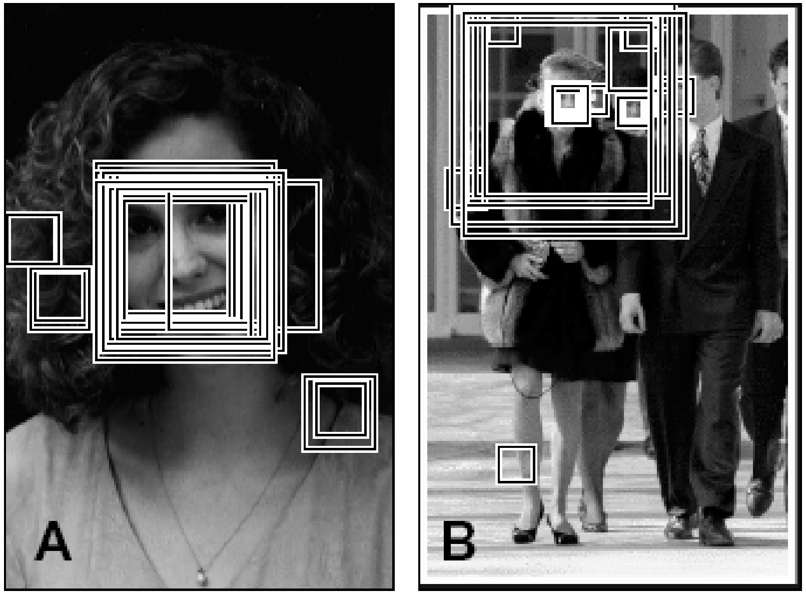
Classifiers for Template Recognition

Reading: Chapter 22 (skip 22.3)

Face Recognition

- Examine each window of an image
- Classify object class within each window based on a *training set* images



Slide credits for this class:
David Lowe, Frank Dellaert, Forsyth & Ponce, Paul Viola, Christopher Rasmussen
A. Roth for face recognition

Classification

- Idea: we are taught to recognize objects, motions, textures … etc. by being presented examples
- How do we use this idea to construct machine based classifiers
- Previous classes we saw some approaches that did template matching
- Today extend this idea further and discuss classification of objects using features
- Very important area of computational science/statistics
  - Techniques are used in diverse areas such as vision, audition, credit scores, automatic diagnosis, DNA matching ….

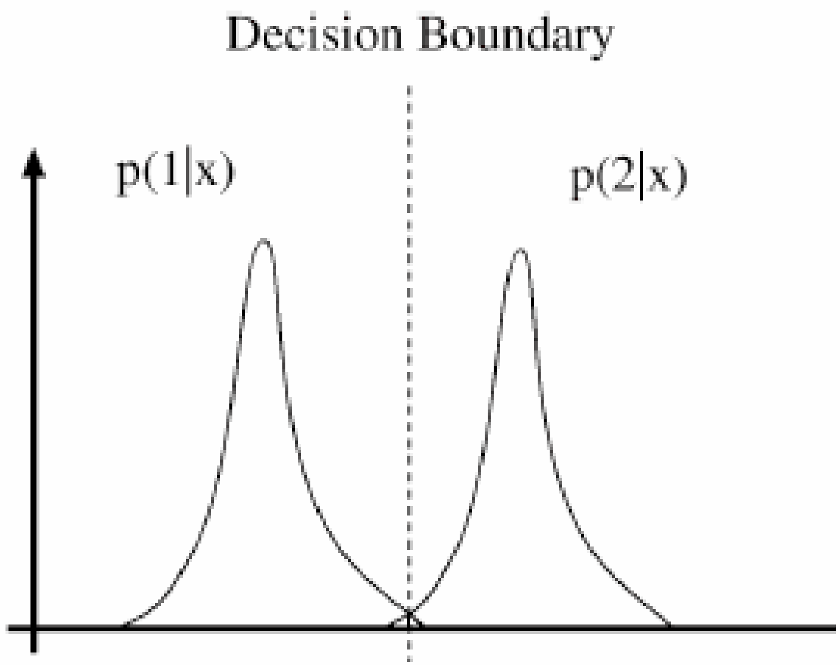
Example: A Classification Problem

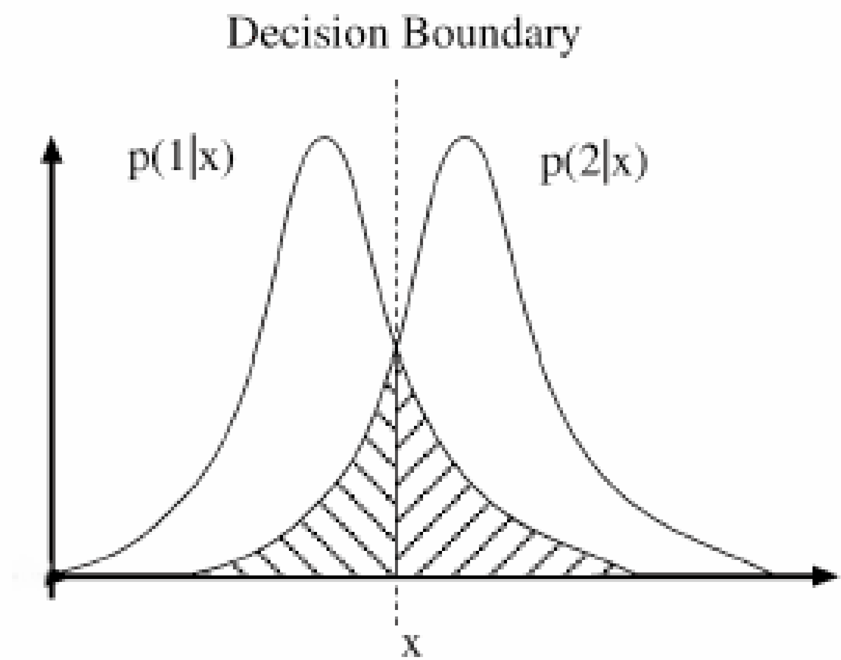
- Categorize images of fish—say, “Atlantic salmon” vs. “Pacific salmon”
- Use features such as length, width, lightness, fin shape & number, mouth position, etc.
- Steps
  1. Preprocessing (e.g., background subtraction)
  2. Feature extraction
  3. Classification

example from Duda & Hart

Bayes Risk

Some errors may be inevitable: the minimum risk (shaded area) is called the Bayes risk

