Dimension Reduction

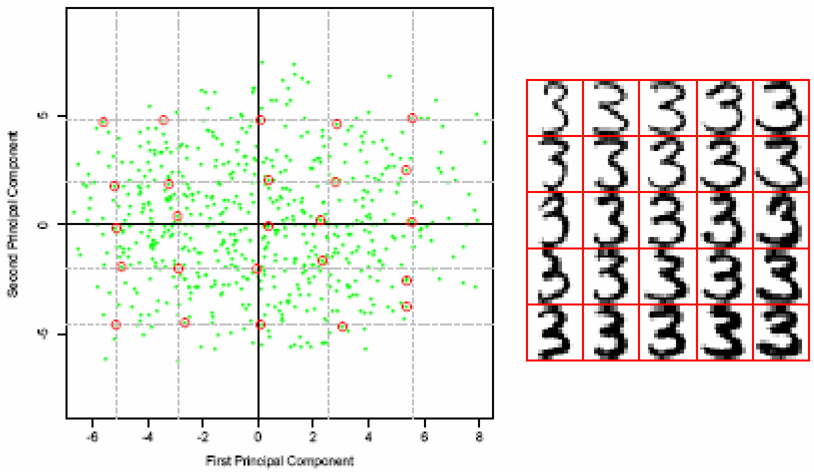
Dr. Zhang Hongxin
State key lab of CAD&CG, ZJU
2005-06-23

Introduction

- Goal: choosing suitable transforms, so as to obtain high “information packing”.
  - Raw data -> Meaningful features.
  - Unsupervised/Automatic methods.
- To exploit and remove information redundancies via transform.

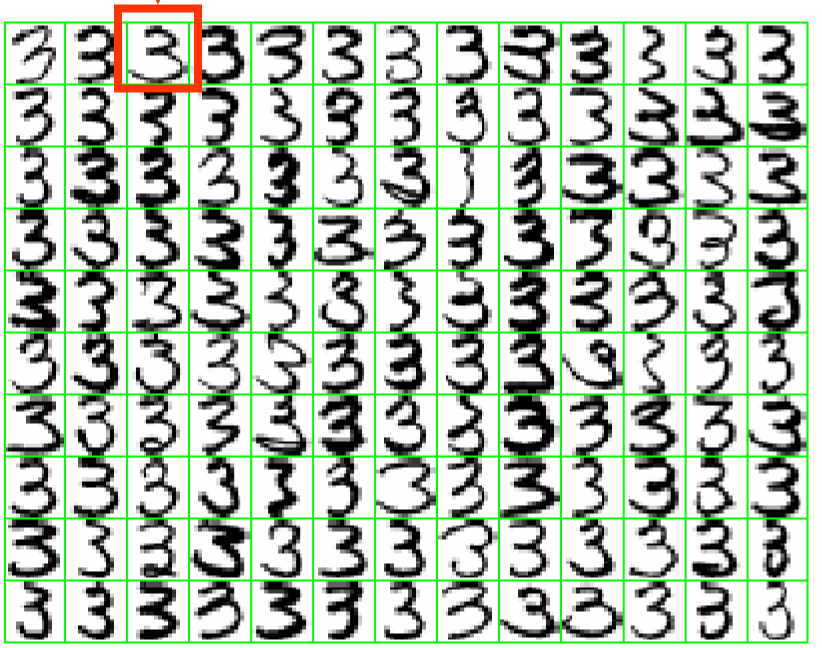
Feature extraction

- Data independent
  - DFT, DWT, DCT
    - A single piece of signal



- Data dependent
  - PCA, K-PCA, ICA, ISO-MAP, LLE …
    - A set of signals (images, motion data, shapes,…)
- Key: define desirable transforms
  - Raw data -> Feature space

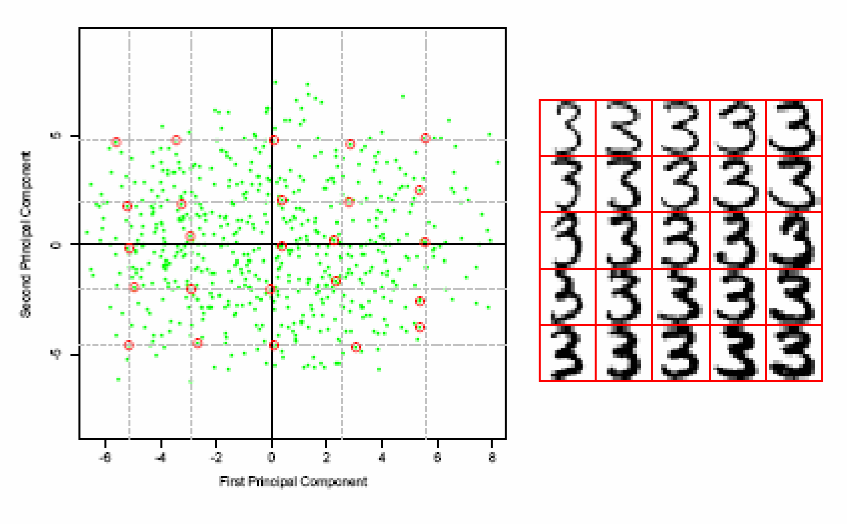
PCA: example
Digit data



$$\boldsymbol{X} = \begin{pmatrix} x_{0,0} & x_{1,0} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0,p-1} & x_{1,p-1} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{N\times p}$$

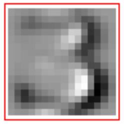
130 threes, a subset of 638 such threes and part of the handwritten digit dataset. Each three is a 16 × 16 greyscale image, and the variables Xj, j = 1, . . . , 256 are the greyscale values for each pixel.

Digit: rank-2 model for threes



Two-component model has the form

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$=$  $+ \lambda_1 \cdot$  $+ \lambda_2 \cdot$  .

Here we have displayed the first two principal component directions, v_1 and v_2, as images.

Principal Components

- Suppose we have *N* measurements on each of *p* variables $X_j, \quad j = 1, ..., k$. There are several equivalent approaches to principal components:
  - Produce a derived (and small) set of uncorrelated variables $Z_k = a_k^T X, k = 1, ..., q < p$ that are linear combinations of the original variables, and that explain most of the variation in the original set.
  - Approximate the original set of *N* points in $\Re^p$ by a least-squares optimal linear manifold of co-dimension $q < p$.
  - Approximate the $q < p$ data matrix $\boldsymbol{X}$ by the best rank-q matrix $\hat{\boldsymbol{X}}(p)$. This is the usual motivation for the SVD.

