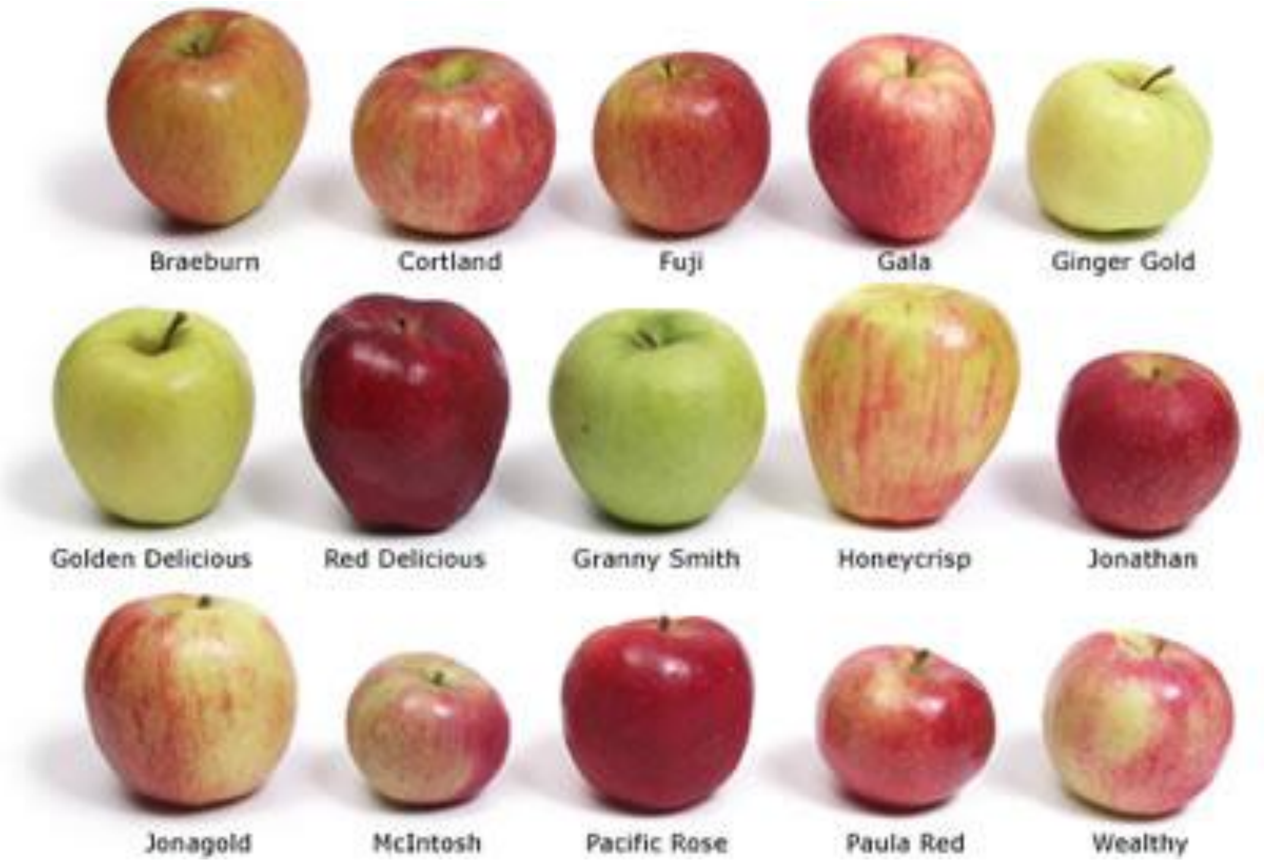
Unsupervised Learning

Supervised vs. Unsupervised Learning

<***x*,*y***> : ***x*** = input, **y** = decision

x_i : often high-dimensional

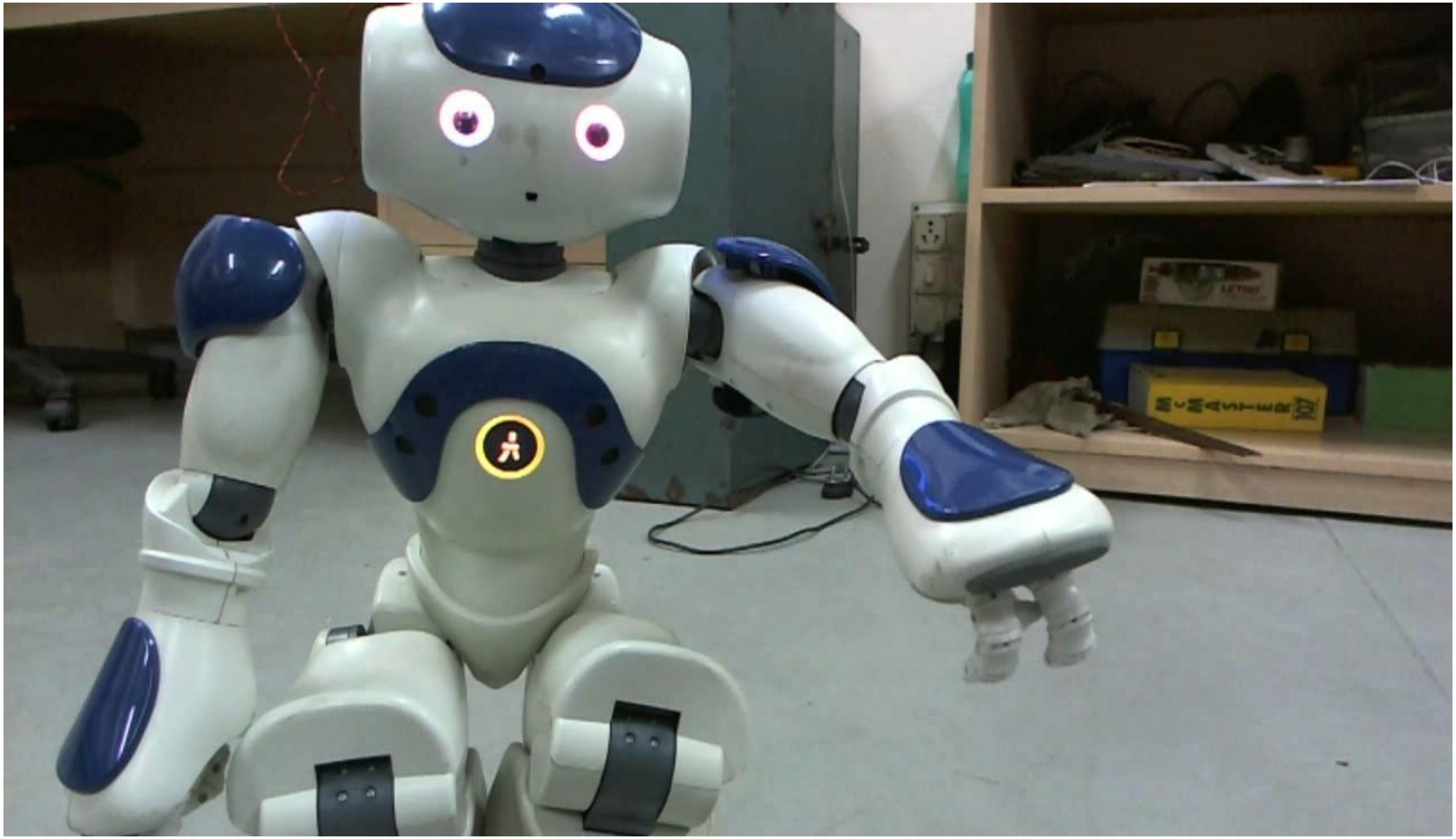
Input spaces : often sparse



images: 100 x 100 pixels

Ack: A. Efros, original images from hormel corp.

Learning to represent



Supervised vs. Unsupervised Learning

<x,y> : x = input, y = decision

x : often high-dimensional

$f : x \rightarrow y$

Difficulty:

Much of the work in identifying a good *f*, is also that of discovering structure in ***x.***

Representations in AI

Representation:

expected to be compact

Traditionally, given as part of the problem specs

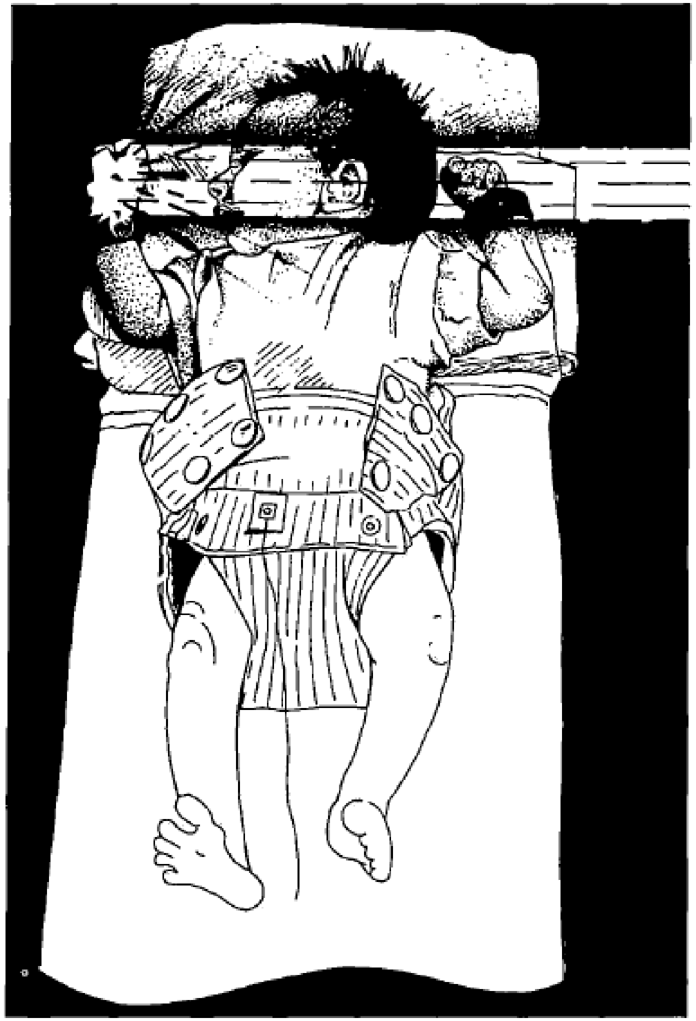
(e.g. determined by a knowledge engineer)

Q. Can we learn representations?

Role of Perception?

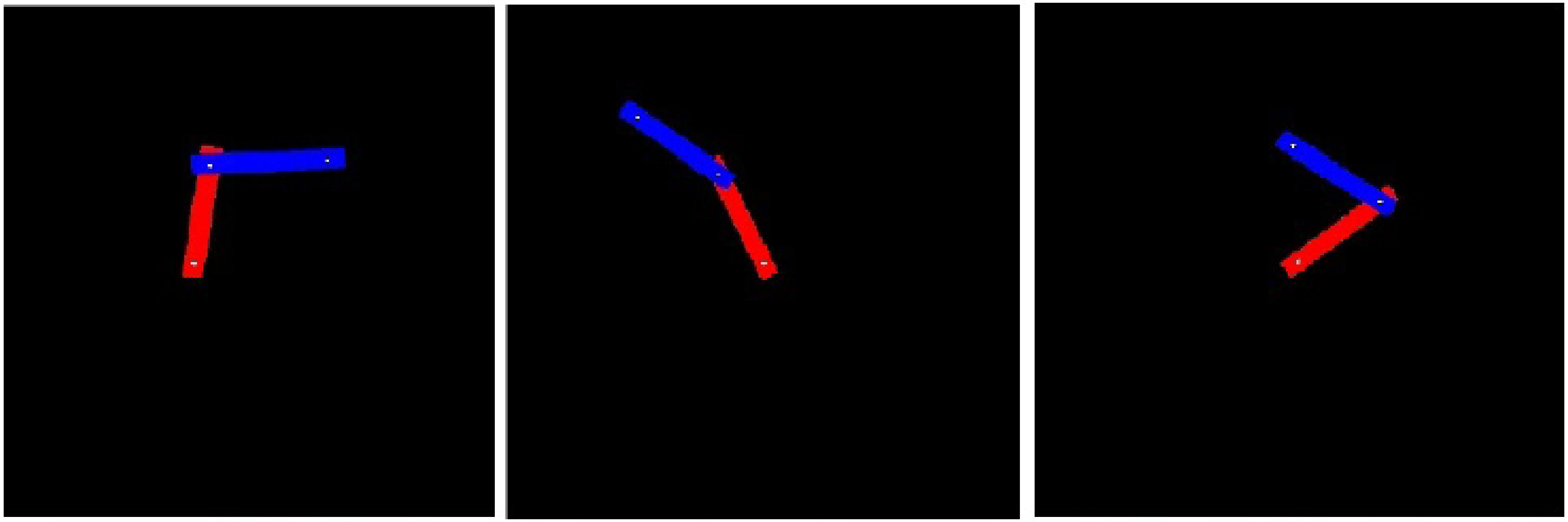
Newborns (10-24 day old) in dark room work hard to position hand so it is visbile in a narrow beam of light. …

Q. Can perception help in learning a representation?



[A. van der Meer, 1997: Keeping the arm in the limelight]

Learning to represent: robot motions



Representations in AI

How to represent a “robot”?