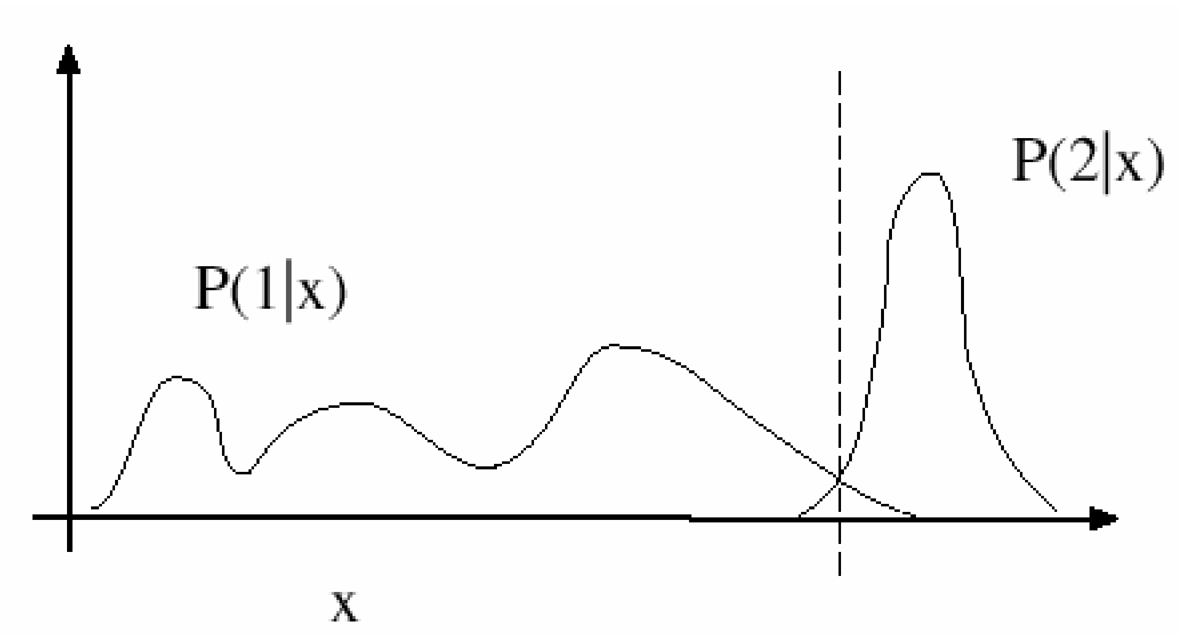




Probability density functions (area under each curve sums to 1)

Discriminative vs Generative Models

Finding a decision boundary is not the same as modeling a conditional density.



Loss functions in classifiers

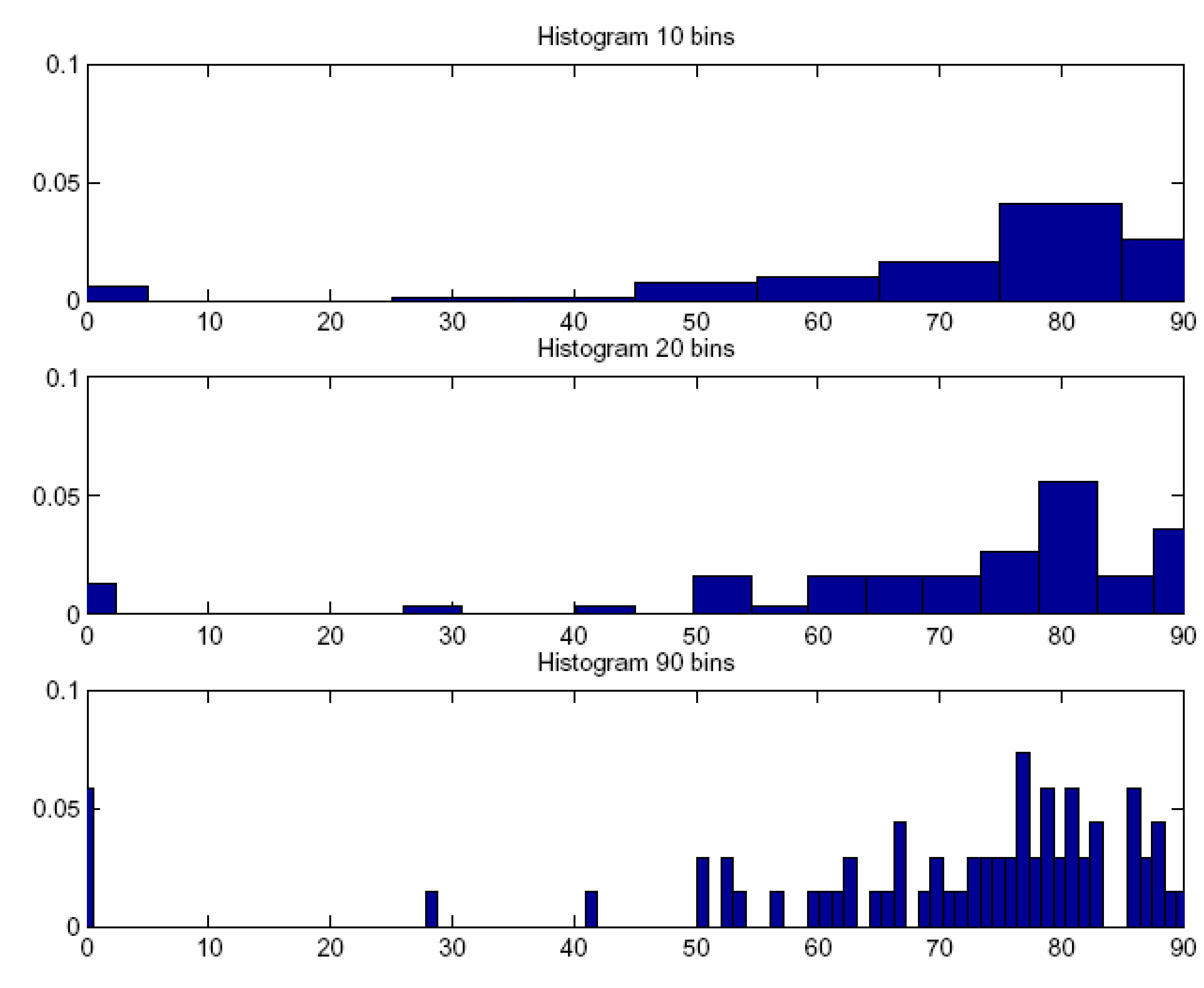
- Loss
  - some errors may be more expensive than others
    - e.g. a fatal disease that is easily cured by a cheap medicine with no side-effects -> false positives in diagnosis are better than false negatives
  - We discuss two class classification: L(1->2) is the loss caused by calling 1 a 2
- Total risk of using classifier s

$$R(s) = Pr\left\{1 \to 2 | \text{using } s\right\} L(1 \to 2) + Pr\left\{2 \to 1 | \text{using } s\right\} L(2 \to 1)$$

