Machine Learning

Dimensionality Reduction

slides thanks to Xiaoli Fern (CS534, Oregon State Univ., 2011)

Dimensionality reduction

- Many modern data domains involve huge numbers of features / dimensions
  - Documents: thousands of words, millions of bigrams
  - Images: thousands to millions of pixels
  - Genomics: thousands of genes, millions of DNA polymorphisms

Why reduce dimensions?

- High dimensionality has many costs
  - Redundant and irrelevant features degrade performance of some ML algorithms
  - Difficulty in interpretation and visualization
  - Computation may become infeasible
    - what if your algorithm scales as $O(\ n^3\ )$?
  - Curse of dimensionality

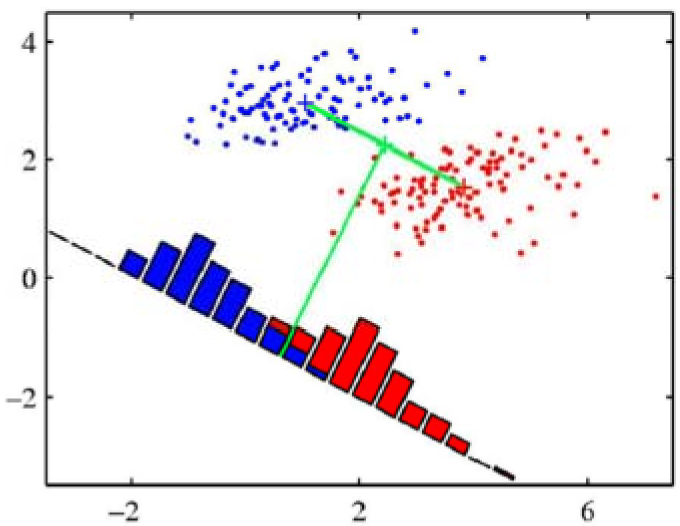
Extract Latent Linear Features

- Linearly project *n*-d data onto a *k*-d space
  - e.g., project space of 10^4 words into 3-dimensions
- There are infinitely many k-d subspaces that we can project the data into, which one should we choose
- This depends on the task at hand
  - If supervised learning, we would like to maximize the separation among classes: Linear discriminant analysis (LDA)
  - If unsupervised, we would like to retain as much data variance as possible: principal component analysis (PCA)

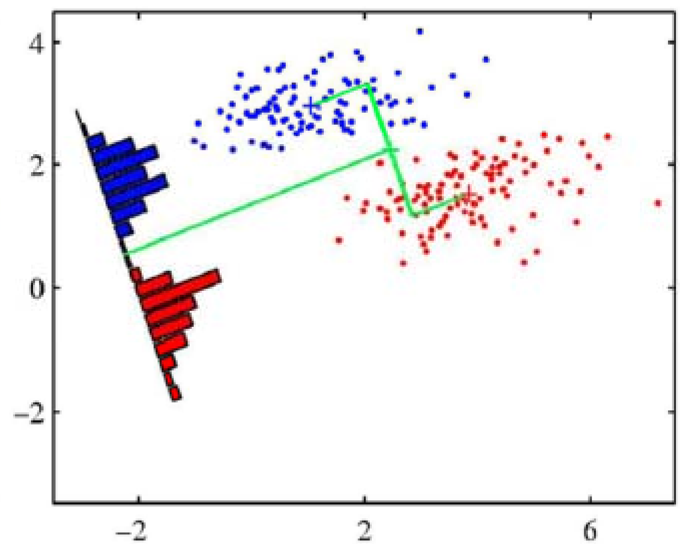
LDA for two classes

$$\mathbf{w} = S_w^{-1}(\mathbf{m_1} - \mathbf{m_2})$$

- Projecting data onto one dimension that maximizes the ratio of between-class scatter and total within-class scatter



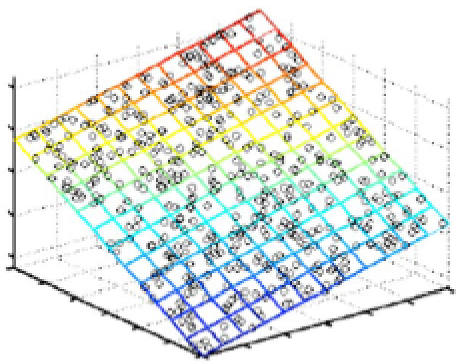
$w = m_1 - m_2$



$w = S_w^{-1}(m_1 - m_2)$

Unsupervised Dimension Reduction

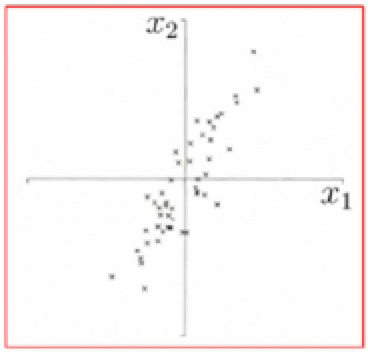
- Consider data without class labels
- Try to find a more compact representation of the data



$3d \Rightarrow 2d$

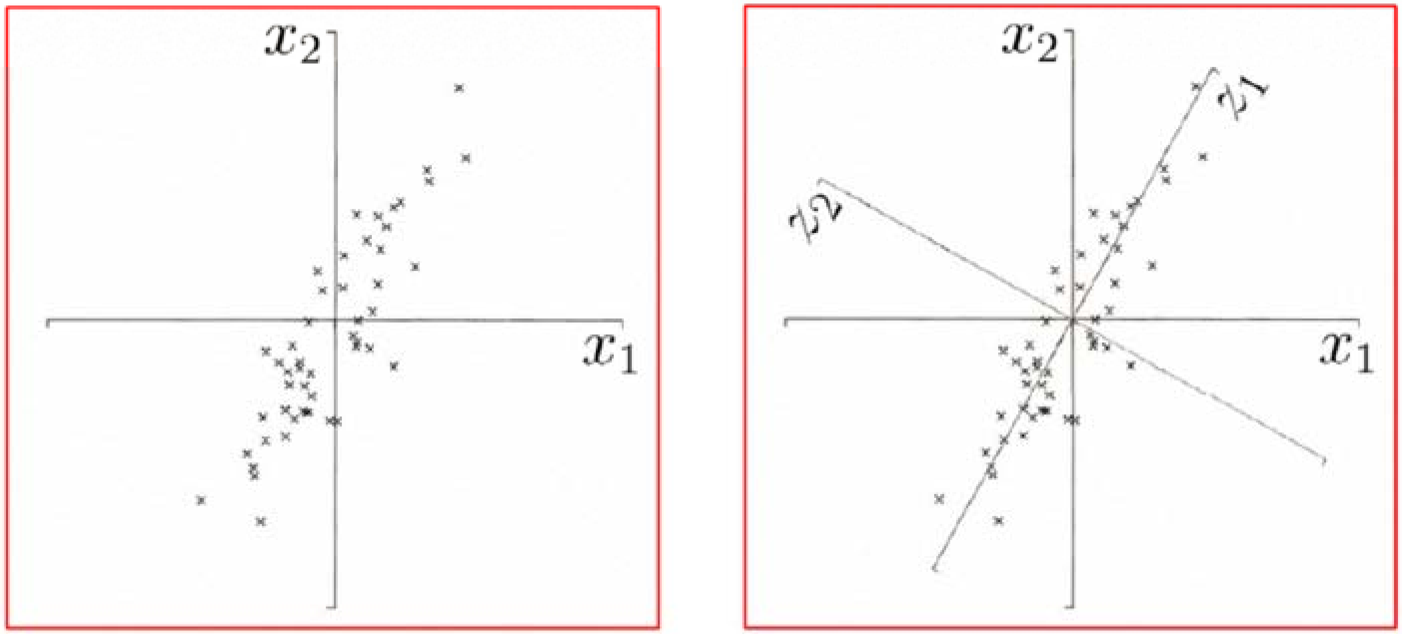
- Assume that the high dimensional data actually resides in a inherent low-dimensional space
- Additional dimensions are just random noise
- Goal is to recover these inherent dimensions and discard noise dimensions

