Algorithms in Nature

Dimensionality Reduction

Slides adapted from Tom Mitchell and Aarti Singh

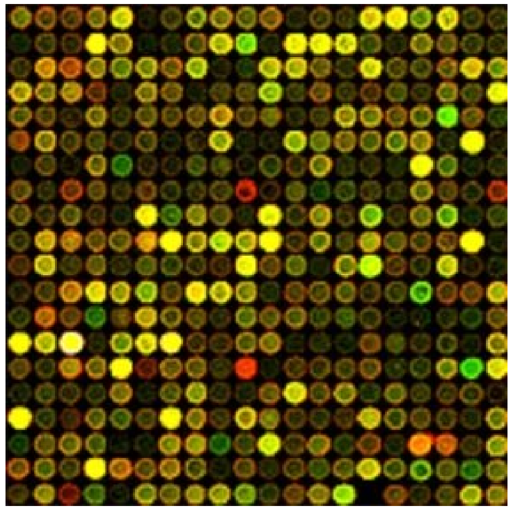
High-dimensional data

(i.e. lots of features)

<u>Document classification</u>:
Billions of documents x Thousands/Millions of words/bigrams matrix

<u>Recommendation systems</u>:
480,189 users x 17,770 movies matrix

<u>Clustering gene expression profiles</u>:
10,000 genes x 1,000 conditions



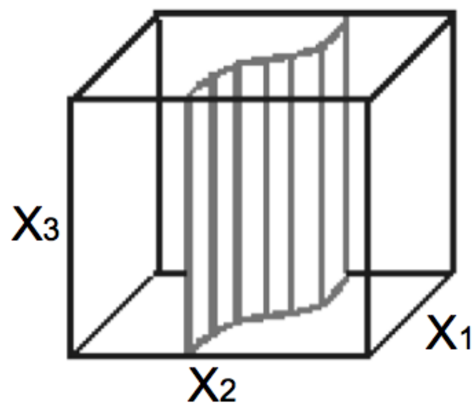
Curse of dimensionality

Why might many features be bad?

- Harder to interpret and visualize
  - provides little intuition of the underlying structure of the data
- Harder to store data and learn complex models
  - statistically and computationally challenging to classify
  - dealing with redundant features and noise
- Possibly worse generalization

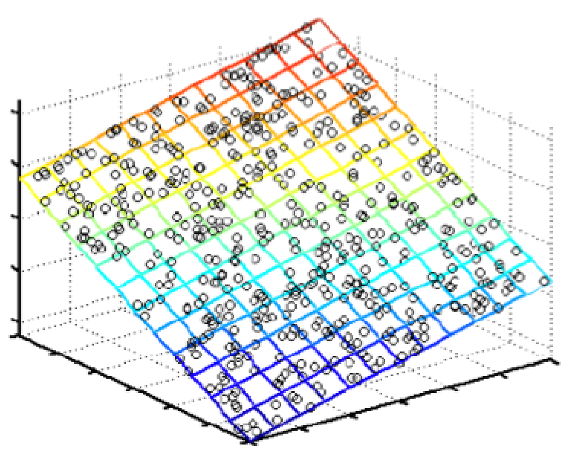
Two types of dimensionality reductions

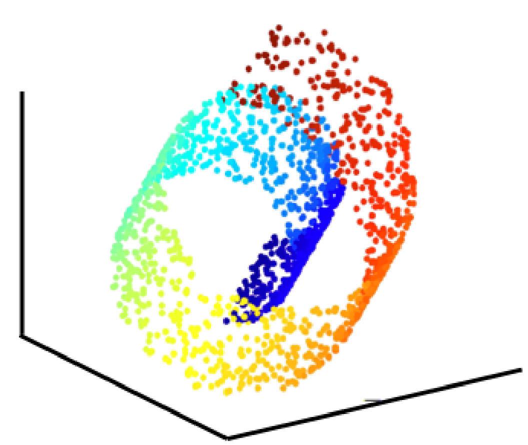
<u>Feature selection</u>: only a few features are relevant to the task



X_3 - Irrelevant

<u>Latent features</u>: a (linear) combination of features provides a more efficient representation than the observed features (e.g. PCA)





For example, topics (sports, politics, economics) instead of individual documents

Facial recognition

Say we wanted to build a human facial recognition system.