Basis Vectors and Images

- Input samples

$$\mathbf{X}^T = [X(1), X(2), \ldots, X(p)]$$

- Unitary *p*xp matrix $\mathbf{A}$ and transformed Vector

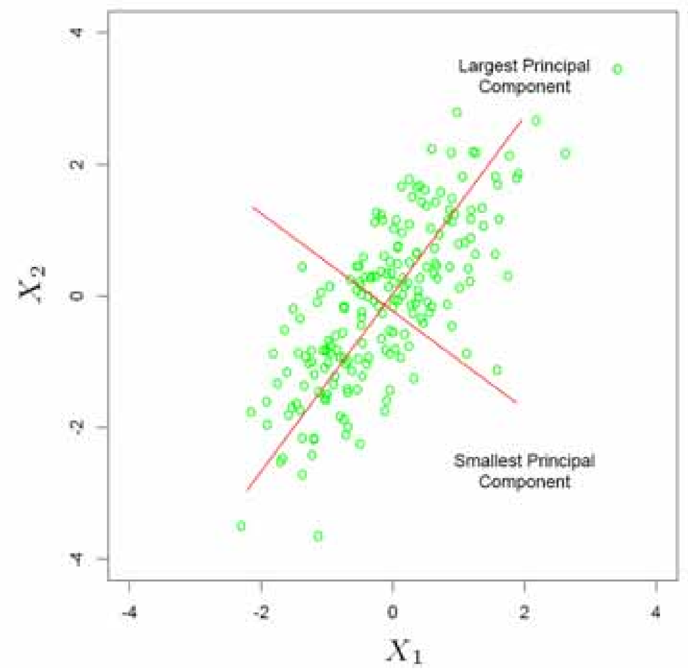
$$\mathbf{Z} = \mathbf{A}\mathbf{X}$$

- Basis vector representation

$$\mathbf{x} = \mathbf{A}\mathbf{z} = \sum_{i=0}^{N-1} z(i)\mathbf{a}_i$$

$$< \mathbf{a}_j, \mathbf{x} >= \mathbf{a}_j^T \mathbf{x} = \sum_{i=1}^{p} z(i) < \mathbf{a}_j, \mathbf{a}_i > = z(j)$$

PCA:
Derived Variables



$$\mathbf{\Sigma} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$$

- $Z_1 = a_1 X$ is the projection of the data onto the longest direction, and has the largest variance amongst all such normalized projections.
- α_1 is the largest eigenvalue of Σ, the sample covariance matrix of X. Z_2 and α_2 correspond to the second-largest eigenvector.

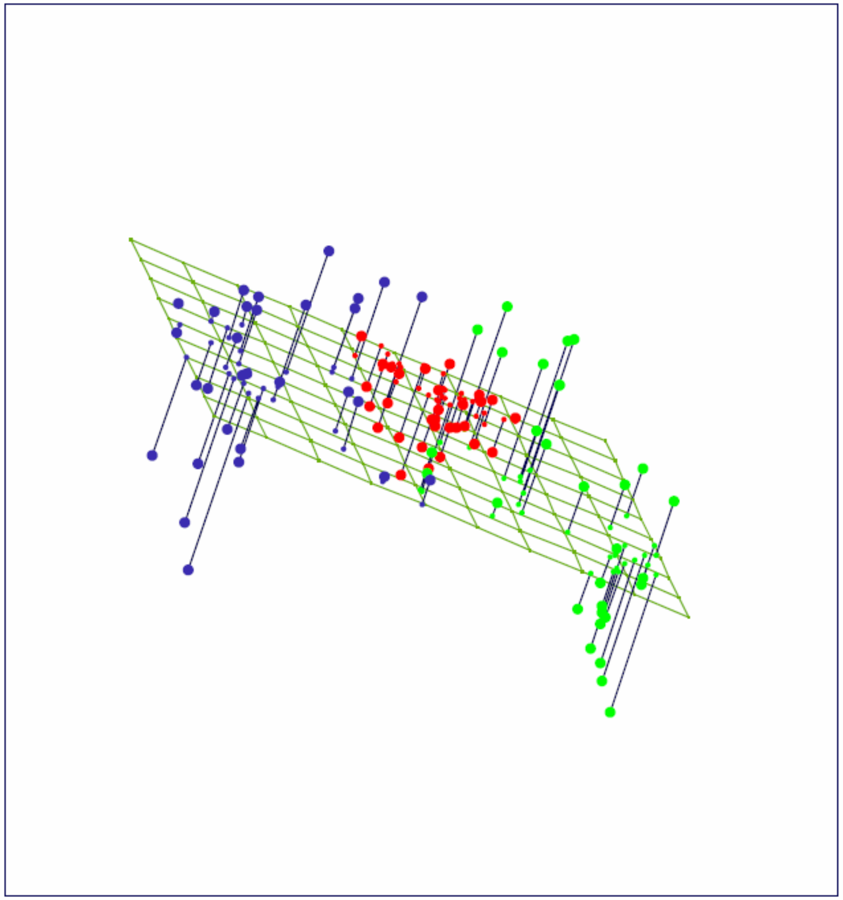
PCA:
Least Squares Approximation

Find the linear manifold

$$f(\lambda) = \mu + \boldsymbol{V}_q \lambda$$

that best approximates the data in a least-squares sense:

$$\min_{\mu, \{\lambda_i\}, \boldsymbol{V}_q} \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - \mu - \boldsymbol{V}_q \lambda_i \right\|$$



Solution:

$$\mu = \overline{x}, v_k = a_k, \lambda_k = Z_k$$

PCA:
Singular Value Decomposition

Let $\hat{X}$ be the centered $N \times p$ data matrix (assume $N > p$).