Geometric picture of principal components (PCs)



Goal: to account for the variation in the data in as few dimensions as possible

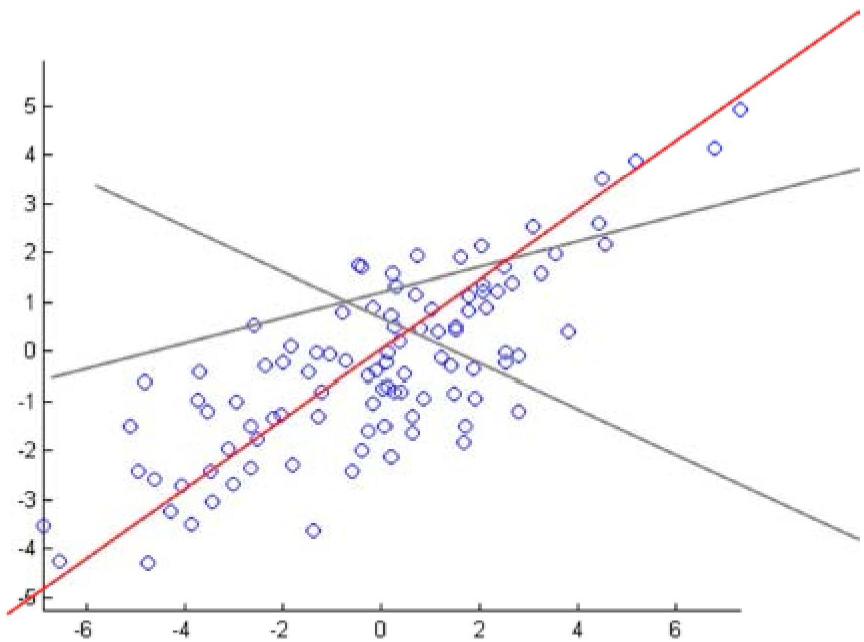
Geometric picture of principal components (PCs)



- The 1st PC is the projection direction that maximizes the variance of the projected data
- The 2nd PC is the projection direction that is orthogonal to the 1st PC and maximizes the variance

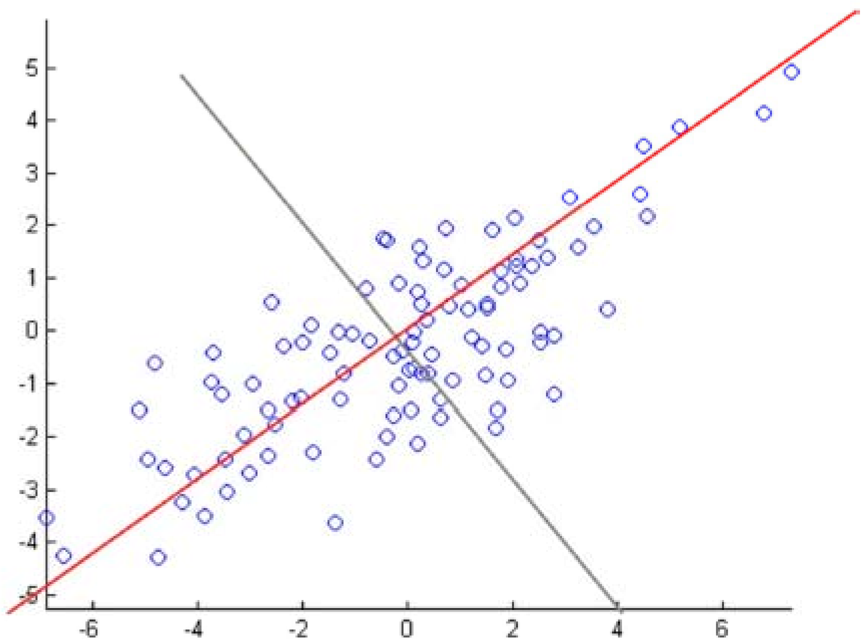
Conceptual Algorithm

- Find a line such that when the data is projected onto that line, it has the maximum variance



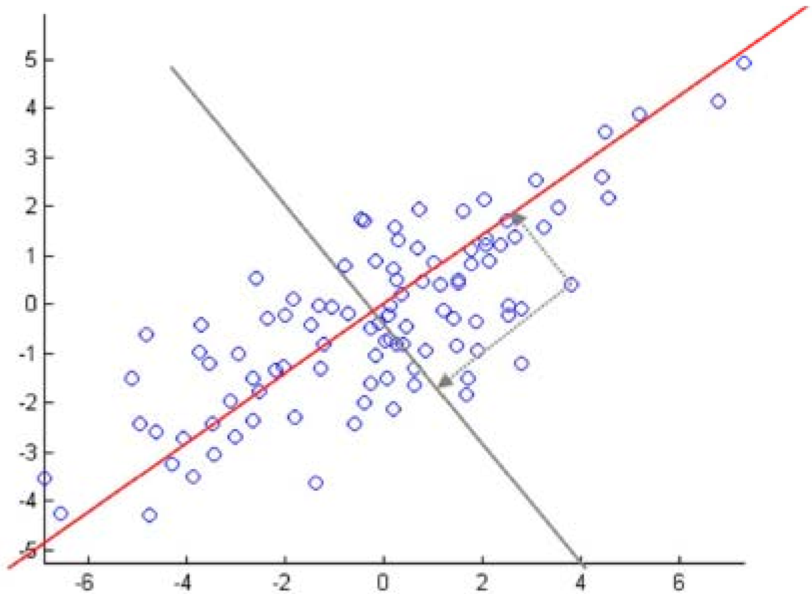
Conceptual Algorithm

- Find a new line, orthogonal to the first, that has maximum projected variance:



Repeat until *m* lines

- The projected position of a point on these lines gives the coordinates in the m-dimensional reduced space