Option 1: enumerate all 6 billion faces, update as necessary.

Option 2: learn a low-dimensional basis that can be used to represent *any* face (PCA: Today)

Option 3: learn the basis using insights from how the brain does it (NMF: Wednesday)

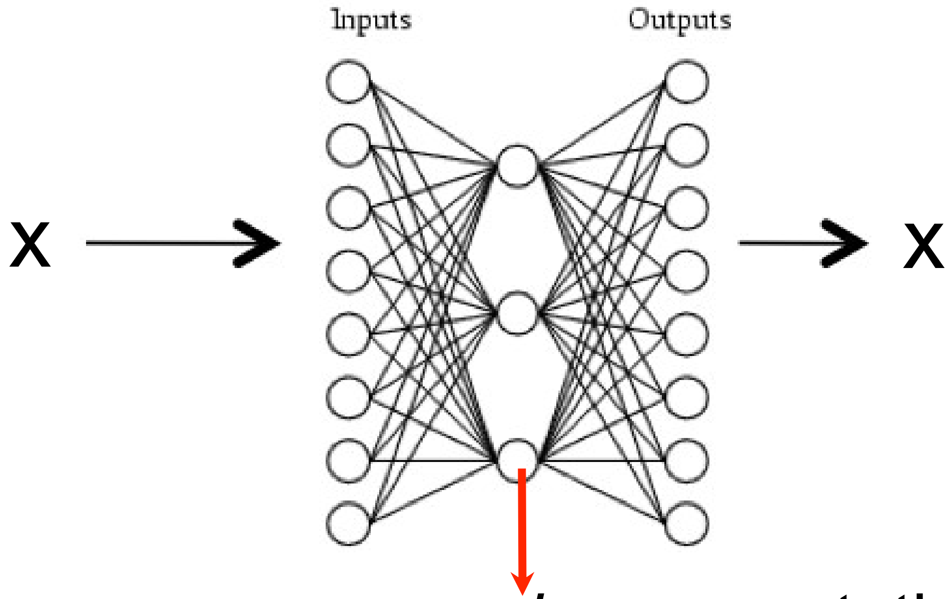
.....

(high-dimensionality space of possible human faces)

Principal Component Analysis

A dimensionality reduction technique similar to auto-encoding neural networks:

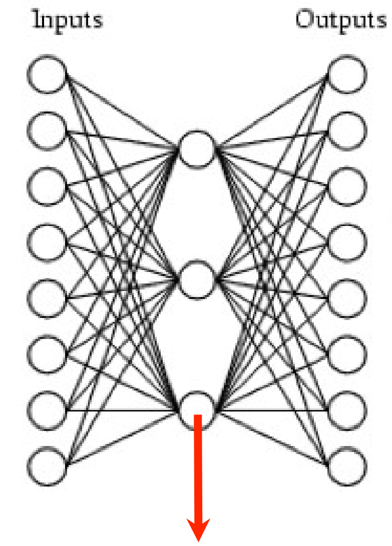
Learn a *linear* representation of the input data that can best reconstruct it



Hidden layer: a *compressed* representation of the input data. Think of compression as a form of pattern recognition.