Histogram based classifiers

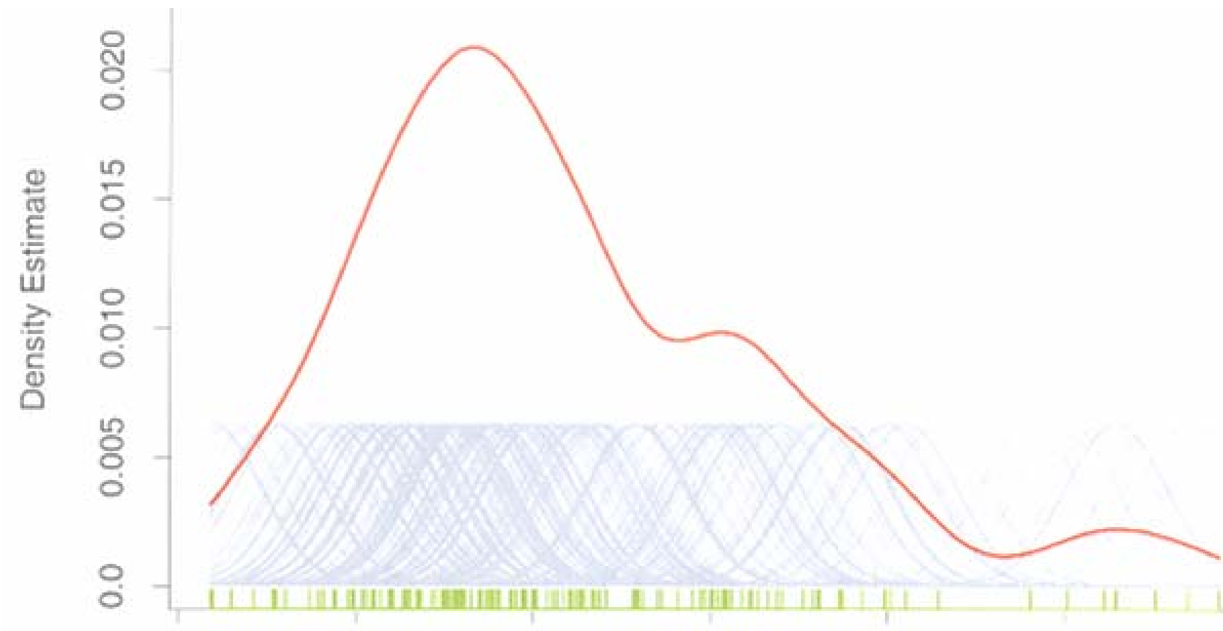
- Use a histogram to represent the class-conditional densities
  - (i.e. $p(x|1)$, $p(x|2)$, etc)
- Advantage: Estimates converge towards correct values with enough data
- Disadvantage: Histogram becomes big with high dimension so requires too much data
  - but maybe we can assume feature independence?

Example Histograms



Kernel Density Estimation

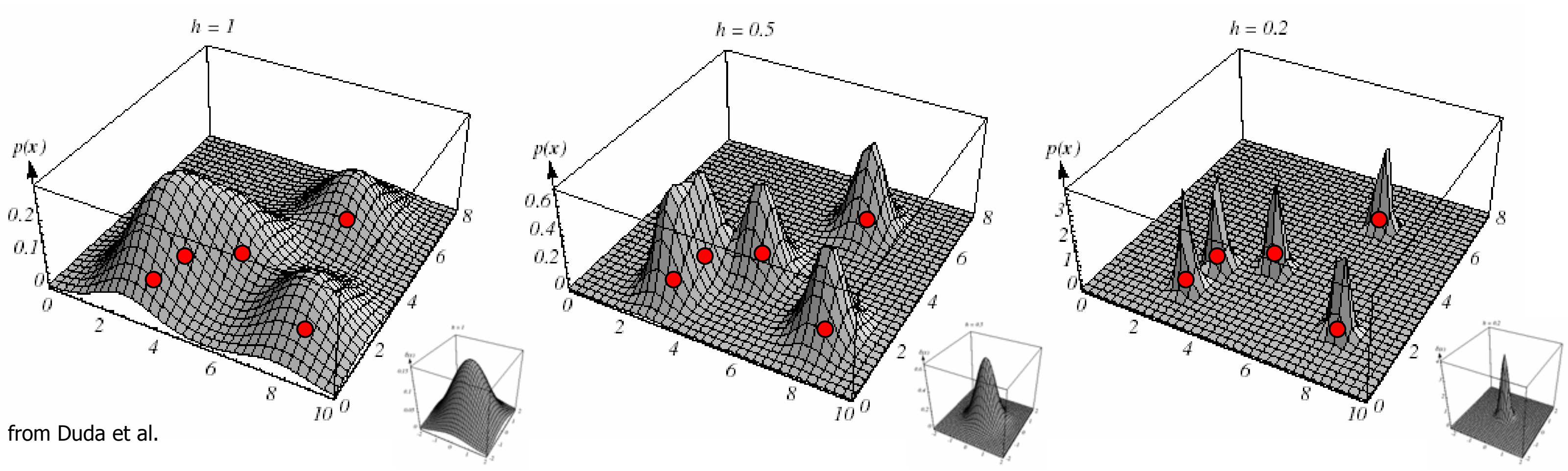
- **Parzen windows**: Approximate probability density by estimating local density of points (same idea as a histogram)
  - Convolve points with window/kernel function (e.g., Gaussian) using scale parameter (e.g., sigma)