Steps in principal component analysis

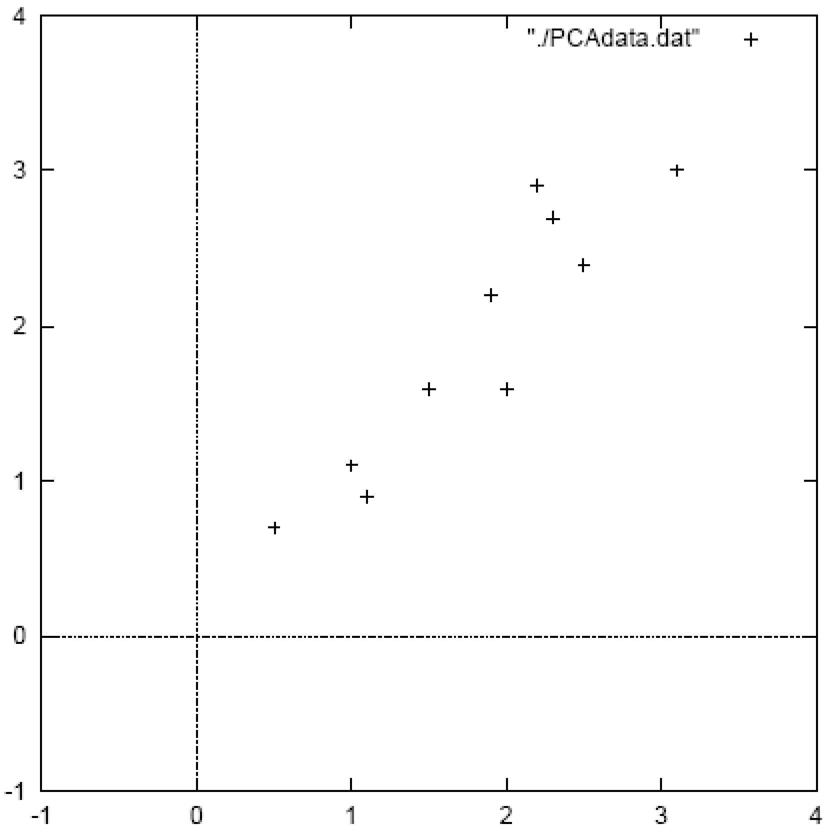
- Mean center the data
- Compute covariance matrix Σ
- Calculate eigenvalues and eigenvectors of Σ
  - Eigenvector with largest eigenvalue λ_1 is 1st principal component (PC)
  - Eigenvector with k^{th} largest eigenvalue λ_k is k^{th} PC
  - $\lambda_k / \Sigma_i \lambda_i$ = proportion of variance captured by k^{th} PC

Applying a principal component analysis

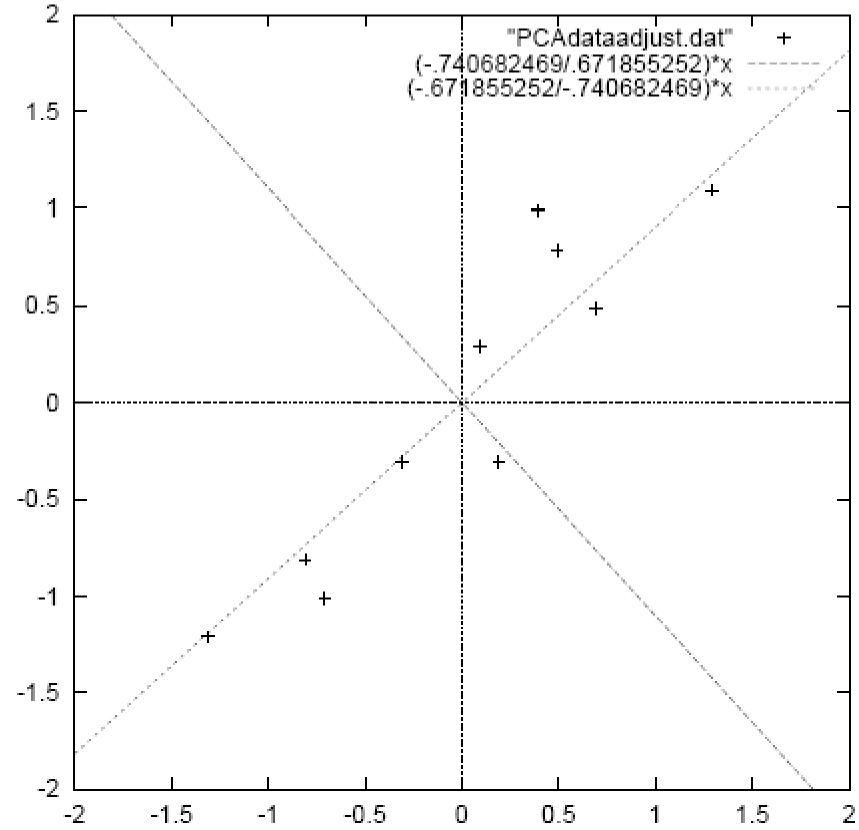
- Full set of PCs comprise a new orthogonal basis for feature space, whose axes are aligned with the maximum variances of original data.
- Projection of original data onto first *k* PCs gives a reduced dimensionality representation of the data.
- Transforming reduced dimensionality projection back into original space gives a reduced dimensionality *reconstruction* of the original data.
- Reconstruction will have some error, but it can be small and often is acceptable given the other benefits of dimensionality reduction.