Principal Components Analysis

$$\text{face}_i = \sum_k c_{ik} \ \text{eigenface}_k$$

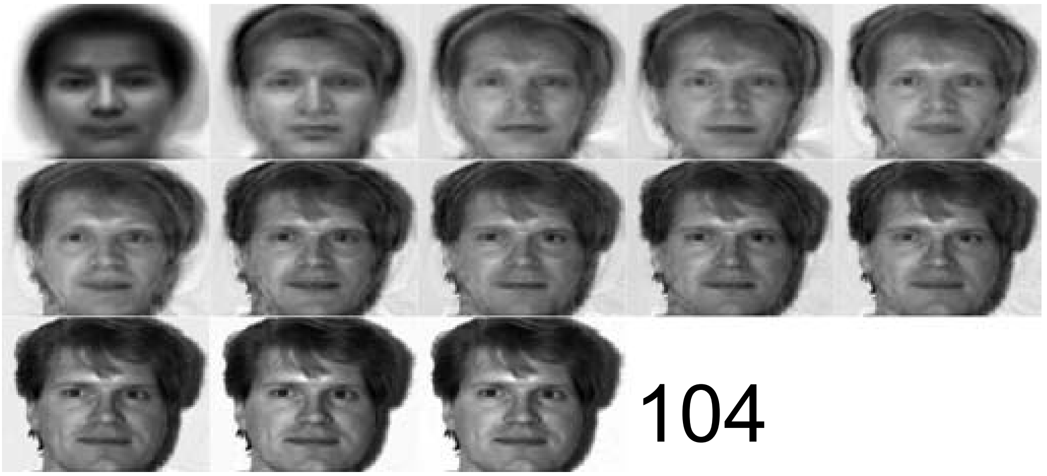
face →  → face

"eigenfaces"

Face reconstruction using PCA

Reconstruction using the first 25 components (eigenfaces), one at a time

Same, but adding 8 PCA components at each step



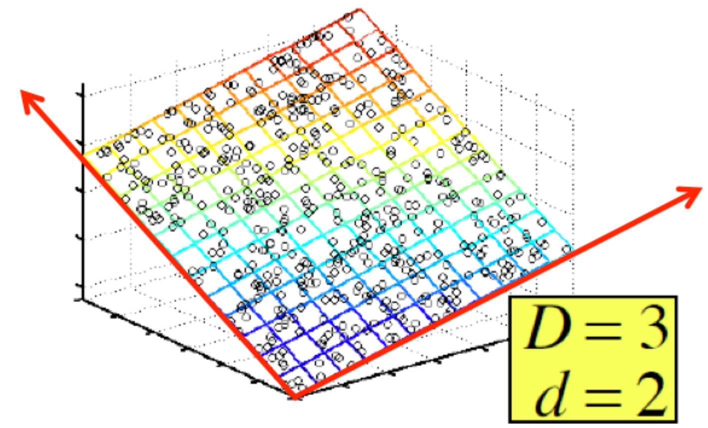
In general: top k dimensions are the k-dimensional representation that minimizes reconstruction (sum of squared) error.

Principal Component Analysis

Given data points in d-dimensional space, project them onto a lower dimensional space while preserving as much information as possible.

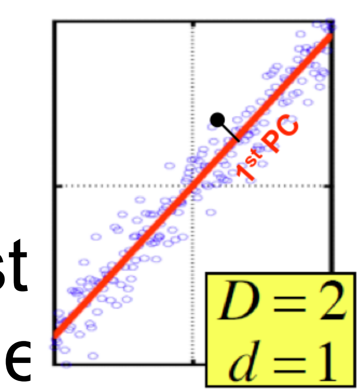
- e.g. find best planar approx to 3D data

- e.g. find best planar approx to 10^4D data



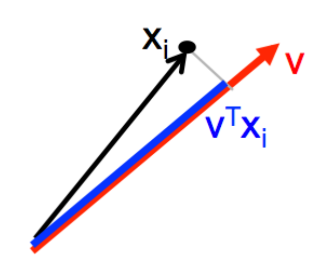
Principal components are orthogonal directions that capture variance in the data:

1st PC: direction of greatest variability in the data
2nd PC: next orthogonal (uncorrelated) direction of greatest variability: remove variability in the first direction, the find the next direction of greatest variability.



Etc.

