Face Recognition

- Face is the most common biometric used by humans
- Applications range from static, mug-shot verification to a dynamic, uncontrolled face identification in a cluttered background
- Challenges:
  - automatically locate the face
  - recognize the face from a general viewpoint under different illumination conditions, facial expressions, and aging effects

Authentication vs Identification

- Face Authentication/Verification (1:1 matching)
- Face Identification/recognition (1:N matching)

Applications

- Access Control
- Video Surveillance (On-line or off-line)
Why is Face Recognition Hard?

Many faces of Madonna

[Image of Madonna's various faces]

Sinha and Poggio 1996

Why are these people?

[Sinha and Poggio 1996]

Face Recognition Difficulties

- Identify similar faces (inter-class similarity)
- Accommodate intra-class variability due to:
  - head pose
  - illumination conditions
  - expressions
  - facial accessories
  - aging effects
  - Cartoon faces

Inter-class Similarity

- Different persons may have very similar appearance

Twins

Father and son

Intra-class Variability

- Faces with intra-subject variations in pose, illumination, expression, accessories, color, occlusions, and brightness
Example: Face Detection

- Scan window over image.
- Classify window as either:
  - Face
  - Non-face

Example: Finding skin

Non-parametric Representation of CCD

- Skin has a very small range of (intensity independent) colors, and little texture
  - Compute an intensity-independent color measure, check if color is in this range, check if there is little texture (median filter)
  - See this as a classifier - we can set up the tests by hand, or learn them.
  - get class conditional densities (histograms), priors from data (counting)

Classifier is

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

Face Detection Algorithm

Input Image

Face Detection Algorithm

Face Localization

Lighting Compensation

Skin Color Detection

Variance-based Segmentation

Connected Component & Grouping

Eye/Mouth Detection

Face Boundary Detection

Verifying/Weighting Eyes-Mouth Triangles

Face Detection

Output Image

Face Recognition: 2-D and 3-D

Time (video)

Recognition Comparison

2-D

3-D

Face Database

Prior knowledge of face class

Recognition Data

Taxonomy of Face Recognition

Algorithms

Pose-dependent

Pose-invariant

Object-centered Models

Face Representation

Appearance-based

Hybrid

Feature-based

PCA, LDA

LFA

EDGM

--- Gordon et al., 1995
--- Langagne et al., 1996
--- Atick et al., 1996
--- Yan et al., 1996
--- Zhao et al., 2000
--- Zhang et al., 2000

Image as a Feature Vector

Consider an n-pixel image to be a point in an n-dimensional space, \( x \in \mathbb{R}^n \).

Each pixel value is a coordinate of \( x \).
Nearest Neighbor Classifier

\{ R_j \} are set of training images.

\[ ID = \min \{ \text{dist}(R_j, I) \} \]

Comments

- Sometimes called “Template Matching”
- Variations on distance function (e.g. L1, robust distances)
- Multiple templates per class - perhaps many training images per class.
- Expensive to compute \( k \) distances, especially when each image is big (N dimensional).
- May not generalize well to unseen examples of class.
- Some solutions:
  - Bayesian classification
  - Dimensionality reduction

Eigenface (Turk, Pentland, 91) -1

- Use Principle Component Analysis (PCA) to determine the most discriminating features between images of faces.

Eigenfaces: linear projection

- An \( n \)-pixel image \( x \in \mathbb{R}^n \) can be projected to a low-dimensional feature space \( y \in \mathbb{R}^m \) by

\[ y = W^T x \]

where \( W \) is an \( n \times m \) matrix.
- Recognition is performed using nearest neighbor in \( \mathbb{R}^m \).
- How do we choose a good \( W \)?

Eigenfaces: Principal Component Analysis (PCA)

Assume we have a set of \( n \) feature vectors \( x_i (i = 1, \ldots, n) \) in \( \mathbb{R}^d \). Write

- \( \mu = \frac{1}{n} \sum x_i \)

- \( \Sigma = \frac{1}{n-1} \sum (x_i - \mu)(x_i - \mu)^T \)

The unit eigenvectors of \( \Sigma \) — which we write as \( v_1, v_2, \ldots, v_k \), where the order is given by the size of the eigenvalue and \( v_1 \) has the largest eigenvalue — give a set of features with the following properties:

- They are independent.
- Projection onto the basis \( \{ v_1, \ldots, v_k \} \) gives the \( k \)-dimensional set of linear features that preserves the most variance.