Singular values

$$\boldsymbol{X} = \begin{pmatrix} x_{0,0} & \underset{x_1}{\boxed{\begin{matrix} x_{1,0} \\ x_{1,1} \\ \vdots \\ x_{1,p-1} \end{matrix}}} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{0,p-1} & & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{N \times p} = \boldsymbol{USV}$$

Unitary Matrices

is the SVD of $\hat{X}$, where

- U is $N \times p$ orthogonal, the left singular vectors.
- V is $p \times p$ orthogonal, the right singular vectors.
- S is diagonal, with $d_1 \geq d_2 \geq \ldots \geq d_p \geq 0$, the singular values.

✓ The SVD always exists, and is unique up to signs. The columns of V are the principal components, and $Z_j = U_j d_j$.

PCA:
Singular Value Decomposition

$$\boldsymbol{X} = \left(\begin{array}{c|c|cccc} x_{0,0} & x_{1,0} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0,p-1} & x_{1,p-1} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{array}\right)_{N\times p} = \overset{\text{Singular values}}{\underset{\text{Unitary Matrices}}{\boldsymbol{U}\overset{\downarrow}{\boldsymbol{S}}\boldsymbol{V}}}$$

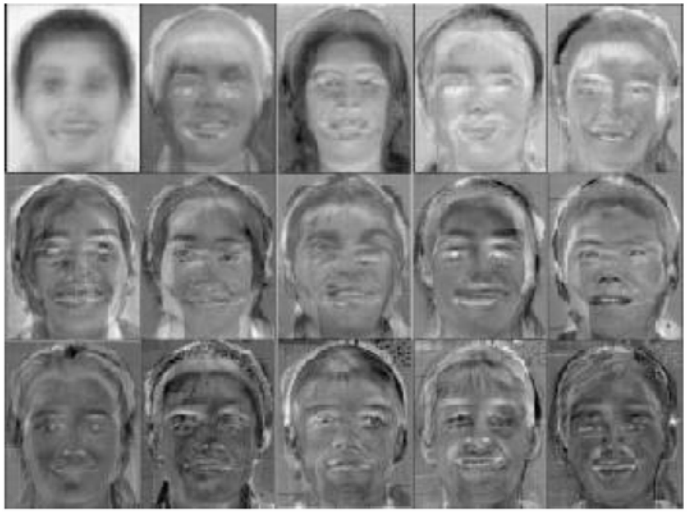
x_1

Let $\boldsymbol{s}_q$ be $\boldsymbol{s}$ with all but the first q diagonal elements set to zero. Then $\hat{\boldsymbol{X}}_q = \boldsymbol{U}\boldsymbol{S}_q\boldsymbol{V}^T$ solves

$$\min_{rank(\hat{X}_q)=q} \left\| \hat{\boldsymbol{X}} - \hat{\boldsymbol{X}}_q \right\|$$

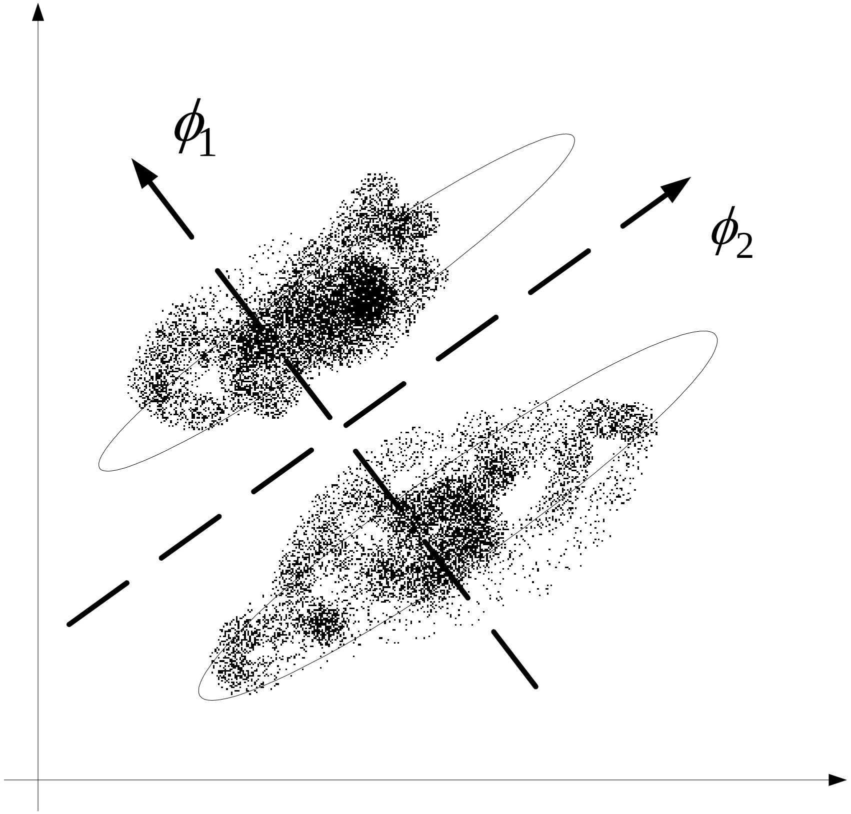
PCA: example Eigenfaces

- G. D. Finlayson, B. Schiele & J. Crowley. Comprehensive colour image normalization. ECCV 98 pp. 475~490.



- Eigen-X, ☺

Problems of PCA



- Only suitable for normal distributed data

- More consideration
  - ICA: Independent components.
  - K-PCA: Nonlinear
  - …

Nonlinear dimension reduction algorithms:

- Locally Linear Embedding (LLE), *Science*

  Sam T. Roweis and Lawrence K. Saul

- A Global Geometric Framework for Nonlinear Dimensionality Reduction (Isomap), *Science*

  Joshua B. Tenenbaum, Vin de Silva, John C. Langford

- BoostMap: A Method for Efficient Approximate Similarity Rankings, *CVPR 2004*

  Vassilis Athitsos, Jonathan Alon, Stan Sclaroff, and George Kollios

Locally Linear Embedding (LLE)

- Recovers global nonlinear structure from locally linear fits.
- Each data point and it’s neighbors is expected to lie on or close to a locally linear patch.
- Each data point is constructed by it’s neighbors:

$$\vec{\hat{X}}_i = \sum_j W_{ij} \vec{X}_j$$

$$W_{ij} = 0 \ \text{ if } \ \vec{X}_j \ \text{ is not a neighbor of } \ \vec{X}_i$$