PCA example

original data

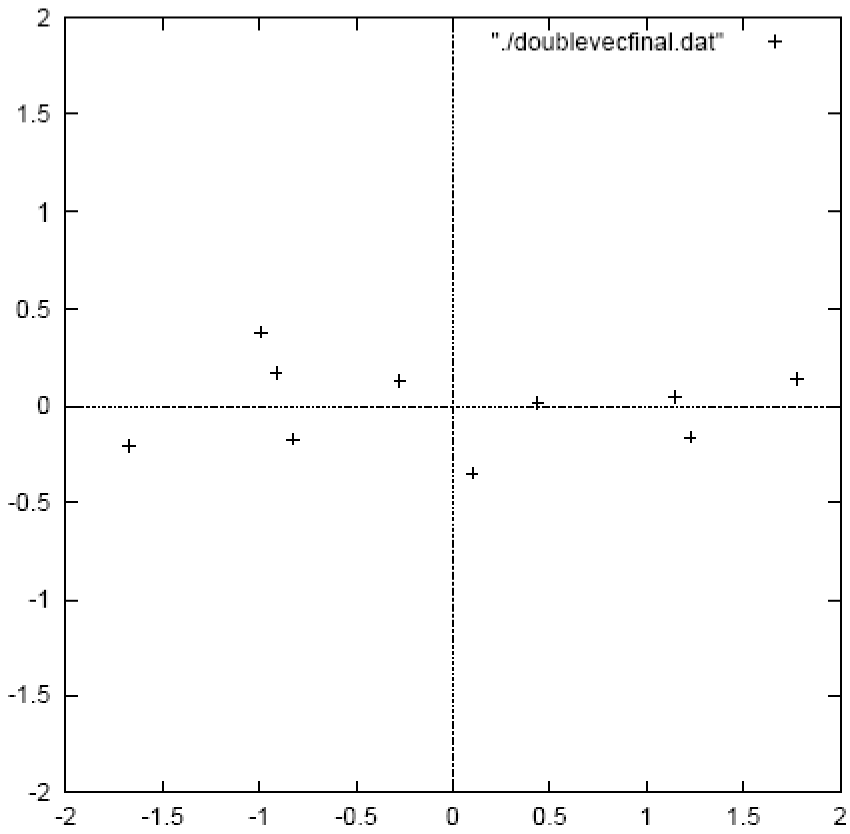


mean centered data with PCs overlayed

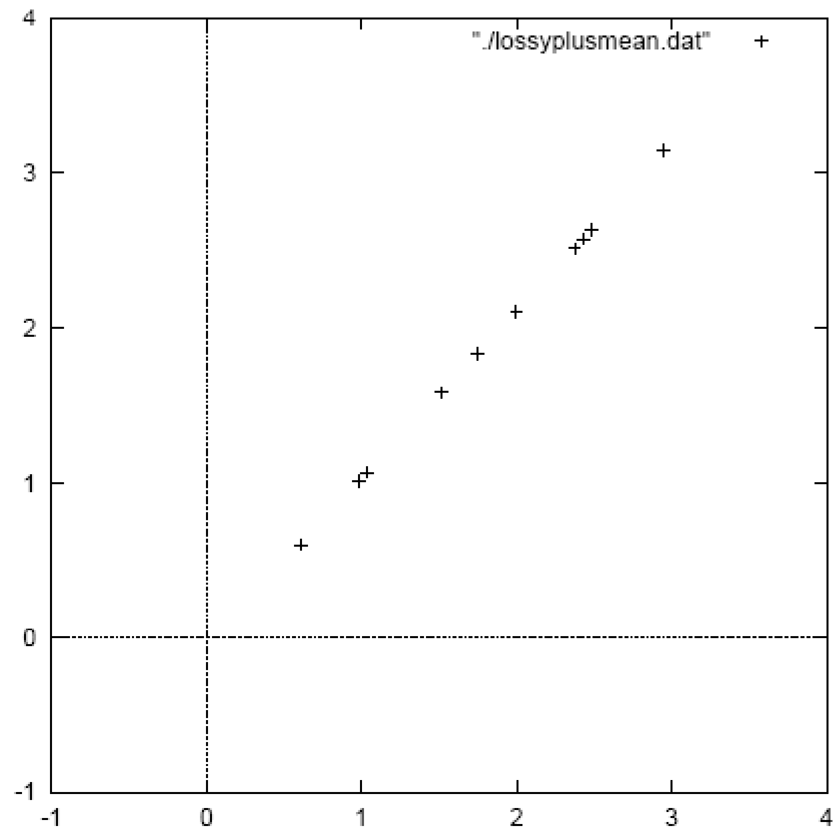


PCA example

original data projected Into full PC space

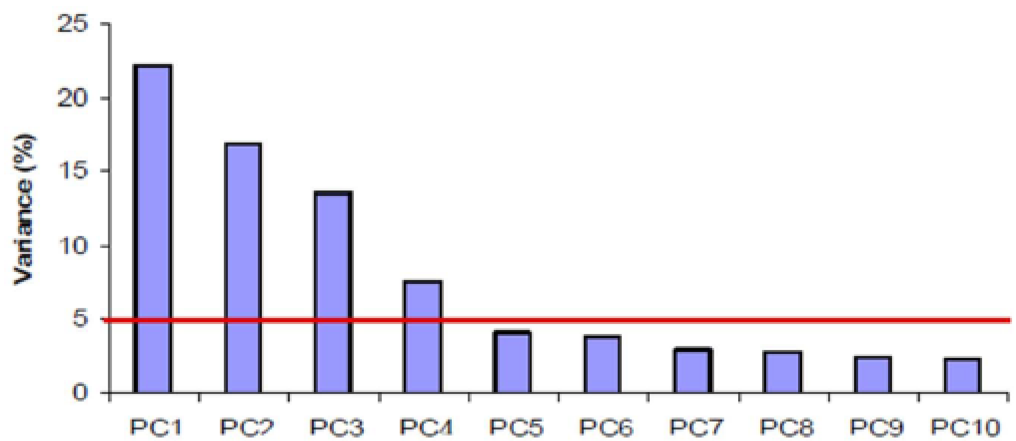


original data reconstructed using only a single PC



Dimension Reduction Using PCA

- Calculate the covariance matrix of the data S
- Calculate the eigen-vectors/eigen-values of S
- Rank the eigen-values in decreasing order
- Select eigen-vectors that retain a fixed percentage of the variance, (e.g., 80%, the smallest d such that $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_i \lambda_i} \geq 80\%$)



You might loose some info. But if the eigen-values are small, not much is lost.

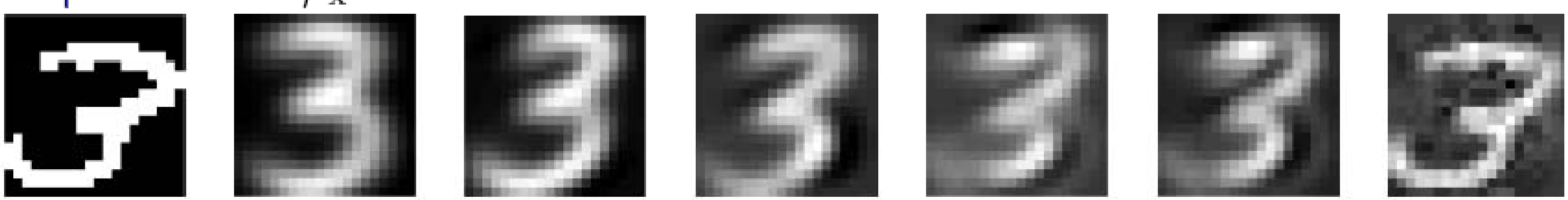
Choosing the dimension *k*

- The eigenvectors (columns of Φ) form a basis
- We can look at the expansion

$$\tilde{\mathbf{x}} \;=\; \mu_{\mathbf{x}} + \sum_{j=1}^{k} (\phi_j^T \mathbf{x}) \phi_j,$$

and examine the residual $\|\mathbf{x} - \tilde{\mathbf{x}}\|$

Input $\mathbf{x}$ | $\mu_{\mathbf{x}}$ | $k = 1$ | $k = 2$ | $k = 5$ | $k = 10$ | $k = 100$