Algorithm 22.5: Principal components analysis identifies a collection of linear features that are independent, and capture as much variance as possible from a dataset.

Some details: Use Singular value decomposition, “trick” described in text to compute basis when \( n \ll d \)

How do you construct Eigenspace?

Construct data matrix by stacking vectorized images and then apply Singular Value Decomposition (SVD)
Matrix Decompositions

• Definition: The factorization of a matrix $M$ into two or more matrices $M_1, M_2, \ldots, M_n$, such that $M = M_1 M_2 \ldots M_n$.
• Many decompositions exist…
  – QR Decomposition
  – LU Decomposition
  – LDU Decomposition
  – Etc.

Singular Value Decomposition

Excellent ref: ‘Matrix Computations,’ Golub, Van Loan

• Any $m$ by $n$ matrix $A$ may be factored such that
  $$ A = U \Sigma V^T $$
  $$ [m \times n] = [m \times m][m \times n][n \times n] $$
• $U$: $m$ by $m$, orthogonal matrix
  – Columns of $U$ are the eigenvectors of $AA^T$
• $V$: $n$ by $n$, orthogonal matrix,
  – Columns are the eigenvectors of $A^TA$
• $\Sigma$: $m$ by $n$, diagonal with non-negative entries ($\sigma_1, \sigma_2, \ldots, \sigma_s$) with $s = \min(m,n)$ are called the singular values
  – Singular values are the square roots of eigenvalues of both $AA^T$ and $A^TA$
• Result of SVD algorithm: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s$

SVD Properties

• In Matlab $[u s v] = \text{svd}(A)$, and you can verify that $A = u^*s^*v^*$
• $r = \text{Rank}(A) = \# \text{ of non-zero singular values.}$
• $U, V$ give us orthonormal bases for the subspaces of $A$:
  – $1 \leq r$ columns of $U$: Column space of $A$
  – Last $m - r$ columns of $U$: Left nullspace of $A$
  – $1 \leq r$ columns of $V$: Row space of $A$
  – Last $n - r$ columns of $V$: Nullspace of $A$
• For $d \leq r$, the first $d$ column of $U$ provide the best $d$-dimensional basis for columns of $A$ in least squares sense.

Thin SVD

• Any $m$ by $n$ matrix $A$ may be factored such that
  $$ A = U \Sigma V^T $$
  $$ [m \times n] = [m \times m][n \times n][n \times n] $$
• If $m > n$, then one can view $\Sigma$ as:
  $$ \Sigma = \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} $$
• Alternatively, you can write:
  $$ A = U' \Sigma' V^T $$
• In Matlab, thin SVD is $[U S V] = \text{svds}(A)$

Application: Pseudoinverse

• Given $y = Ax$, $x = A^+ y$
• For square $A$, $A^+ = A^{-1}$
• For any $A$…
  $$ A^+ = V \Sigma^1 U^T $$
• $A^+$ is called the pseudoinverse of $A$.
• $x = A^+ y$ is the least-squares solution of $y = Ax$.
• Alternative to previous solution.

Performing PCA with SVD

• Singular values of $A$ are the square roots of eigenvalues of both $AA^T$ and $A^TA$ & Columns of $U$ are corresponding Eigenvectors
• And $\sum a_i a_i^T = [a_1, a_2, \ldots, a_n][a_1, a_2, \ldots, a_n]^T = AA^T$
• Covariance matrix is:
  $$ \Sigma = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T $$
• So, ignoring $1/n$, subtract mean image $\mu$ from each input image, create data matrix, and perform thin SVD on the data matrix.
Eigenfaces

- **Modeling**
  1. Given a collection of n labeled training images,
  2. Compute mean image and covariance matrix.
  3. Compute k Eigenvectors (note that these are images) of covariance matrix corresponding to k largest Eigenvalues.
  4. Project the training images to the k-dimensional Eigenspace.

- **Recognition**
  1. Given a test image, project to Eigenspace.
  2. Perform classification to the projected training images.

Eigenfaces: Training Images

Eigenfaces

Variable Lighting

Projection, and reconstruction

- An n-pixel image $x \in \mathbb{R}^n$ can be projected to a low-dimensional feature space $y \in \mathbb{R}^m$ by

  $y = Wx$

- From $y \in \mathbb{R}^m$, the reconstruction of the point is $W^Ty$

- The error of the reconstruction is: $||x - W^TWx||$
Reconstruction using Eigenfaces

- Given image on left, project to Eigenspace, then reconstruct an image (right).