Must include: degrees of freedom (2)

parameters (θ_1, θ_2)

+ rules / functions

A representation for an object is a “frame” or collection of parameters and function associated with the object.

Manifold Learning

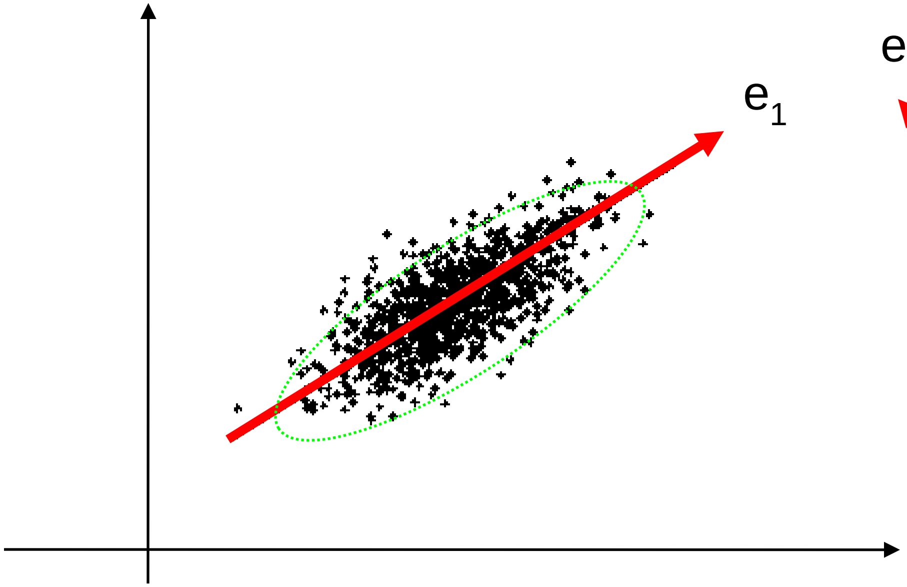
Linear dimensionality reduction

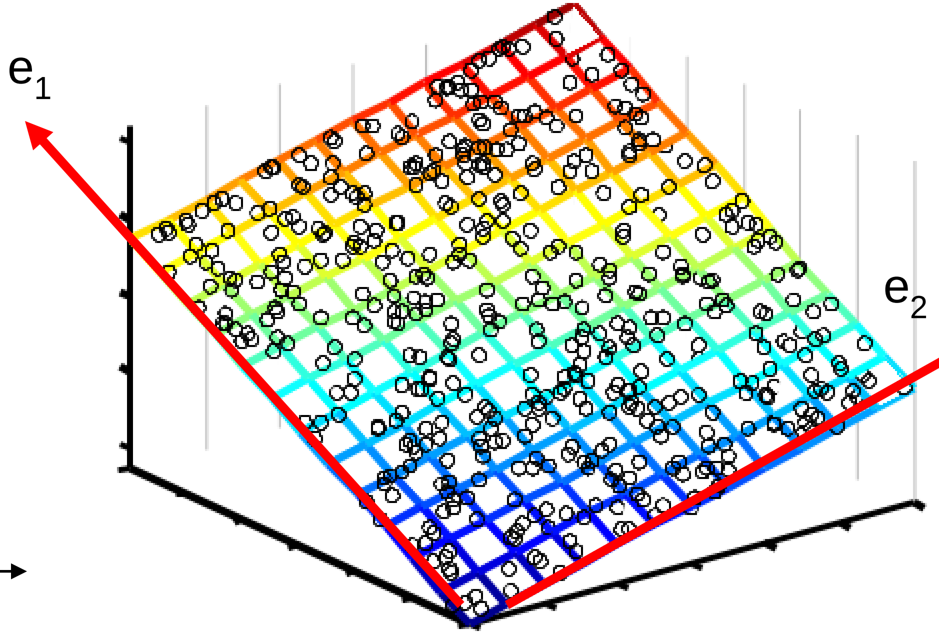
project data onto subspace of maximum variance

PCA: principal components analysis

[A] = top eigenvectors of covariance matrix [XX^T]