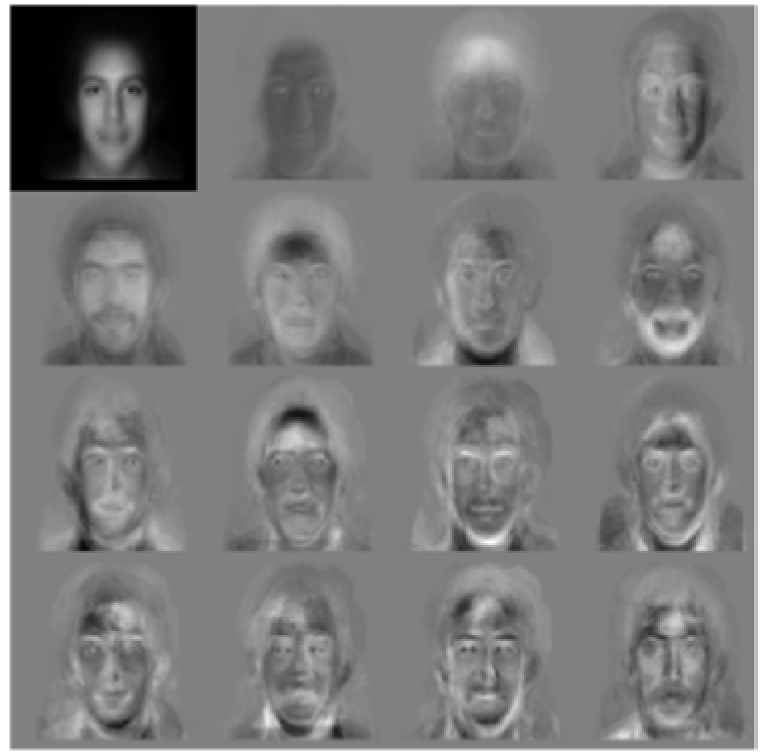


Example: Face Recognition

- An typical image of size 256 x 128 is described by n = 256x128 = 32768 dimensions
- Each face image lies somewhere in this high-dimensional space
- Images of faces are generally similar in overall configuration, thus
  - They cannot be randomly distributed in this space
  - We should be able to describe them in a much low-dimensional space

PCA for Face Images: Eigenfaces

- Database of 128 carefully-aligned faces.
- Here are the mean and the first 15 eigenvectors.
- Each eigenvector can be shown as an image
- These images are face-like, thus called eigenface

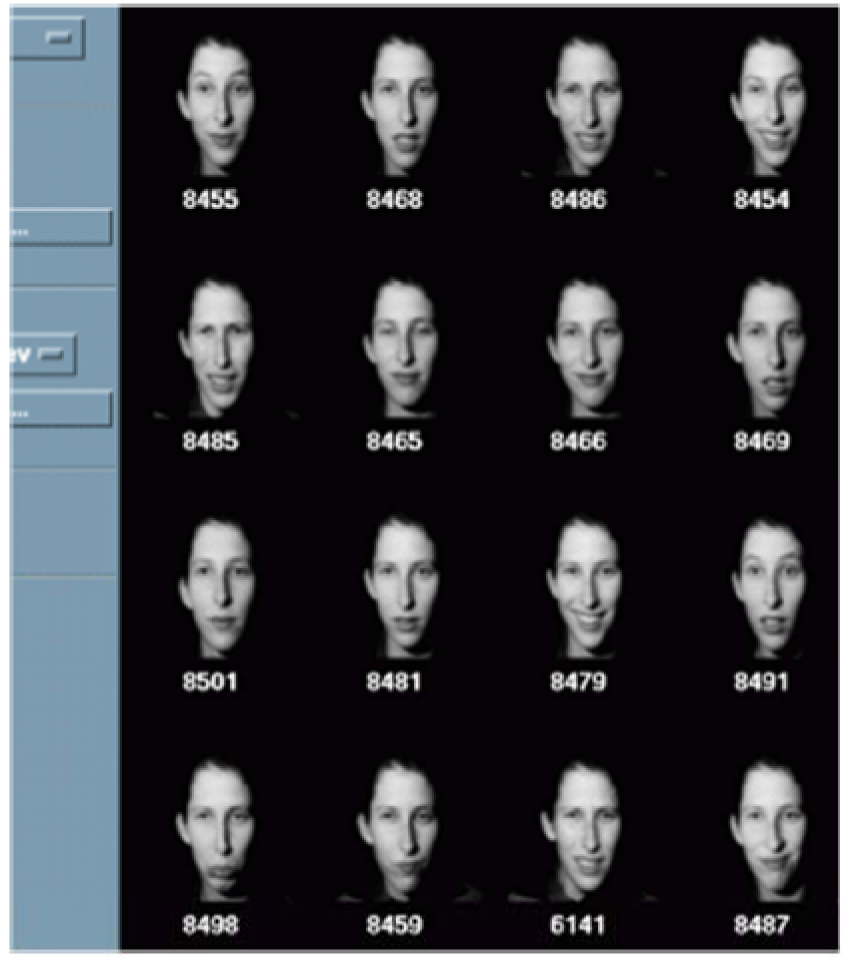


Face Recognition in Eigenface space
(Turk and Pentland 1991)

- Nearest Neighbor classifier in the eigenface space
- Training set always contains 16 face images of 16 people, all taken under the same conditions of lighting, head orientation, and image size
- Accuracy:
  - variation in lighting: 96%
  - variation in orientation: 85%
  - variation in image size: 64%

Face Image Retrieval

- Left-top image is the query image
- Return 15 nearest neighbor in the eigenface space
- Able to find the same person despite
  - different expressions
  - variations such as glasses