from Hastie *et al.*

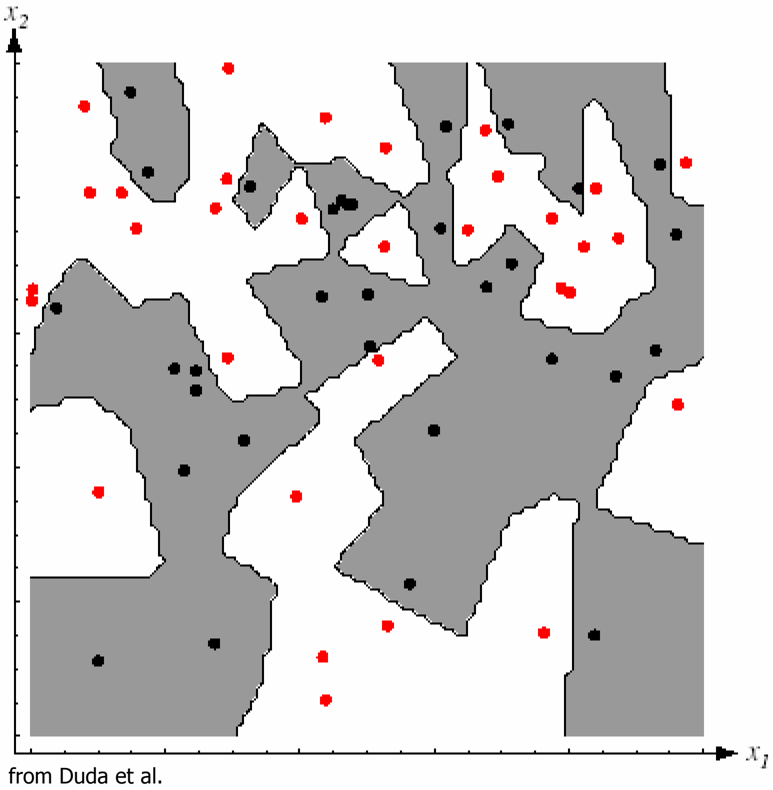
Density Estimation at Different Scales

- Example: Density estimates for 5 data points with differently-scaled kernels
- Scale influences accuracy vs. generality (overfitting)

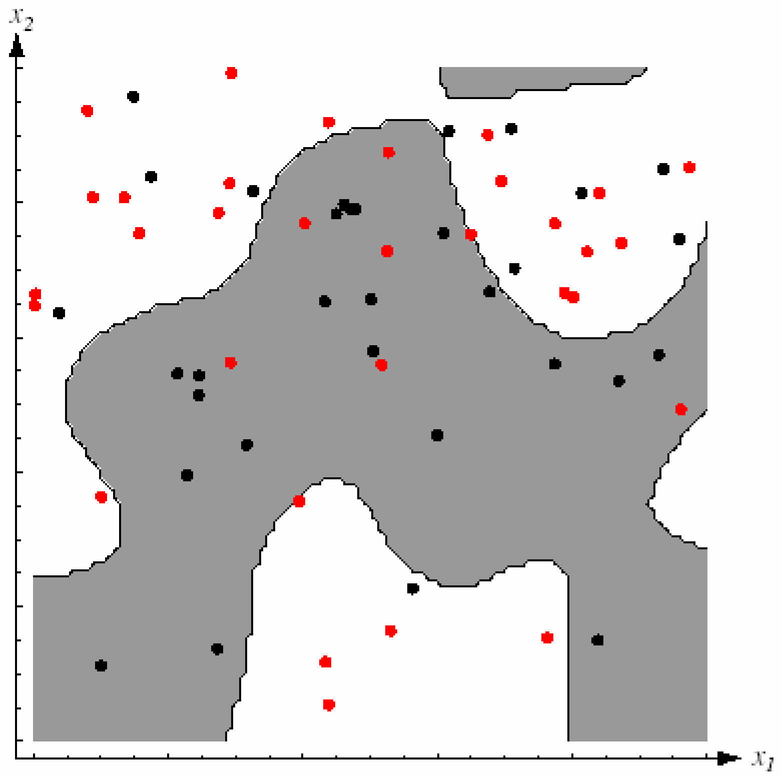


from Duda et al.