LLE:
Getting the Reconstruction Weights

- We want to minimize the error function:

$$\varepsilon(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- With the constrains:

$$W_{ij} = 0 \quad \text{if} \quad \vec{X}_j \text{ is not a neighbor of } \vec{X}_i$$

$$\sum_j W_{ij} = 1$$

  - Solution (using lagrange multipliers):

$$W_j = \sum_k C_{jk}^{-1}(\vec{X}\vec{\eta}_k + \lambda)$$

$$\lambda = 1 - \sum_{jk} C_{jk}^{-1}(\vec{X}\vec{\eta}_k) \Big/ \sum_{jk} C_{jk}^{-1}$$

LLE: Find Embedded Coordinates

- Choose d-dimensional coordinates, Y, to minimize: $\phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2$

  Under: $\sum_i \vec{Y}_i = \vec{0}, \quad \frac{1}{\mathrm{N}} \sum_i \vec{Y}\vec{Y}^T = I$

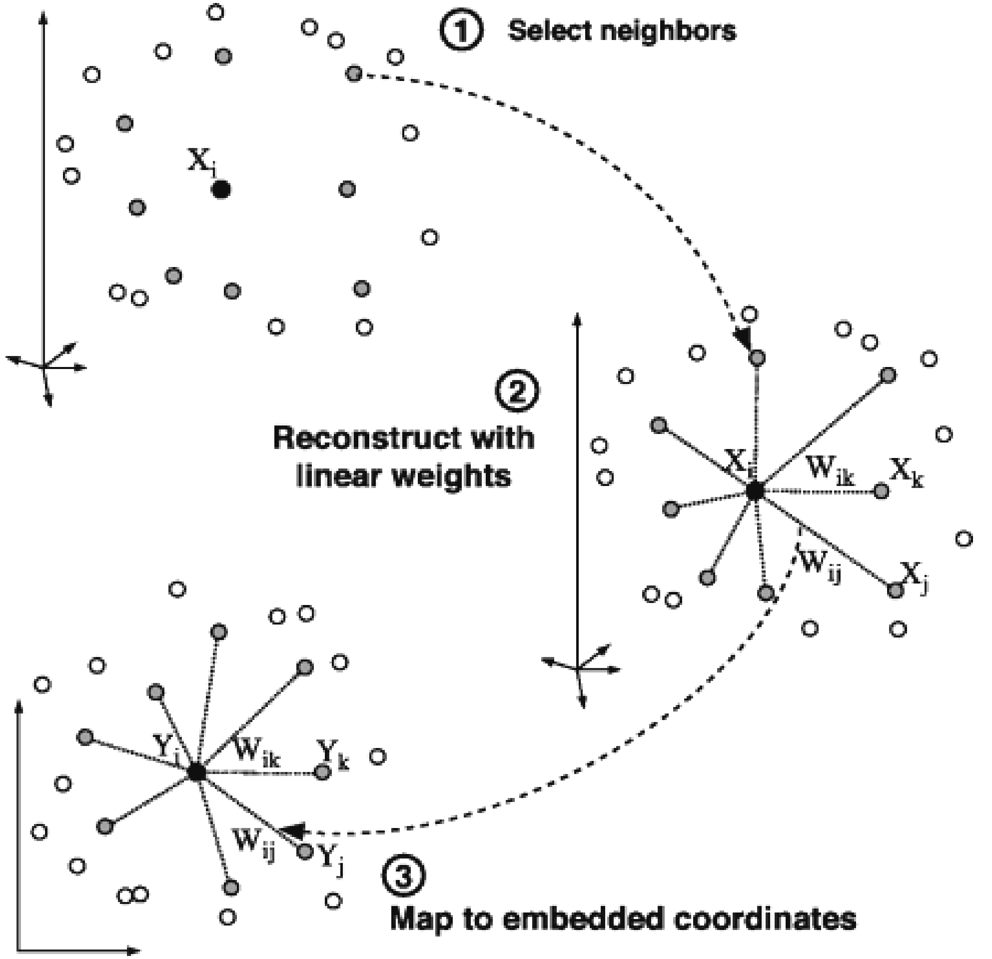
  Quadratic form: $\phi(Y) = \sum_{\mathrm{ij}} \mathrm{M}_{\mathrm{ij}} (\vec{Y}_i \vec{Y}_j)$

  where: $M = (I - W)^T (I - W)$

- Solution: compute bottom d+1 eigenvectors of M. (discard the last one)