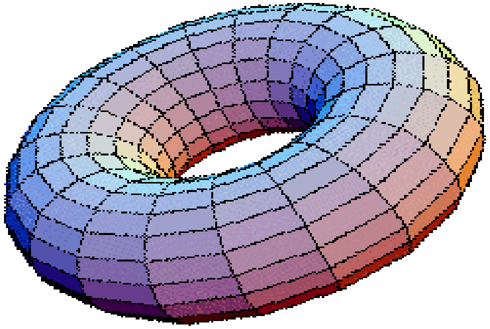
Y = [A] X





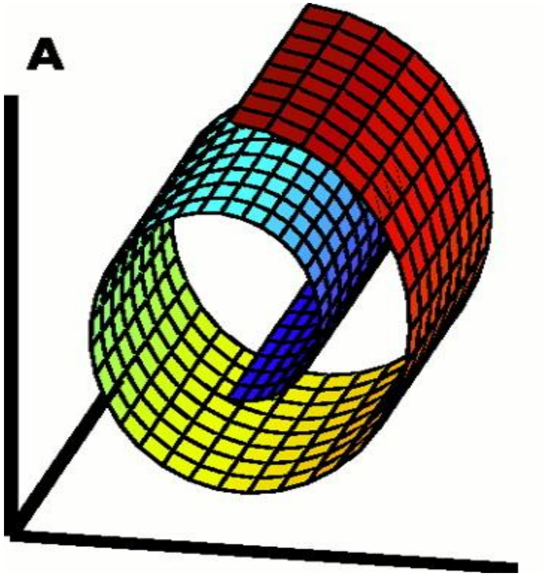
Non-Linear Dimensionality Reduction: Manifolds

A manifold is a topological space which is locally Euclidean.



- nbrhood N in R^n ↔ ball B in R^d (homeomorphic)

- Homeomorphic: Every x in N has a map to a y in B
- Dimensionality of manifold = d
- Embedding dimension = n



Manifolds

A manifold is a topological space which is locally Euclidean.

nbrhood in R^n ↔ ball in R^d
(homeomorphic)

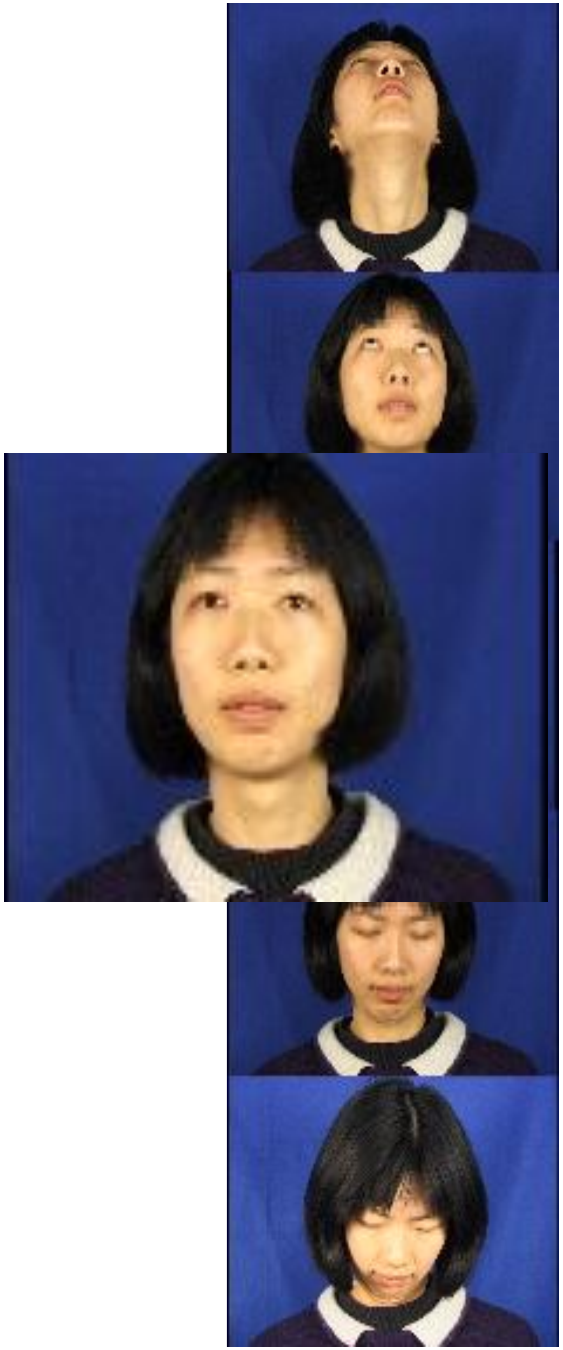
Dimensionality of manifold = d

Embedding dimension = n

Real life data (e.g. images) : $D = 10^5$
motions = smooth variation
of just a few parameters