Example: Kernel Density Estimation Decision Boundaries



Smaller



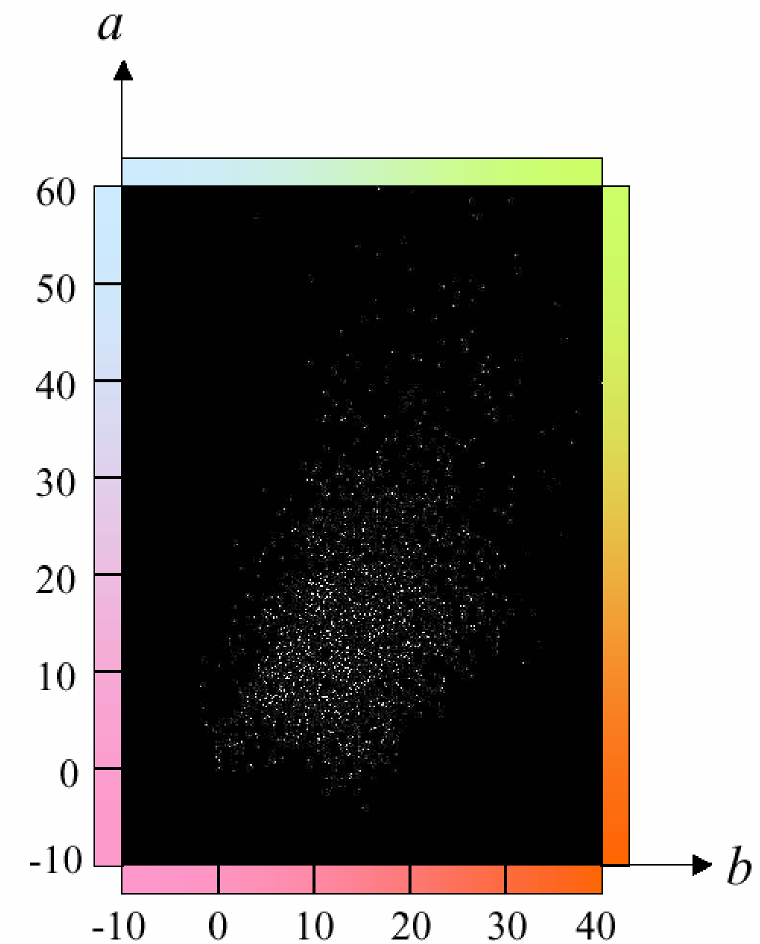
Larger

Application: Skin Colour Histograms

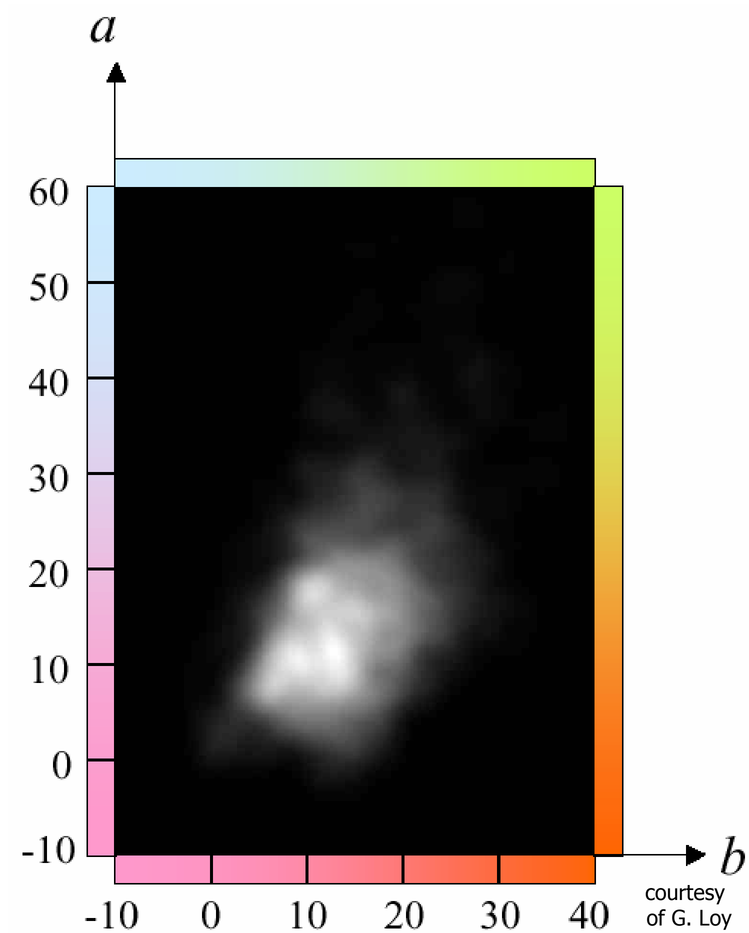
- Skin has a very small range of (intensity independent) colours, and little texture
  - Compute colour measure, check if colour is in this range, check if there is little texture (median filter)
  - Get class conditional densities (histograms), priors from data (counting)
- Classifier is

- if $p(\text{skin}|\boldsymbol{x}) > \theta$, classify as skin
- if $p(\text{skin}|\boldsymbol{x}) < \theta$, classify as not skin
- if $p(\text{skin}|\boldsymbol{x}) = \theta$, choose classes uniformly and at random

Skin Colour Models



Skin chrominance points



Smoothed, [0,1]-normalized

Skin Colour Classification

For every pixel $\mathbf{p}_i$ in $\mathbf{I}_{test}$

- Determine the chrominance values (a_i, b_i) of $\mathbf{I}_{test}(\mathbf{p}_i)$
- Lookup the skin likelihood for (a_i, b_i) using the skin chrominance model.
- Assign this likelihood to $\mathbf{I}_{skin}(\mathbf{p}_i)$

$\mathbf{I}_{test}$

$\mathbf{I}_{skin}$

courtesy of G. Loy

Results

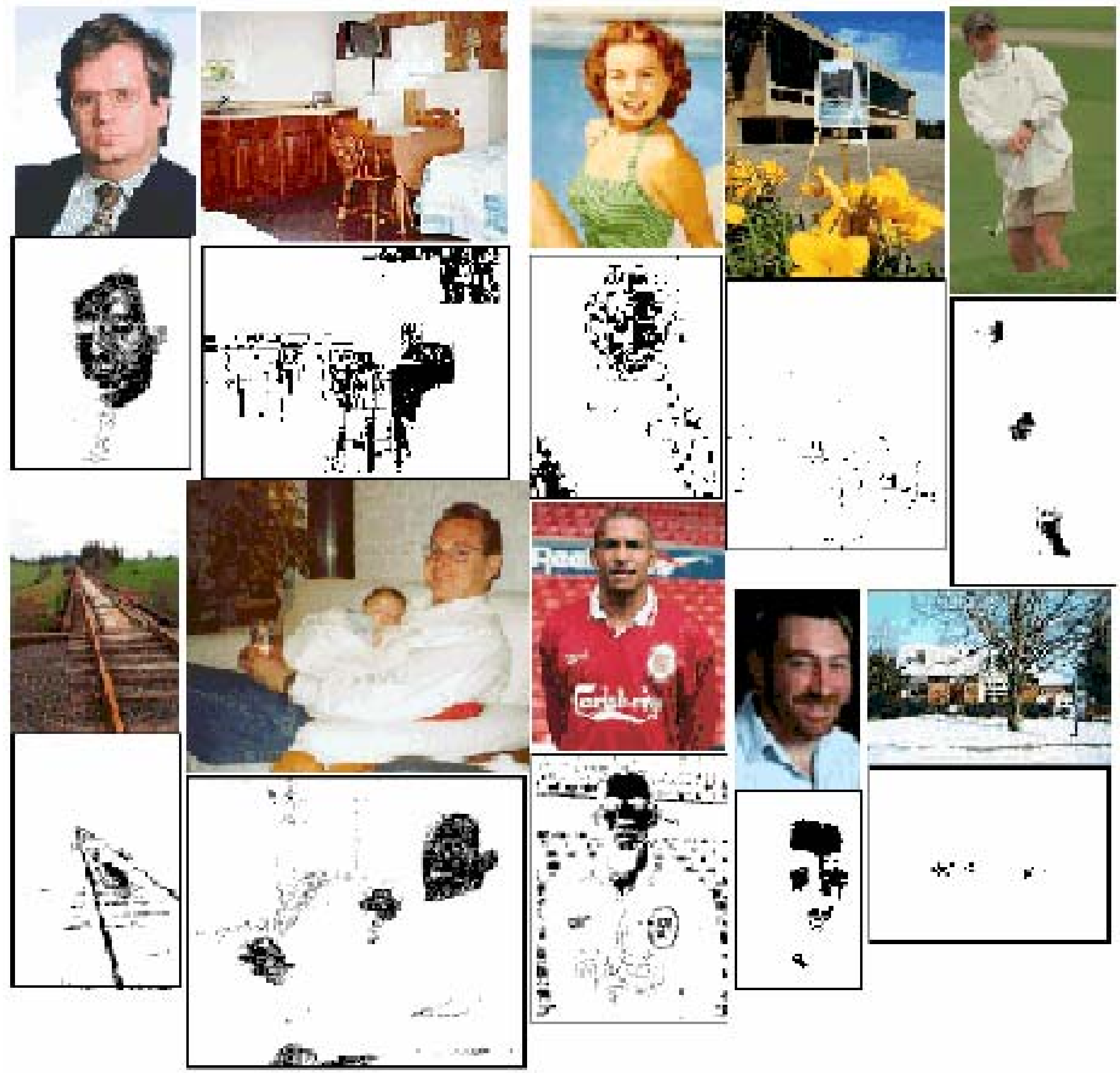


Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 copyright 1999, IEEE

ROC Curves
(Receiver operating characteristics)