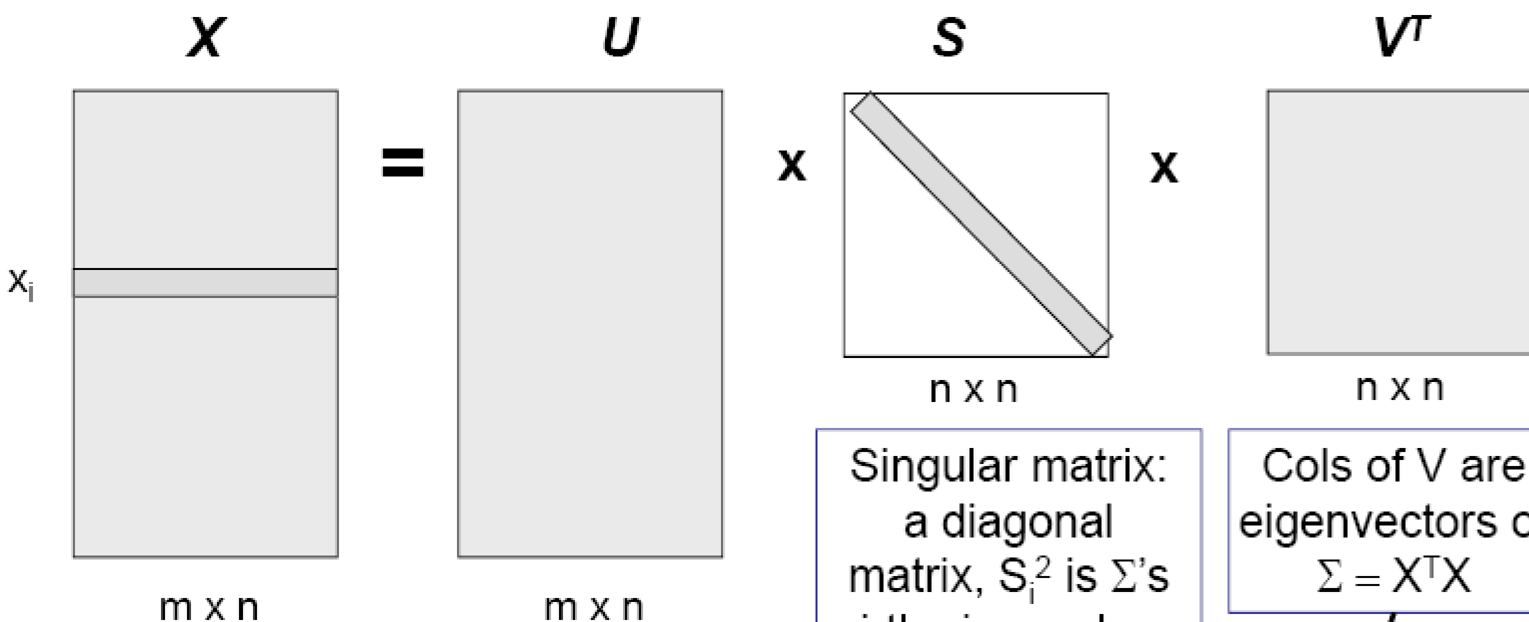
PCA: a useful preprocessing step

- Helps reduce computational complexity.
- Can help supervised learning.
  - Reduced dimension $\Rightarrow$ simpler hypothesis space.
  - Smaller VC dimension $\Rightarrow$ less risk of overfitting.
- PCA can also be seen as noise reduction.
- Caveats:
  - Fails when data consists of multiple separate clusters.
  - Directions of greatest variance may not be most informative (i.e. greatest classification power).

Practical Issue: Scaling Up

- Covariance of the image data is BIG!
  - size of Σ = 32768 x 32768
  - finding eigenvector of such a matrix is slow.
- SVD comes to rescue!
  - Can be used to compute principal components
  - Efficient implementations available, e.g., Matlab svd

Singular Value Decomposition: $X=USV^T$



Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue

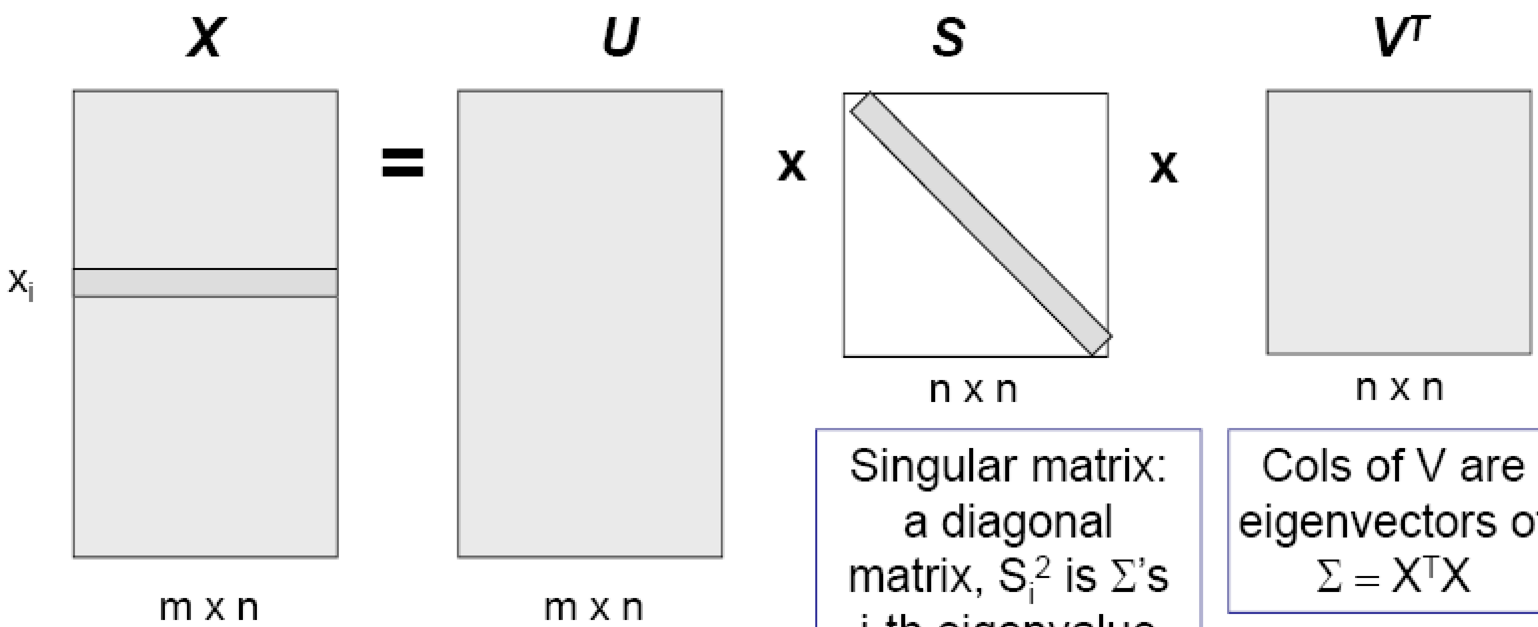
Cols of V are eigenvectors of $\Sigma = X^T X$

X: our m x n data matrix, one row per data point

Each row of US gives coordinates of a data point in the projected space

$$\boxed{X^T X v_1 = VSU^T USV^T v_1 = s_1^2 v_1}$$

Singular Value Decomposition: $X=USV^T$



Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue

Cols of V are eigenvectors of $\Sigma = X^T X$

X: our m x n data matrix, one row per data point

Each row of US gives coordinates of a data point in the projected space

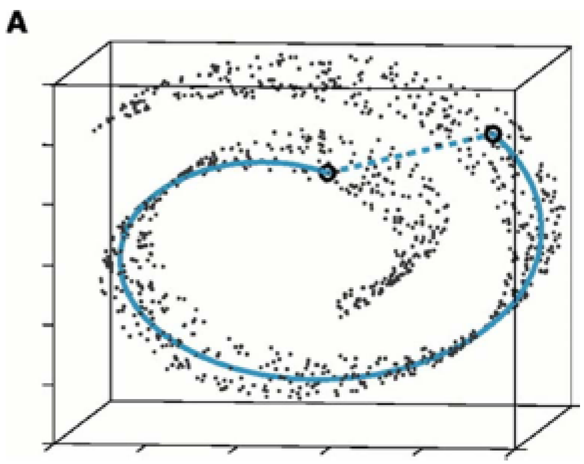
If X is centered, then cols of V are the principal components

SVD for PCA

- Create centered data matrix X
- Solve SVD: $X = USV^T$
- Columns of V are the eigenvectors of Σ sorted from largest to smallest eigenvalues – select the first *k* columns as our principal components

Nonlinear Methods

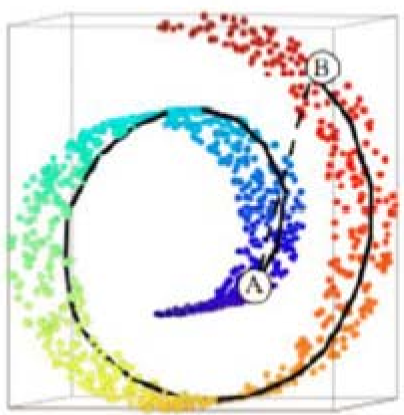
- Data often lies on or near a nonlinear low-dimensional curve
- We call such low dimension structure manifolds