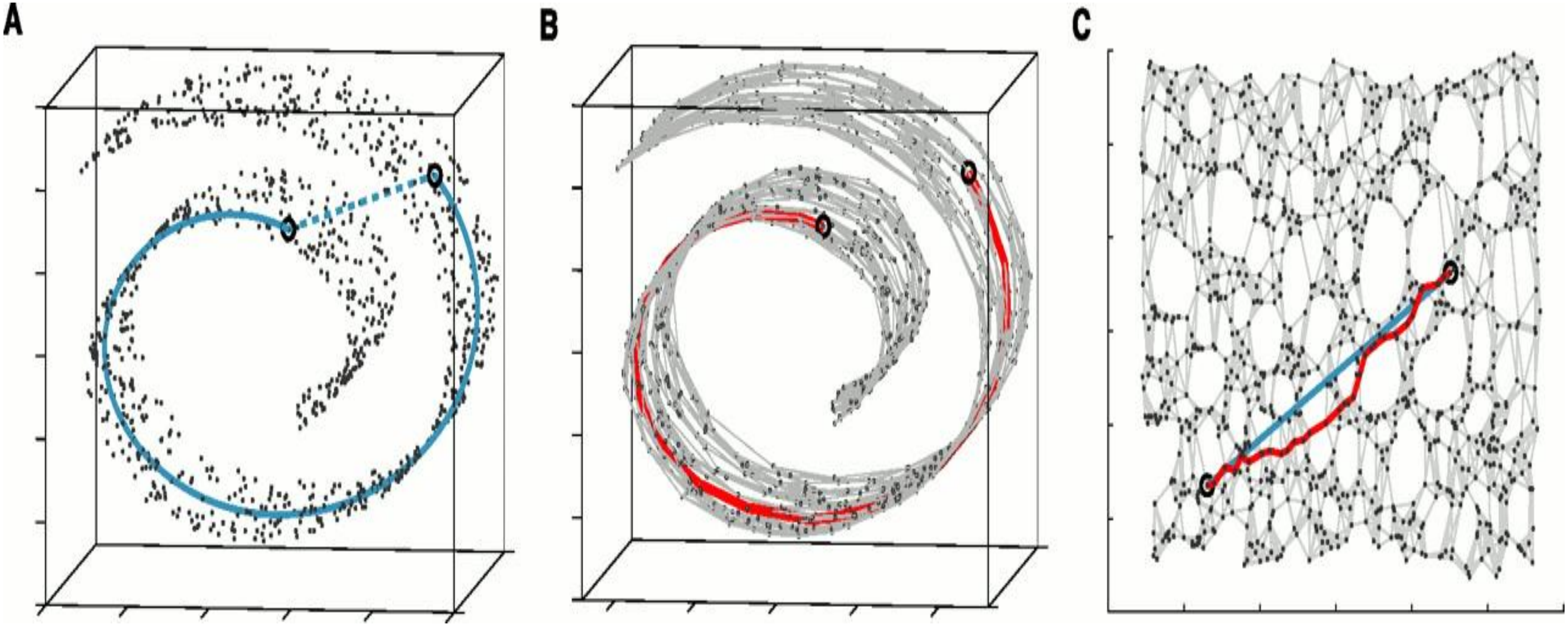
DOFs = pose of faces → $d = 1$

Ideally, d = number of varying parameters



Non-Linear Dimensionality Reduction (NLDR) algorithms: ISOMAP

Euclidean or Geodesic distance?



Geodesic = shortest path along manifold

Isomap Algorithm

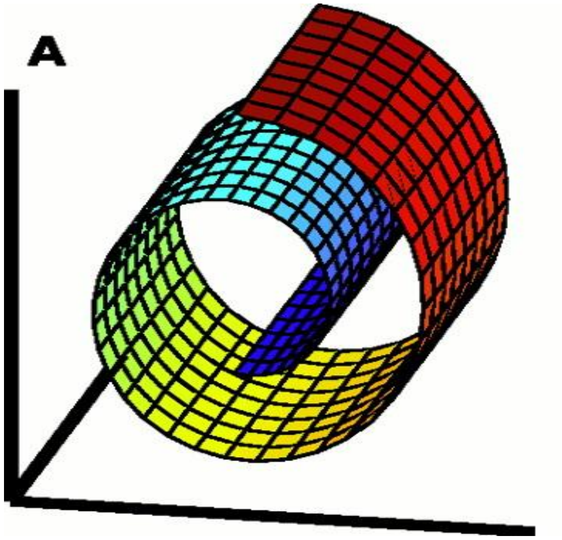
- Identify neighbors.
  - points within epsilon-ball (*ε*-ball)
  - *k* nearest neighbors (*k*-NN)
- Construct neighborhood graph.