Plots trade-off between false positives and false negatives

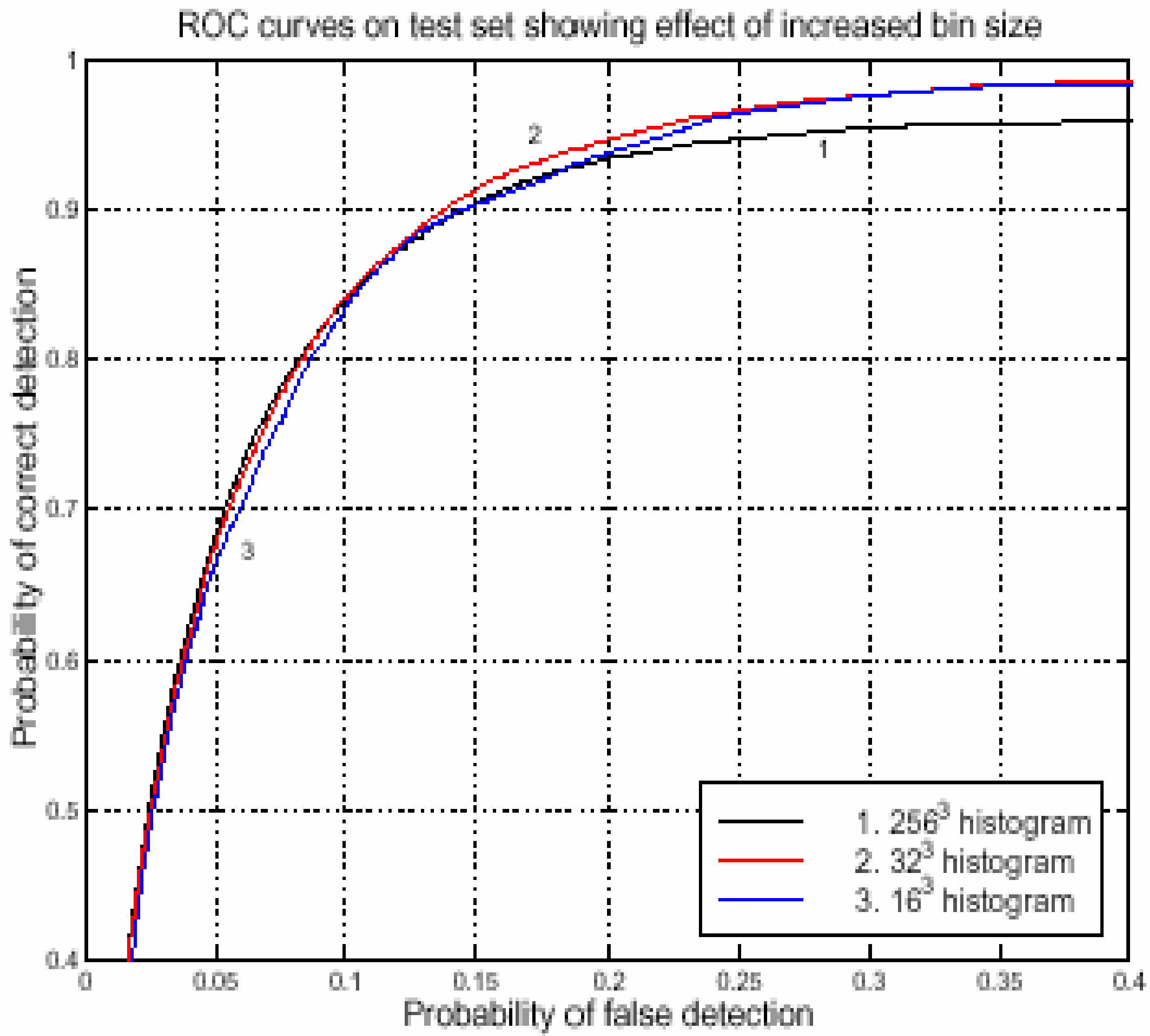
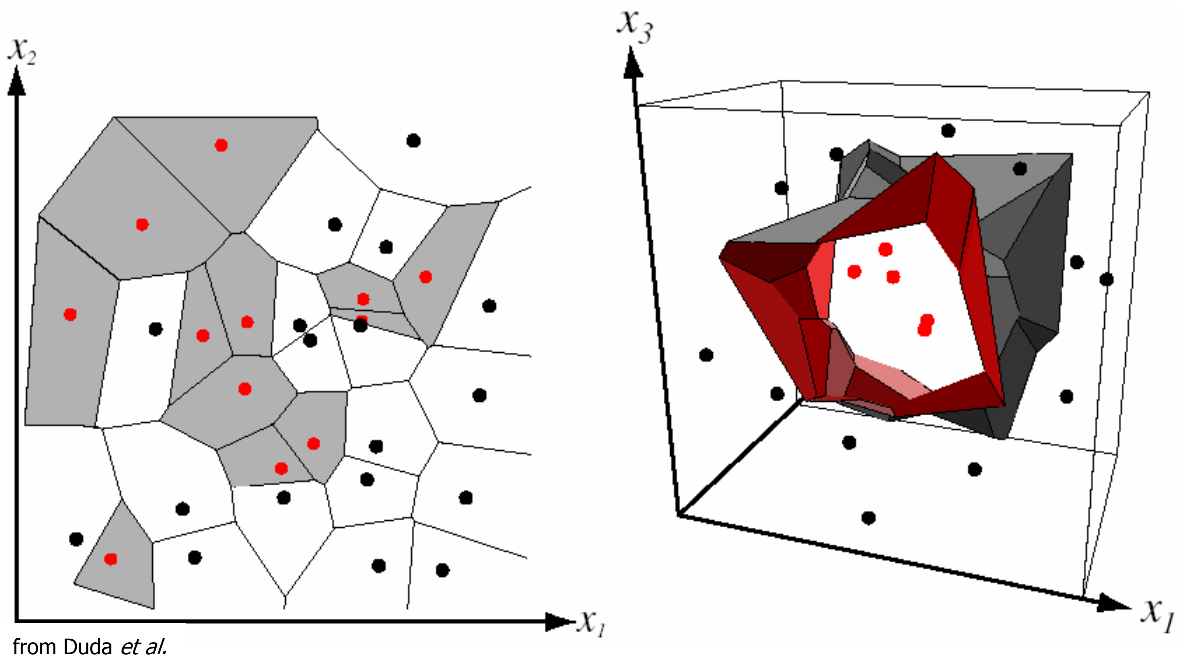


Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 copyright 1999, IEEE

Nearest Neighbor Classifier

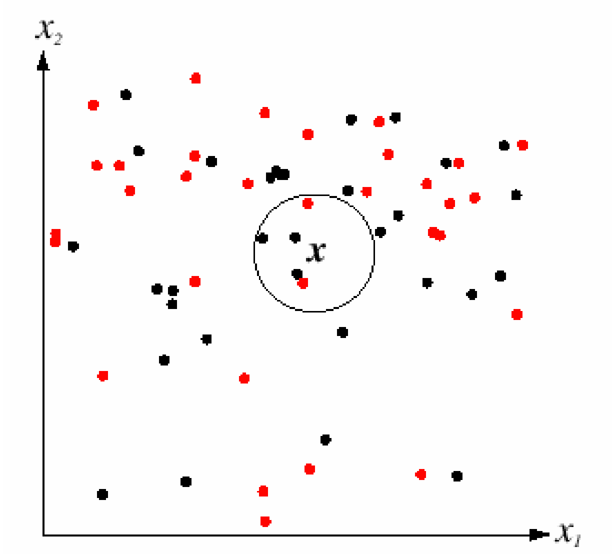
- Assign label of nearest training data point to each test data point



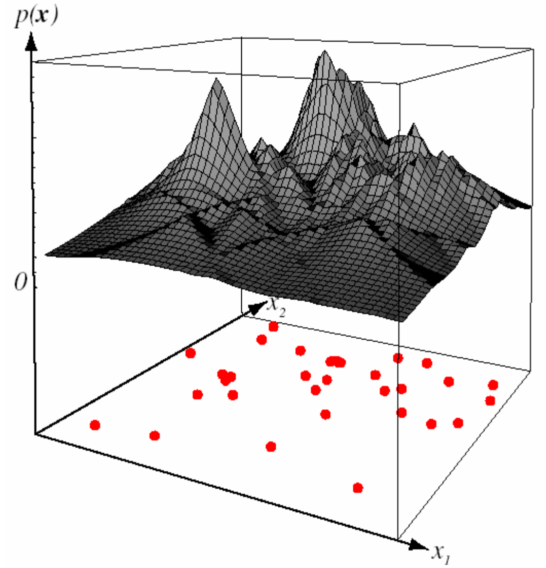
Voronoi partitioning of feature space
for 2-category 2-D and 3-D data

K-Nearest Neighbors

- For a new point, find the k closest points from training data
- Labels of the k points "vote" to classify
- Avoids fixed scale choice—uses data itself (can be very important in practice)
- Simple method that works well if the distance measure correctly weights the various dimensions



$k = 5$



from
Duda et al.

Example density estimate

Face Recognition

- Introduction
- Face recognition algorithms
- Comparison
- Short summary of the presentation

Introduction

- Why we are interested in face recognition?
  - Passport control at terminals in airports
  - Participant identification in meetings
  - System access control
  - Scanning for criminal persons

Face Recognition Algorithms

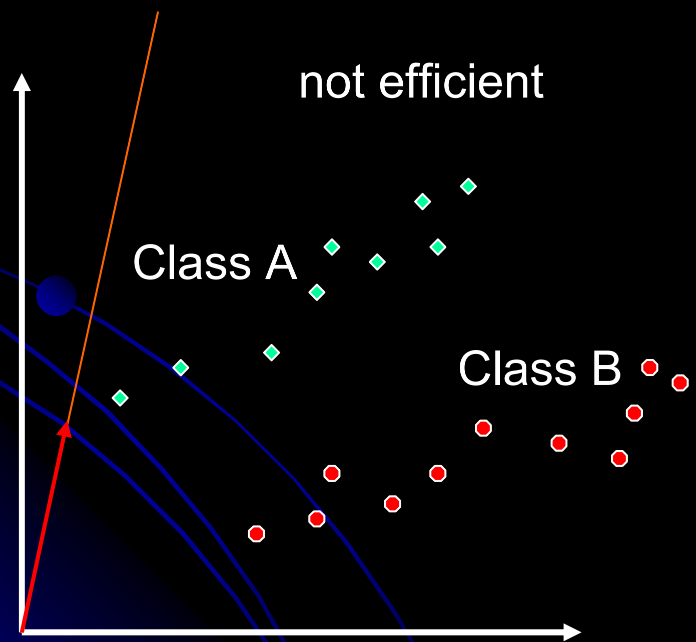
- In this presentation are introduced
  - Eigenfaces
  - F