Projection of data point x_i (a d-dim vector) onto 1st PC v is $v^T x_i$



PCA: find projections to minimize reconstruction error

Assume data is a set of d-dimensional vectors, where n^{th} vector is:

$$x^n = < x_1^n, \cdots, x_d^n >$$

We can represent these in terms of any d orthogonal vectors u_1, ..., u_d:

$$x^n = \sum_{i=1}^{d} z_i^n \mathbf{u_i}$$

<u>Goal</u>: given $M<d$, find u_1, ..., u_M that minimizes: $E_M = \sum_{i=1}^{N} ||x^n - \hat{x}^n||^2$

where $\hat{x}^n = \bar{x} + \sum_{i=1}^{M} z_i^n \mathbf{u}_i$

original data point

reconstructed

origin is mean-centered

coefficient/weight of projection

PCA

Idea: zero reconstruction error if M=d, so all error is due to missing components.

Therefore: $$E_M = \sum_{i=M+1}^{d} \sum_{n=1}^{N} [\mathbf{u}_i^T (x^n - \bar{x})]^2$$

Project difference between the original point and the mean onto the basis vector, take the square

$$= \sum_{i=M+1}^{d} \sum_{n=1}^{N} [\mathbf{u}_i^T (x^n - \bar{x})][\mathbf{u}_i^T (x^n - \bar{x})]$$

Expand and re-arrange

$$= \sum_{i=M+1}^{d} \sum_{n=1}^{N} [\mathbf{u}_i^T (x^n - \bar{x})][(x^n - \bar{x})^T \mathbf{u}_i]$$

$$= \sum_{i=M+1}^{d} \mathbf{u_i}^T \Sigma \mathbf{u_i}$$

Substitute co-variance matrix

Co-variance matrix $$\Sigma_{ij} = \sum_{n=1}^{N} (x_i^n - \bar{x}_i)(x_j^n - \bar{x}_j)^T$$

Measures correlation or inter-dependence between two dimensions

PCA contd.

$$E_M = \sum_{i=M+1}^{d} \mathbf{u_i}^{\mathrm{T}} \boxed{\Sigma \mathbf{u_i}}$$

<u>Review</u>: matrix A has eigenvector *u* with eigenvalue λ if: $Au = \lambda u$

$$\rightarrow \boxed{\Sigma \mathbf{u_i}} = \lambda_i \mathbf{u_i}$$

eigenvalue (scalar)

eigenvector of covariance matrix

$$E_M = \sum_{i=M+1}^{d} \lambda_i$$

The reconstruction error can be exactly computed from the eigenvalues of the covariance matrix

PCA Algorithm

1. $X \leftarrow$ Create Nxd data matrix with one row vector x^n per data point.
2. $X \leftarrow$ subtract mean from each vector x^n in X
3. $\Sigma \leftarrow$ compute covariance matrix of X
4. Find eigenvectors and eigenvalues of Σ
5. PCs ← the M eigenvectors with the largest eigenvalues

Original representation:

$$x^n = < x_1^n, \cdots, x_d^n >$$

Transformed representation:

$$\hat{x}^n = < \mathbf{u_1}^{\mathbf{T}} x^n, \ldots, \mathbf{u_M}^{\mathbf{T}} x^n >$$