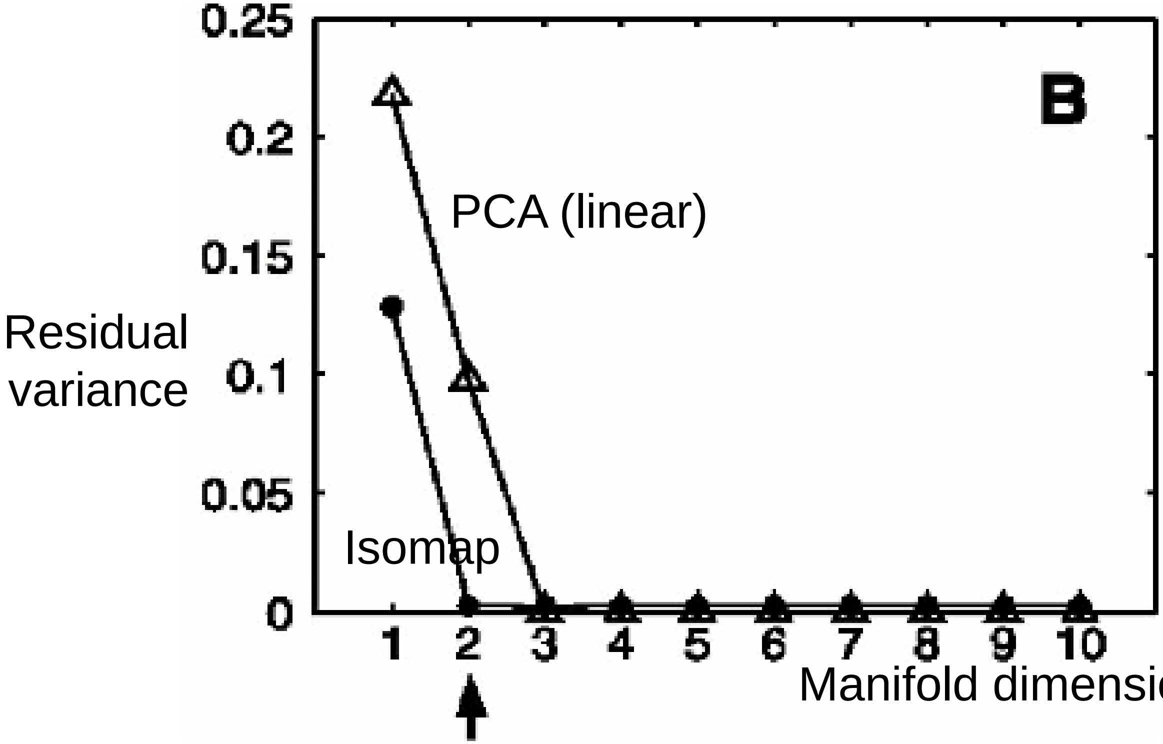
  -- *x* connected to *y* if *neighbor(x,y)*.

  -- edge length = distance(x,y)
- Compute shortest path between nodes
  - Djkastra / Floyd-Warshall algorithm
- Construct a lower dimensional embedding.
  - Multi-Dimensional Scaling (MDS)

[Tenenbaum, de Silva and Langford 2001]