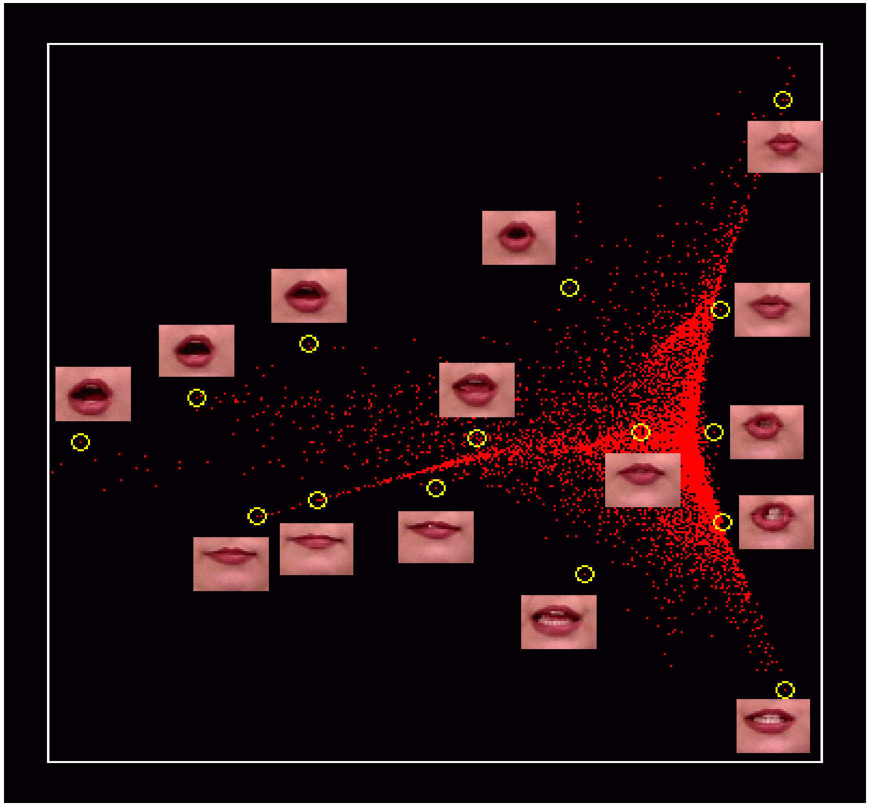
LLE: Summary

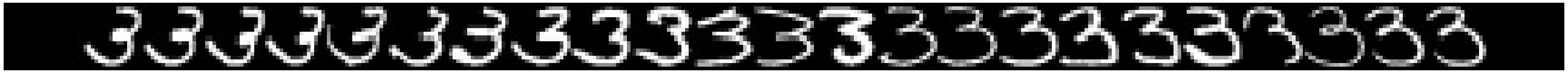
- Input: N data items in D dimension (X).
- Output: d < D dimensional embedding coordinates (Y) for the input points.



LLE: Example

- N=8588 (RGB) images of lips of size 108x84. D=27216
- Num of neighbors K=16

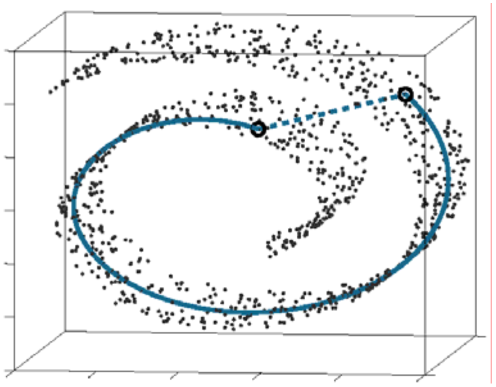


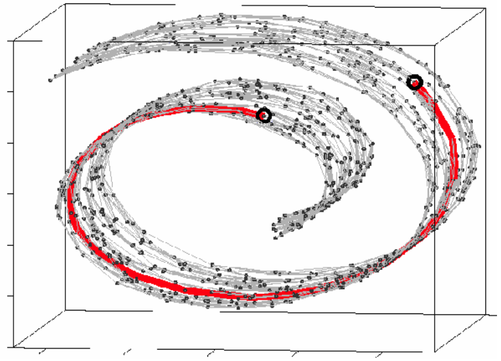


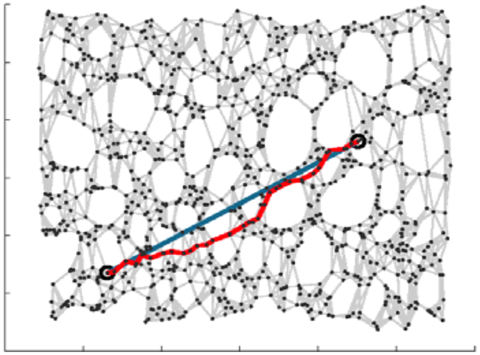
Isomap: (Science 2001)
Isometric feature mapping

- Preserve the intrinsic geometry of the data.
- Use the geodesic manifold distances between all pairs.

Three steps algorithm

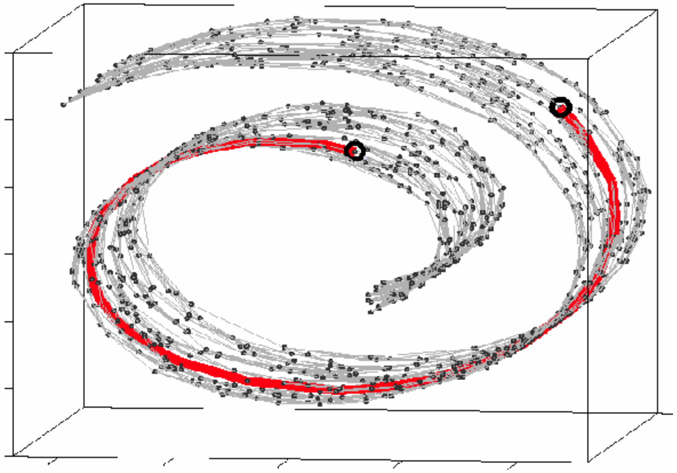






Isomap:
Construct Neighborhood Graph

- Determine which points are neighbors, based on the distances d(i,j) .
  - K nearest neighbors
  - ε-radius

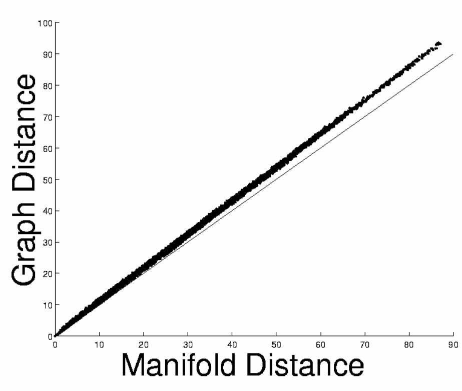


- Create a graph G, with edges between neighbors and distance weights.

Isomap:
Compute Shortest Paths

- Estimate the geodesic distances.
- Compute all-pairs shortest paths in G.
- Can be done using Floyd’s algorithm, $O(N^2 \ln N)$.

$$d_G(i,j) = d(i,j) \quad \text{neighboring i, j}$$
$$d_G(i,j) = \infty \qquad \text{othewise}$$



$$\mathit{for}\ \mathrm{k} = 1,2,\ldots,\mathrm{N}$$
$$d_G(i,j) = \min\{d_G(i,j),\ d_G(i,k) + d_G(k,j)\}$$

Isomap:
Construct d-dimensional Embedding

Classical MDS with $d_G(i,j)$,
minimize the cost function:

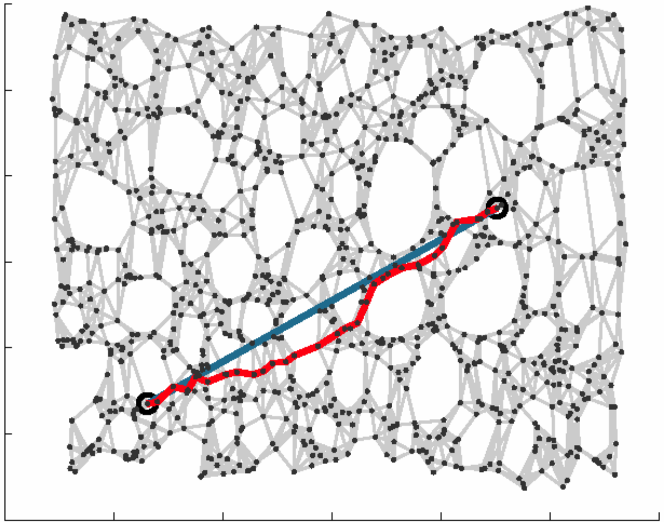
$$E = \left\| \tau(D_G) - \tau(D_Y) \right\|_{L^2}$$

$$\text{where } D_Y(i,j) = \left\| y_i - y_j \right\|$$

$$D_G(i,j) = d_G(i,j)$$

and

$$\tau(D) = \tfrac{-1}{2}(I - \tfrac{1}{N}) D^{.2} (I - \tfrac{1}{N})$$



Solution: take top d eigenvectors of the matrix $\tau(D_G)$

Isomap:
Classical Multi-dimensional Scaling

$$\mathbf{X'X} = -\frac{1}{2}\mathbf{JEJ}$$

E: Euclidian distance matrix

$$\mathbf{B} = -\frac{1}{2}\mathbf{JMJ}$$

M: Manifold distance matrix

$$\begin{aligned} L(\hat{\mathrm{X}}) &= \left\| -\frac{1}{2}\mathrm{J}\left(\mathrm{E}-\mathrm{M}\right)\mathrm{J} \right\| \\ &= \left\| \hat{\mathrm{X}}\hat{\mathrm{X}}' - \mathrm{B} \right\|. \end{aligned}$$

$$\mathrm{B} = \mathrm{Q}\Lambda\mathrm{Q}' \qquad \hat{\mathrm{X}} = \mathrm{Q}_+\Lambda_+^{\frac{1}{2}}$$

$$c_i = \sum_{a=1}^{m} x_{ia}^2$$

$$d_{ij}^2 = \sum_{a=1}^{m} \left(x_{ia} - x_{ja}\right)^2$$

$$\mathbf{E} = \mathbf{c1}' + \mathbf{1c}' - 2\mathbf{XX}'$$

$$\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{11}'$$

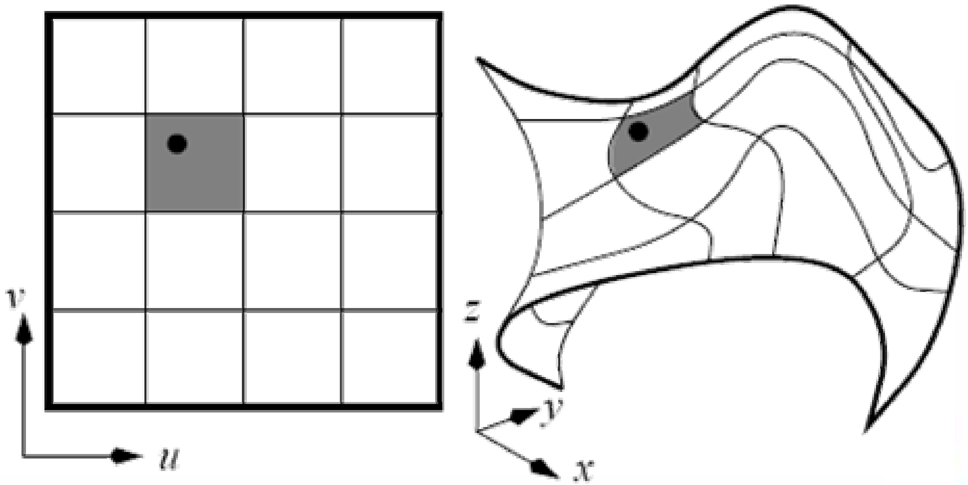
$$\begin{aligned} \mathrm{B} &= -\frac{1}{2}\mathrm{J}(\mathrm{c1}' + \mathrm{1c}' - 2\mathrm{XX}')\mathrm{J} \\ &= -\frac{1}{2}\mathrm{Jc0}' - \frac{1}{2}\mathrm{0c}'\mathrm{J} + \mathrm{JXX}'\mathrm{J} \\ &= \mathrm{XX}'. \end{aligned}$$

Eigen-structure analysis, SVD again

Isomap:
Classical Multi-dimensional Scaling (2D)

$$
\begin{aligned}
\mathbf{J} &= \text{eye}(n) - \text{ones}(n)./n; \\
\mathbf{B} &= -0.5 * \mathbf{J} * \mathbf{M} * \mathbf{J}; \\
& \ \text{\% Find largest eigenvalues+their eigenvectors:} \\
[\mathbf{Q}, \mathbf{L}] &= \text{eigs}(\mathbf{B}, 2, \text{'LM'}); \\
& \ \text{\% Extract the coordinates:} \\
\text{newy} &= \text{sqrt}(\mathbf{L}(1,1)). * \mathbf{Q}(:,1); \\
\text{newx} &= \text{sqrt}(\mathbf{L}(2,2)). * \mathbf{Q}(:,2);
\end{aligned}
$$