PCA example

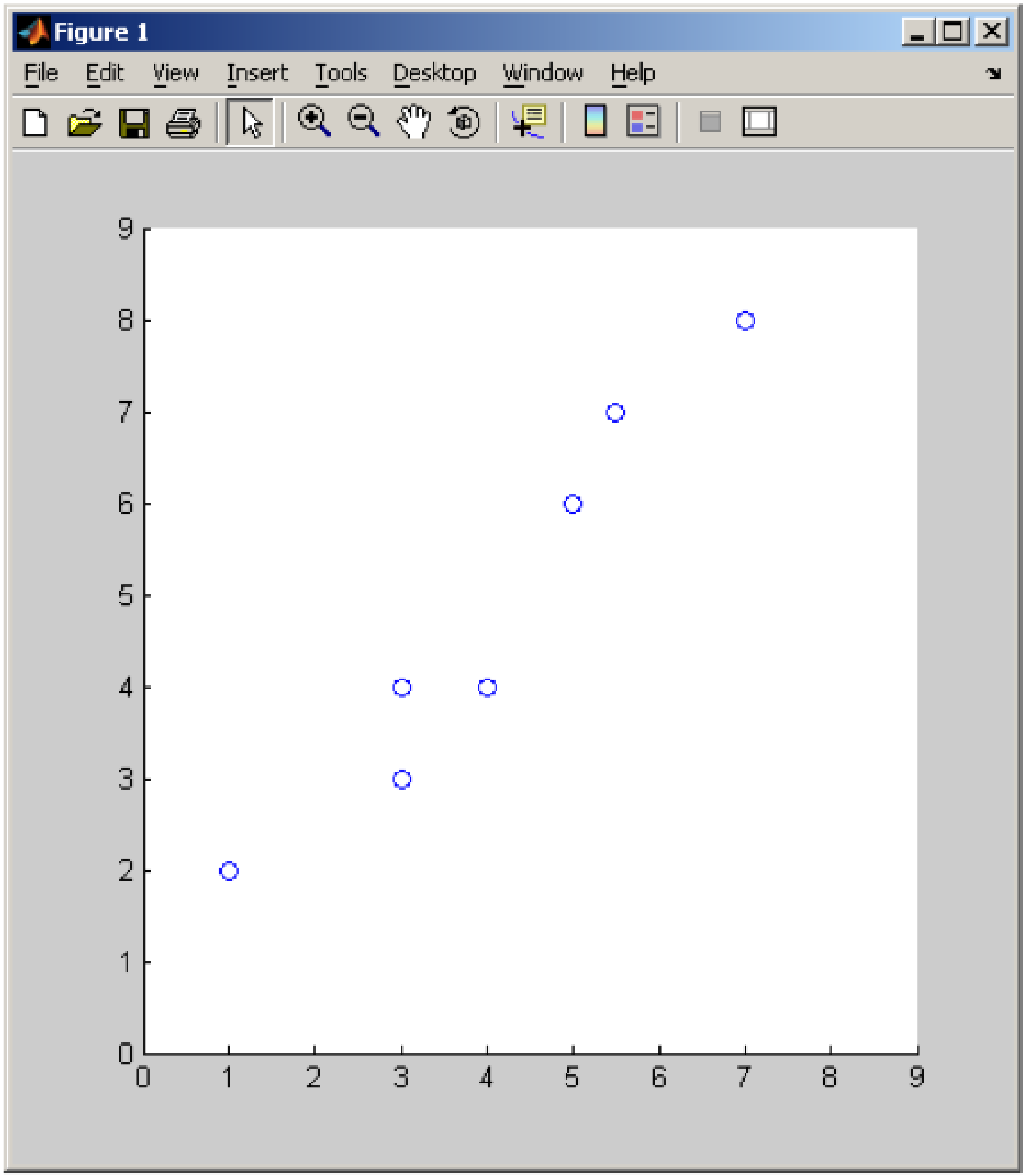


Figure 1
File Edit View Insert Tools Desktop Window Help
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9

