mathbf{V}(\mathbf{C}_1 + \mathbf{C}_2)\mathbf{V}^T}$$

→ Generalized eigenvalue problem

→Similar application of LDA to faces: FisherFaces (Belhumeur, Yale/Columbia)

Example



Example from Belhumeur et al.