Isomap: application texture mapping



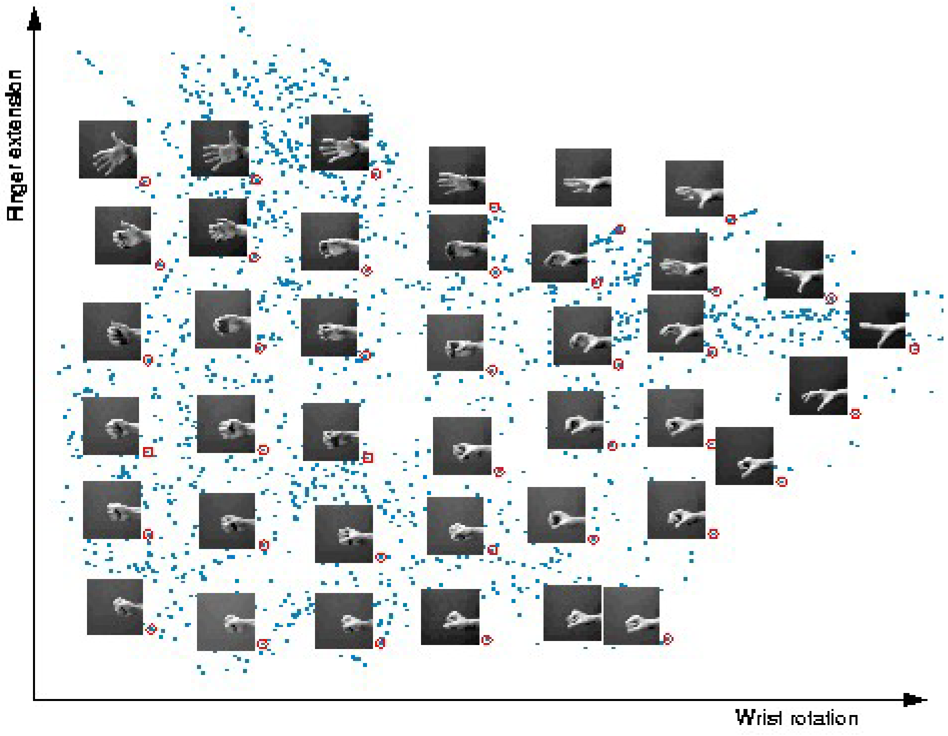

(a) (b)

Fig. 3. An example of a face flattening. (a) A 3D reconstruction of a face. (b) The flattened texture image of the face.


(a) (b)

Isomap: Examples

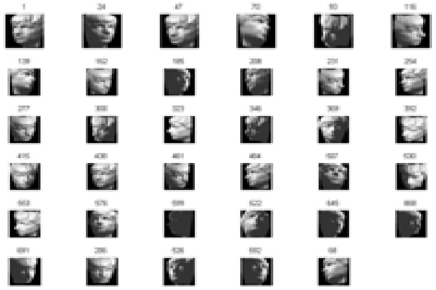
- *N*=2000 images 64x64 pixels K=6



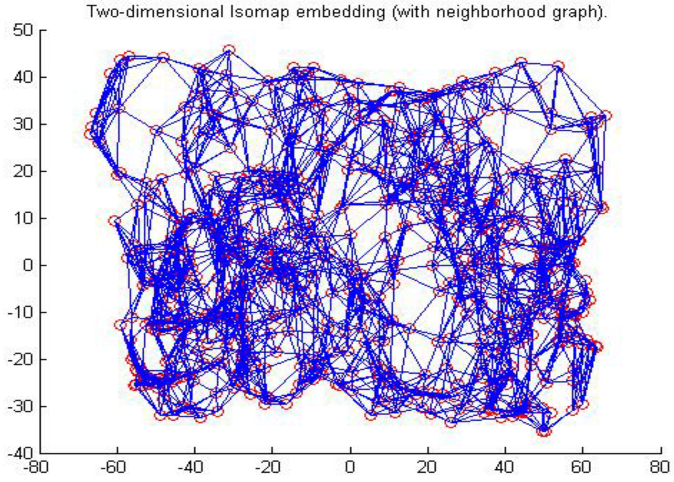
Isomap:
More Results

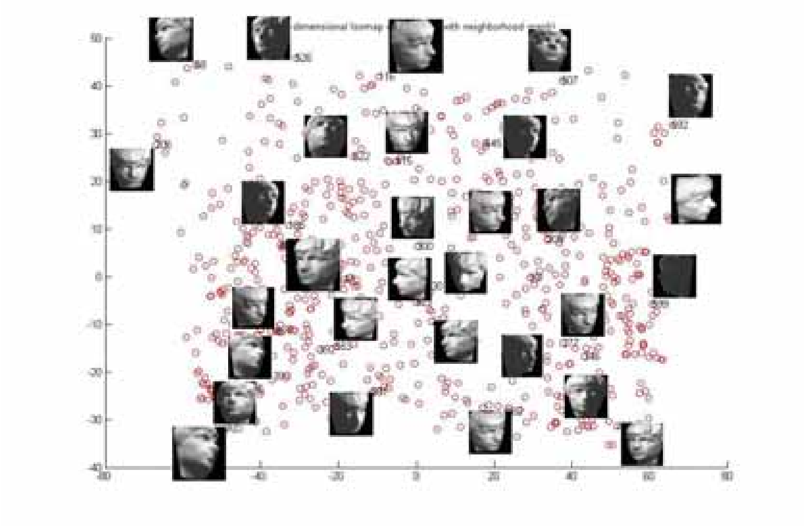
Input: 698
images of 64x64

K=7, d=2



Outputs:

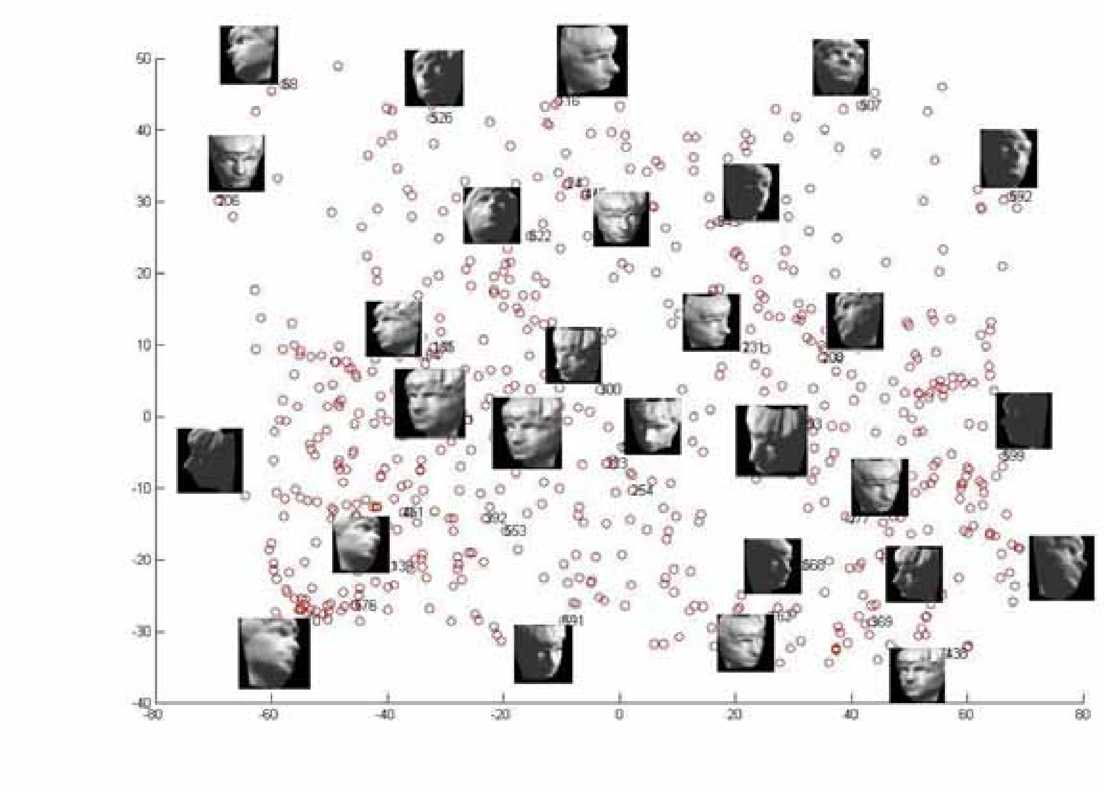




Isomap:
More Results

- Same inputs, but this time with d=3

698 images of 64x64 K=7



BoostMap:
A different perspective of embedding

- Goal – Significantly reduce retrieval time in image database systems.

- Embedding is formulated as a machine learning task.
- AdaBoost is used to combine many simple 1D embeddings into a d-dimensional embedding.
- Obtain ranking of all DB objects in order of similarity to a query object.

BoostMap: Problem Definition

- Embeddings are seen as classifiers.
- Estimate for *a, b, c* if *a* is closer to *b* or *c*.
- X – set of objects
- D_X – distance measure.

$$P_X(q, x_1, x_2) = \begin{cases} 1 \text{ if } \mathrm{D}_\mathrm{X}(q, x_1) < \mathrm{D}_\mathrm{X}(q, x_2) \\ 0 \text{ if } \mathrm{D}_\mathrm{X}(q, x_1) = \mathrm{D}_\mathrm{X}(q, x_2) \\ -1 \text{ if } \mathrm{D}_\mathrm{X}(q, x_1) > \mathrm{D}_\mathrm{X}(q, x_2) \end{cases}$$

- Find an Embedding F: X → R^d and a measure D_{R^d} that is used for evaluating any triplet.

$$\widetilde{F}(q, x_1, x_2) = D_{R^d}(F(q), F(x_2)) - D_{R^d}(F(q), F(x_1))$$

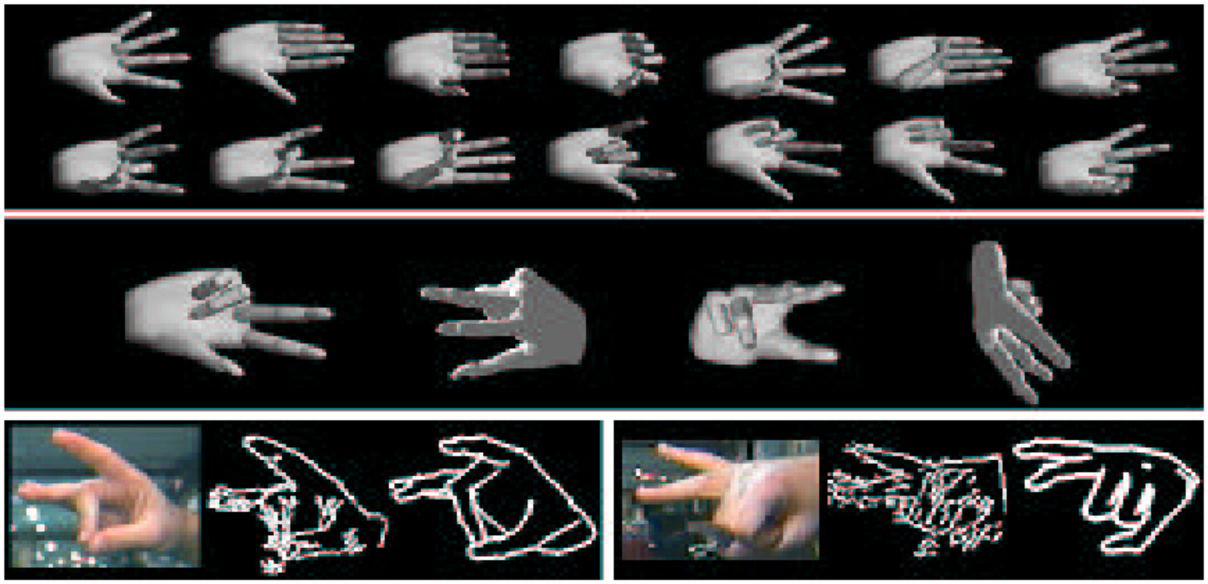
BoostMap - Outputs

- The output is a classifier: $H = \sum_{j=1}^{d} \alpha_j \tilde{F}_j$
- The final output is an embedding F : X → R^d
  And a distance measure D : R^d xR^d → R^d

$$F(x) = \left(F_1(x), \ldots, F_d(x)\right)$$

$$D_{R^d}\left((u_1, \ldots, u_d), (v_1, \ldots, v_d)\right) = \sum_{j=1}^{d} \left(\alpha_j \left|u_j - v_j\right|\right)$$

BoostMap - Results



Hand shapes used in the training set

Orientations used in the training set

Retrieval results

original

The query

Correct match

Summary:
Nonlinear Dimensionality Reduction

- Isomap - Use the geodesic manifold distances between all pairs.
  - sees more than just the Euclidean structure.
  - polynomial time procedure.

- LLE - Recovers global nonlinear structure from locally linear fits.
  - no need no estimate pair-wise distances.
  - optimization do not involve local minima.

- BoostMap - looks at embeddings as classifiers, uses AdaBoost.
  - main usage: similarity retrieval from database.
  - main advantage: trained offline, applicable online.

- Manifold learning …

Homework

- Algorithm implementations
    - Eigenface
    - ISOMAP

- Read the ISOMAP, LLE and BoostMap papers.

- Keep on thinking:
    - How to use dimension reduction results