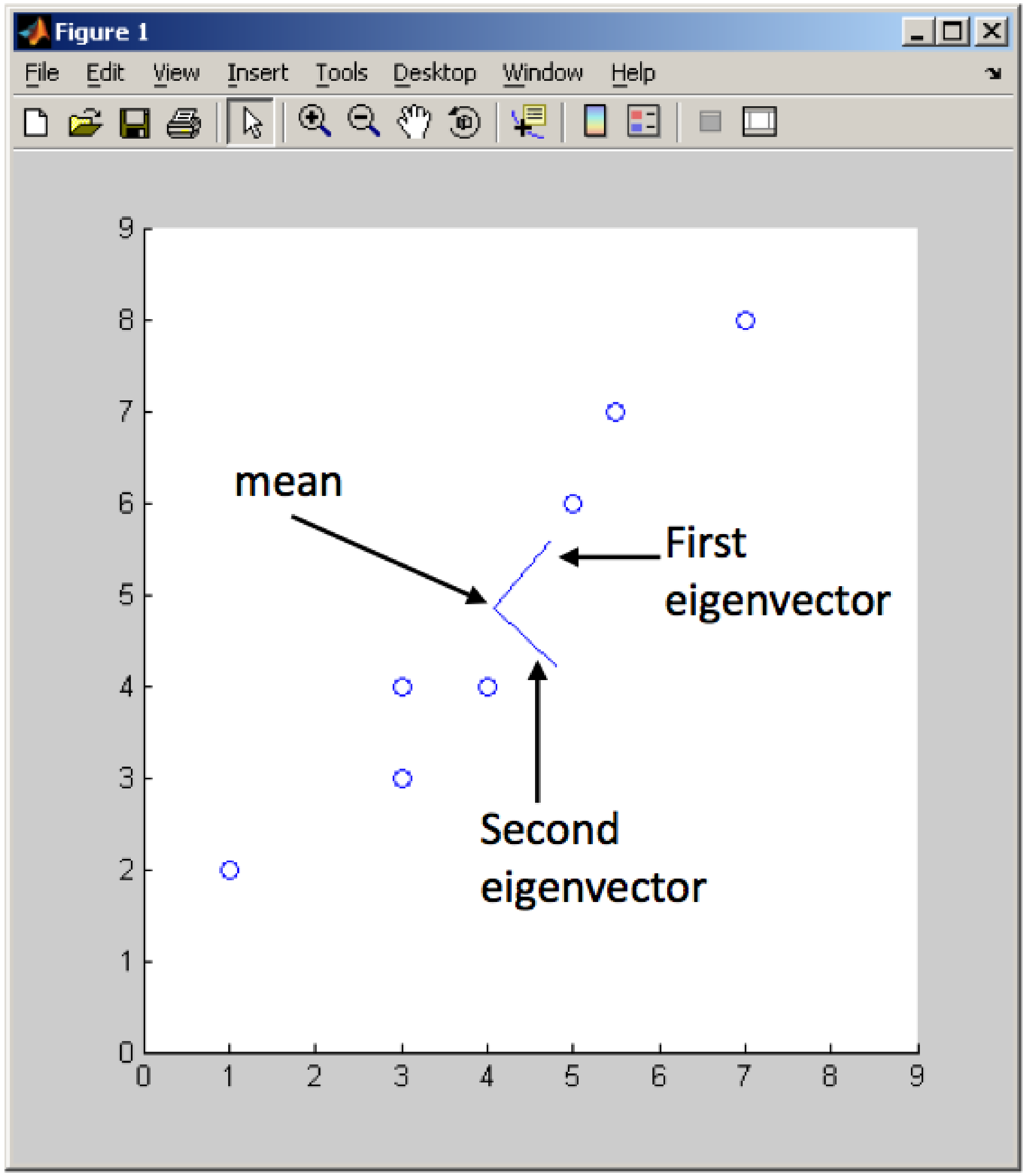
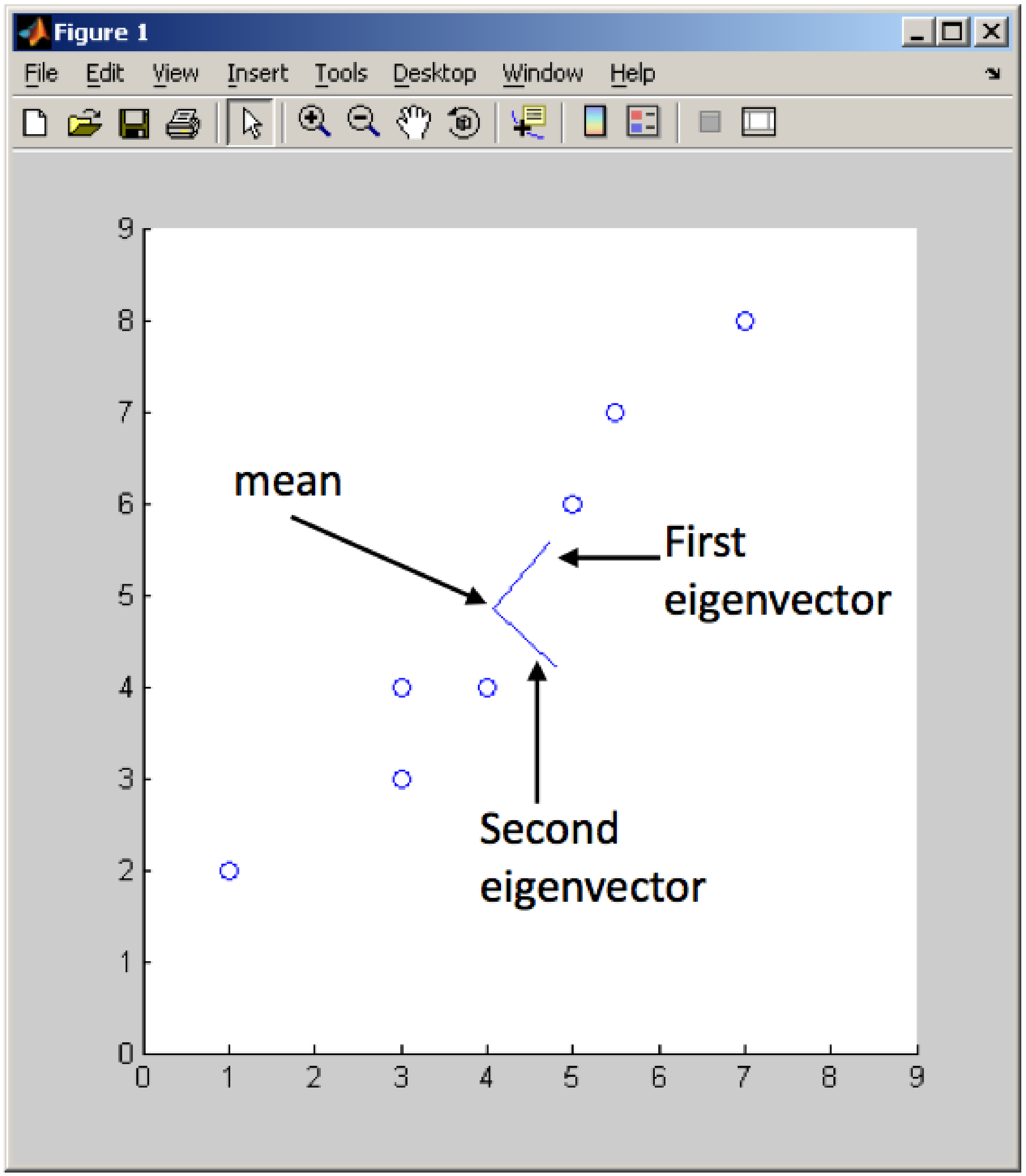
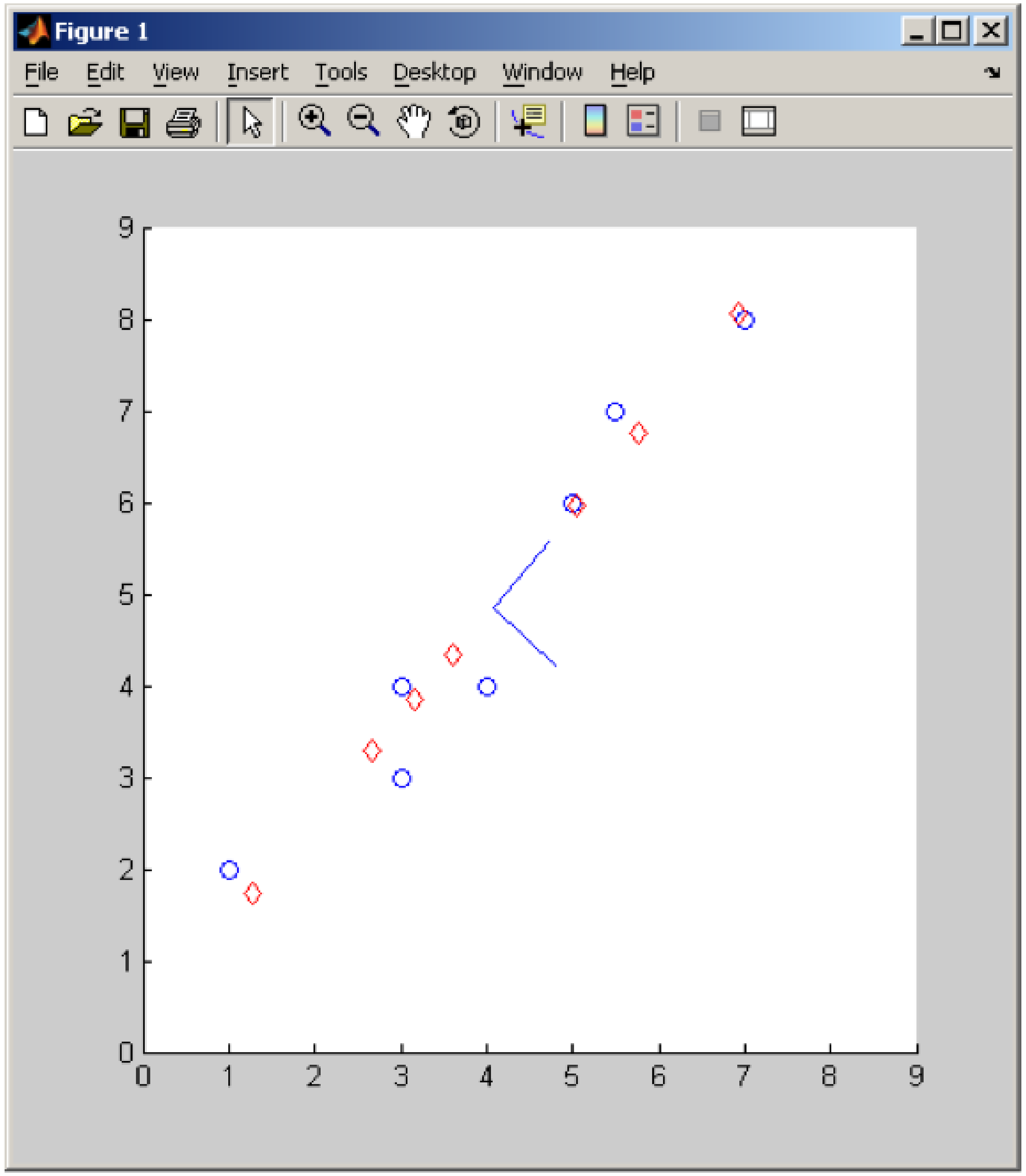


Figure 1
File Edit View Insert Tools Desktop Window Help
mean
First eigenvector
Second eigenvector
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9

PCA example



Reconstructed data using only first eigenvector (M=1)



PCA weaknesses

- Only allows linear projections
- Co-variance matrix is of size dxd. If $d=10^4$, then $|\Sigma| = 10^8$
  - *Solution*: singular value decomposition (SVD)
- PCA restricts to *orthogonal* vectors in feature space that minimize reconstruction error
  - *Solution*: independent component analysis (ICA) seeks directions that are *statistically independent,* often measured using information theory
- Assumes points are multivariate Gaussian
  - *Solution*: Kernel PCA that transforms input data to other spaces

PCA vs. Neural Networks

PCA	Neural Networks
Unsupervised dimensionality reduction	Supervised dimensionality reduction
Linear representation that gives best squared error fit	Non-linear representation that gives best squared error fit
No local minima (exact)	Possible local minima (gradient descent)
Non-iterative	Iterative
Orthogonal vectors (“eigenfaces”)	Auto-encoding NN with linear units may not yield orthogonal vectors

Is this really how humans characterize and identify faces?