Swiss roll data

ISOMAP: Isometric Feature Mapping
(Tenenbaum et al. 2000)

- A nonlinear method for dimensionality reduction
- Preserves the global, nonlinear geometry of the data by preserving the geodesic distances
- Geodesic: originally geodesic means the shortest route between two points on the surface of the manifold



ISOMAP

- Two steps
  1. Approximate the geodesic distance between every pair of points in the data
     - The manifold is locally linear
     - Euclidean distance works well for points that are close enough
     - For the points that are far apart, their geodesic distance can be approximated by summing up local Euclidean distances

  2. Find a Euclidean mapping of the data that preserves the geodesic distance

Geodesic Distance

- Construct a graph by
  - Connecting i and j if
    - $d(i, j) < \varepsilon$ (ε-isomap) or
    - i is one of j's k nearest neighbors (k-isomap)
  - Set the edge weight equal $d(i, j)$ – Euclidean distance
- Compute the Geodesic distance between any two points as the ***shortest path distance***

Compute the Low-Dimensional Mapping

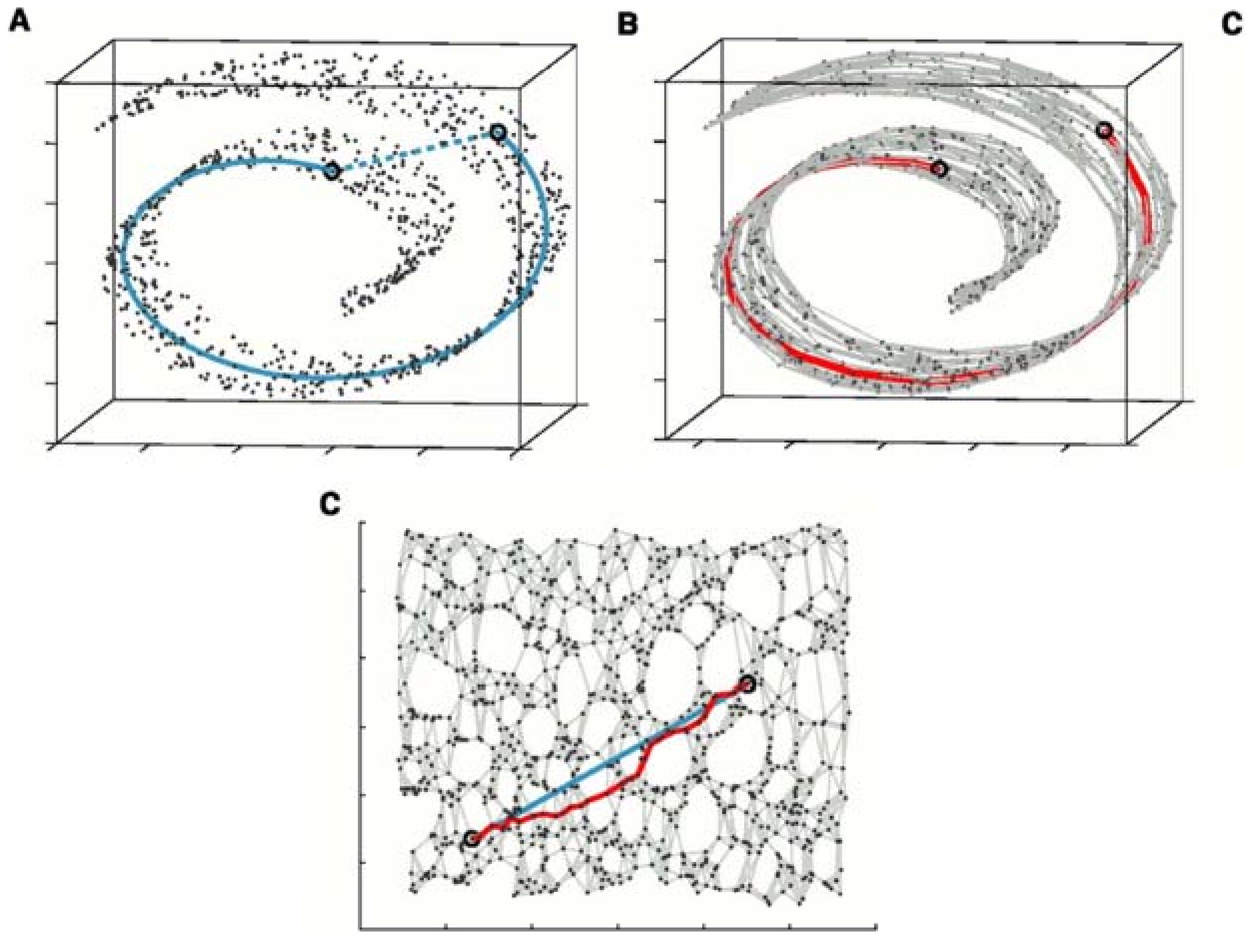
- We can use Multi-Dimensional scaling (MDS), a class of statistical techniques that

Given:

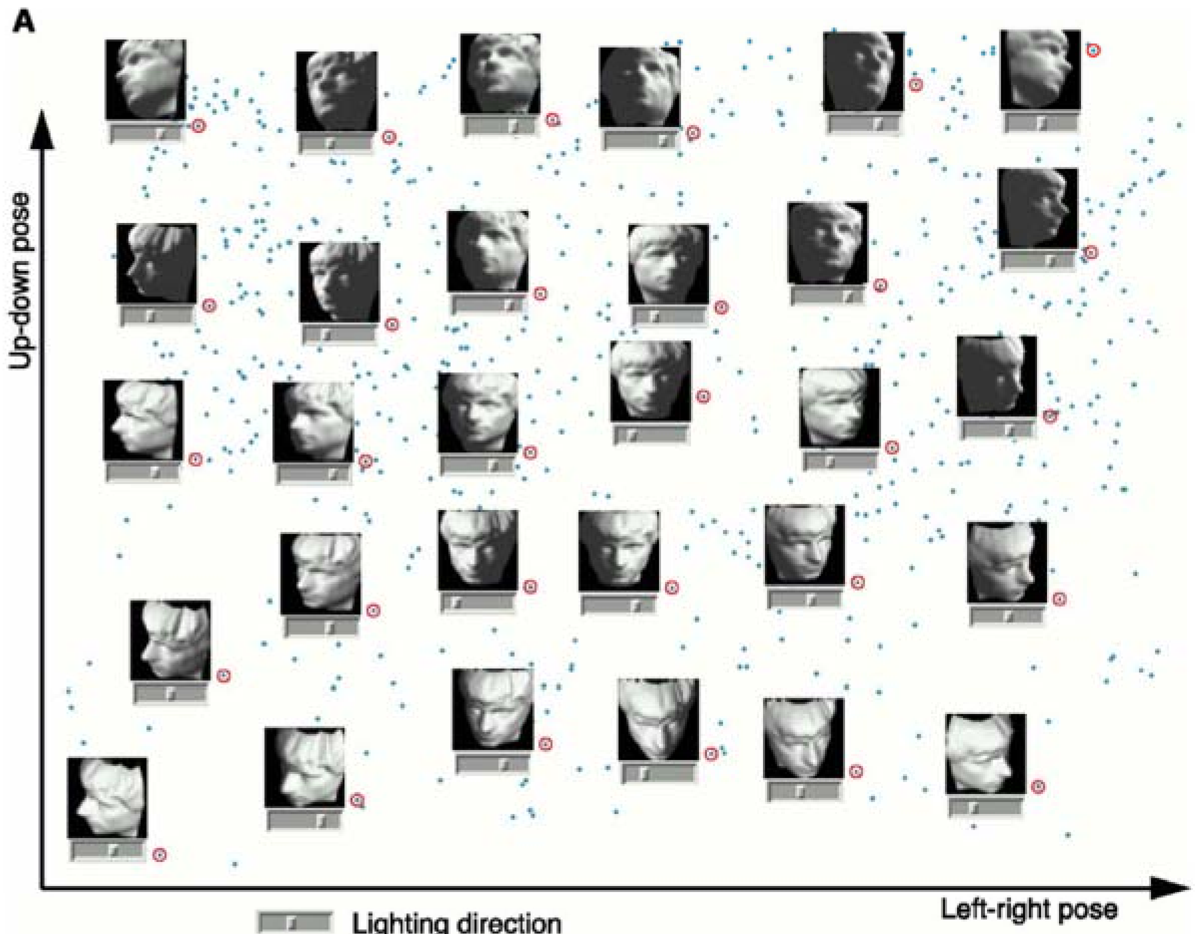
n x *n* matrix of dissimilarities between *n* objects

Outputs: a coordinate configuration of the data in a low-dimensional space R^d whose Euclidean distances closely match given dissimilarities.

ISOMAP on Swiss Roll Data

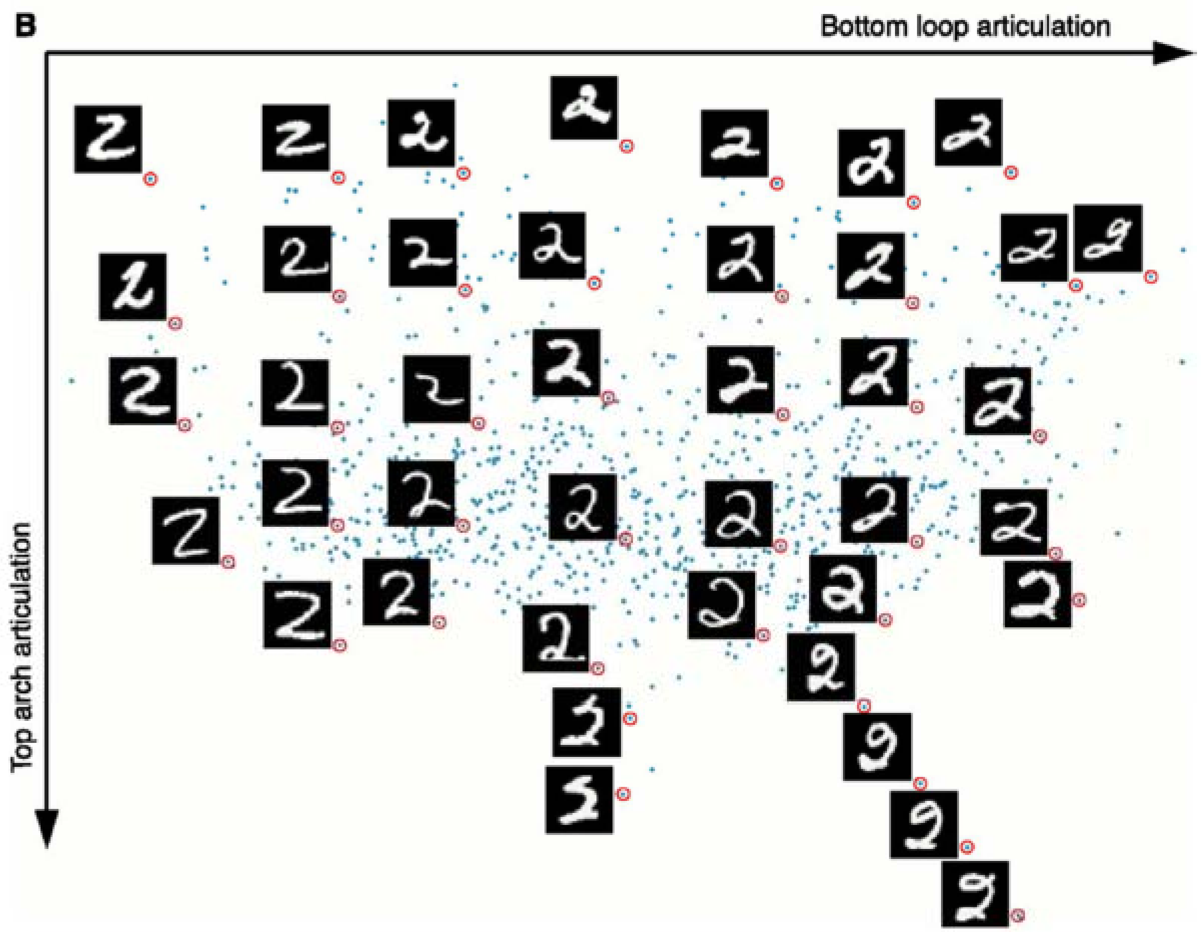


ISOMAP Examples



4096 -> 2d

ISOMAP Examples



Off-the-shelf classifiers

Per Tom Dietterich:

“Methods that can be applied directly to data without requiring a great deal of time-consuming data preprocessing or careful tuning of the learning procedure.”

Off-the-shelf criteria

Criterion	LMS	Logistic	LDA	Trees	Nets	NNbr	SVM	NB	Boosted Trees
Mixed data	no	no	no	yes	no	no	no	yes	yes
Missing values	no	no	yes	yes	no	some	no	yes	yes
Outliers	no	yes	no	yes	yes	yes	yes	disc	yes
Monotone transforms	no	no	no	yes	some	no	no	disc	yes
Scalability	yes	yes	yes	yes	yes	no	no	yes	yes
Irrelevant inputs	no	no	no	some	no	no	some	some	yes
Linear combinations	yes	yes	yes	no	yes	some	yes	yes	some
Interpretable	yes	yes	yes	yes	no	no	some	yes	no
Accurate	yes	yes	yes	no	yes	no	yes	yes	yes

slide thanks to Tom Dietterich (CS534, Oregon State Univ., 2005)

Practical advice on machine learning

from Andrew Ng at Stanford

slides:

http://cs229.stanford.edu/materials/ML-advice.pdf

video:

http://www.youtube.com/v/sQ8T9b-uGVE

(starting at 24:56)