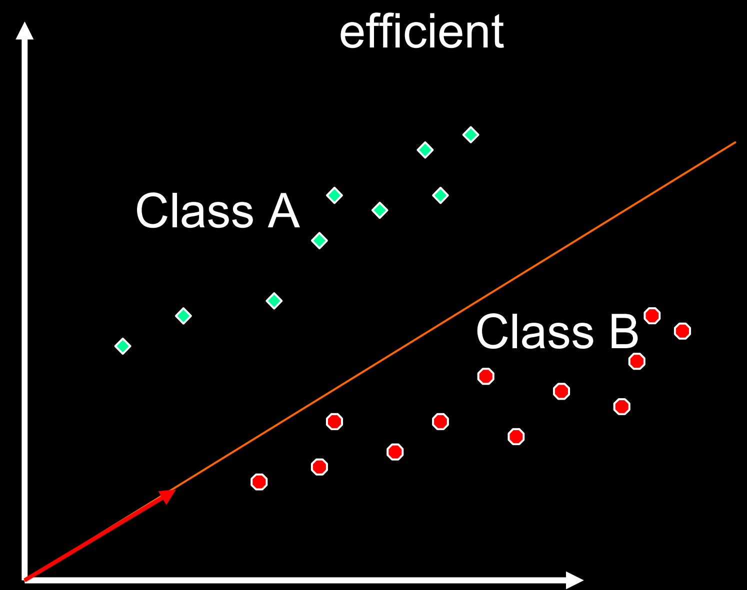
Eigenfaces

- Developed in 1991 by M.Turk
- Based on PCA
- Relatively simple
- Fast
- Robust

Eigenfaces

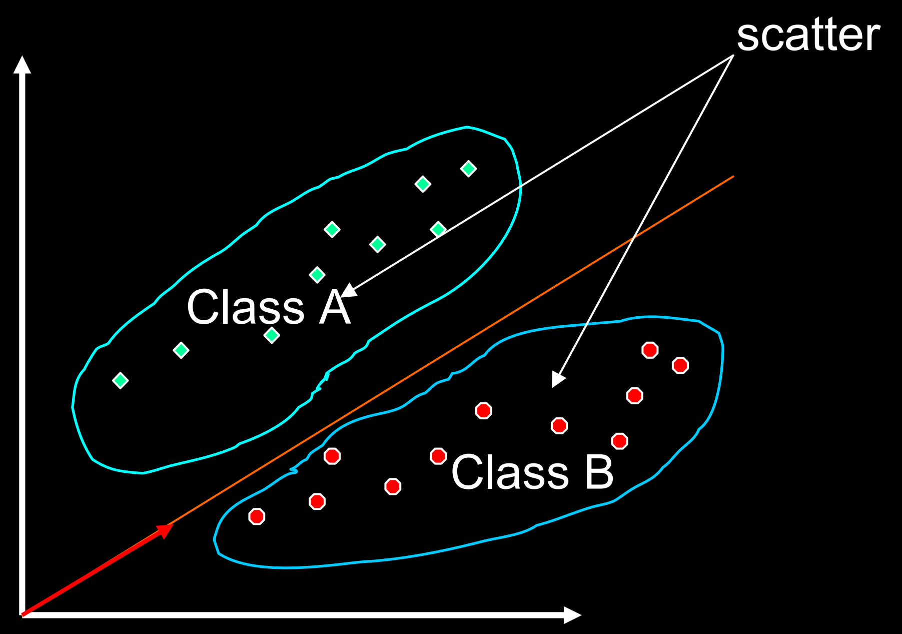
- PCA seeks directions that are efficient for representing the data





Eigenfaces

- PCA maximizes the total scatter



Eigenfaces

- PCA reduces the dimension of the data
- Speeds up the computational time

Eigenfaces, the algorithm

- Assumptions
  - Images with $W \times H = N^2$
  - M is the number of images in the database
  - P is the number of persons in the database

Eigenfaces, the algorithm

- The database

$$= \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- We compute the average face

$$\vec{m} = \frac{1}{M}\begin{pmatrix} a_1 & +b_1 & +\cdots+h_1 \\ a_2 & +b_2 & +\cdots+h_2 \\ \vdots & \vdots & \vdots \\ a_{N^2} & +b_{N^2} & +\cdots+h_{N^2} \end{pmatrix}, \quad where\ M = 8$$

Eigenfaces, the algorithm

- Then subtract it from the training faces

$$\vec{a}_m = \begin{pmatrix} a_1 & - & m_1 \\ a_2 & - & m_2 \\ \vdots & & \vdots \\ a_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{b}_m = \begin{pmatrix} b_1 & - & m_1 \\ b_2 & - & m_2 \\ \vdots & & \vdots \\ b_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{c}_m = \begin{pmatrix} c_1 & - & m_1 \\ c_2 & - & m_2 \\ \vdots & & \vdots \\ c_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{d}_m = \begin{pmatrix} d_1 & - & m_1 \\ d_2 & - & m_2 \\ \vdots & & \vdots \\ d_{N^2} & - & m_{N^2} \end{pmatrix},$$

$$\vec{e}_m = \begin{pmatrix} e_1 & - & m_1 \\ e_2 & - & m_2 \\ \vdots & & \vdots \\ e_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{f}_m = \begin{pmatrix} f_1 & - & m_1 \\ f_2 & - & m_2 \\ \vdots & & \vdots \\ f_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{g}_m = \begin{pmatrix} g_1 & - & m_1 \\ g_2 & - & m_2 \\ \vdots & & \vdots \\ g_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{h}_m = \begin{pmatrix} h_1 & - & m_1 \\ h_2 & - & m_2 \\ \vdots & & \vdots \\ h_{N^2} & - & m_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- Now we build the matrix which is N^2 by M

$$A = \left[\vec{a}_m \; \vec{b}_m \; \vec{c}_m \; \vec{d}_m \; \vec{e}_m \; \vec{f}_m \; \vec{g}_m \; \vec{h}_m \right]$$

- The covariance matrix which is N^2 by N^2

$$Cov = AA^{\mathrm{T}}$$

Eigenfaces, the algorithm

- Find eigenvalues of the covariance matrix
  - The matrix is very large
  - The computational effort is very big

- We are interested in at most M eigenvalues
  - We can reduce the dimension of the matrix

Eigenfaces, the algorithm

- Compute another matrix which is M by M

$$L = A^{\mathrm{T}} A$$

- Find the M eigenvalues and eigenvectors
  - Eigenvectors of Cov and L are equivalent
- Build matrix V from the eigenvectors of L

Eigenfaces, the algorithm

- Eigenvectors of Cov are linear combination of image space with the eigenvectors of L

$$U = AV$$

- Eigenvectors represent the variation in the faces

Eigenfaces, the algorithm

- Compute for each face its projection onto the face space

$$\Omega_1 = U^{\mathrm{T}}\left(\vec{a}_m\right),\quad \Omega_2 = U^{\mathrm{T}}\left(\vec{b}_m\right),\quad \Omega_3 = U^{\mathrm{T}}\left(\vec{c}_m\right),\quad \Omega_4 = U^{\mathrm{T}}\left(\vec{d}_m\right),$$

$$\Omega_5 = U^{\mathrm{T}}\left(\vec{e}_m\right),\quad \Omega_6 = U^{\mathrm{T}}\left(\vec{f}_m\right),\quad \Omega_7 = U^{\mathrm{T}}\left(\vec{g}_m\right),\quad \Omega_8 = U^{\mathrm{T}}\left(\vec{h}_m\right)$$

- Compute the threshold

$$\theta = \frac{1}{2}\max\left\{\left\|\Omega_i - \Omega_j\right\|\right\} \; for \; i, j = 1 .. M$$

Eigenfaces, the algorithm

- To recognize a face

$$= \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N^2} \end{pmatrix}$$

- Subtract the average face from it

$$\vec{r}_m = \begin{pmatrix} r_1 & - & m_1 \\ r_2 & - & m_2 \\ \vdots & & \vdots \\ r_{N^2} & - & m_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- Compute its projection onto the face space

$$\Omega = U^{\mathrm{T}}\left(\vec{r}_m\right)$$

- Compute the distance in the face space between the face and all known faces

$$\varepsilon_i^2 = \left\|\Omega - \Omega_i\right\|^2 \quad for\ i = 1..M$$

Eigenfaces, the algorithm

- Reconstruct the face from eigenfaces

$$\vec{s} = U\Omega$$

- Compute the distance between the face and its reconstruction

$$\xi^2 = \left\|\vec{r}_m - \vec{s}\right\|^2$$

Eigenfaces, the algorithm

- Distinguish between
  - If $\xi \geq \theta$ then it's not a face
  - If $\xi < \theta \; and \; \varepsilon_i \geq \theta, (i = 1..M)$ then it's a new face
  - If $\xi < \theta \; and \; \min\{\varepsilon_i\} < \theta$ then it's a known face

Eigenfaces, the algorithm

- Problems with eigenfaces
  - Different illumination
  - Different head pose
  - Different alignment
  - Different facial expression