Residual Variance and Dimensionality





residual variance = $1 - r^2(D_g, D_y)$; r = linear correlation coefficient

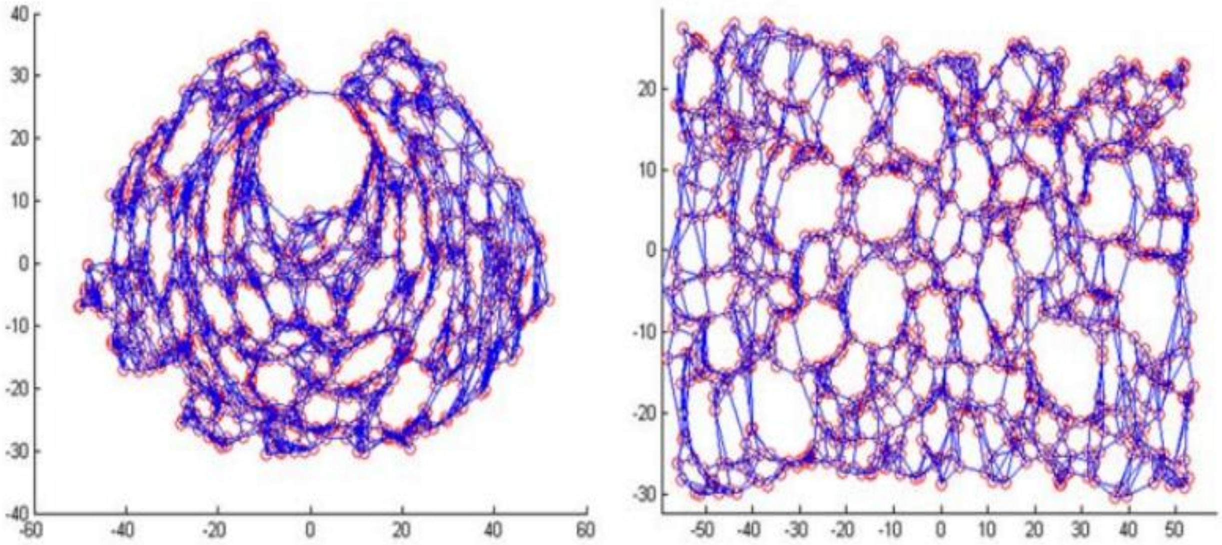
D_g = geodesic distance matrix; D_y = manifold distance

Short Circuits & Neighbourhood selection

neighbourhood size

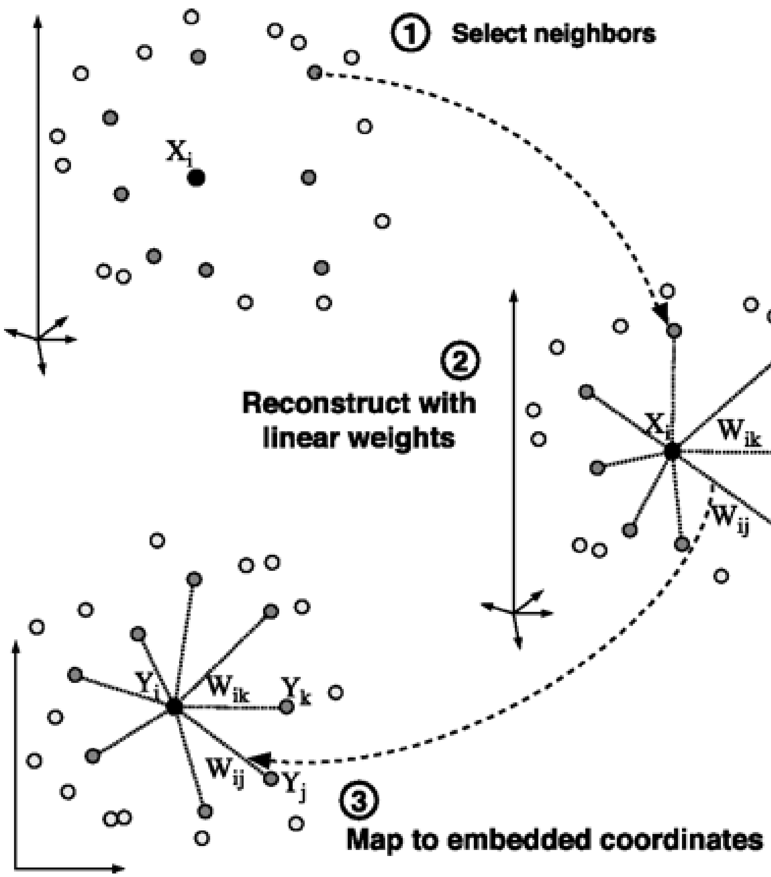
too big: short-circuit errors

too small: isolated patches



[saxena, gupta mukerjee 04]

Locally-Linear Embedding



① Select neighbors
X_i
② Reconstruct with linear weights
X_i
W_ik
X_k
W_ij
X_j
Y_i
W_ik
Y_k
W_ij
Y_j
③ Map to embedded coordinates

$$J(W) = \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{K} w_{ij} x_j \right\|^2$$

$$J_2(Y) = \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{N} w_{ij} y_j \right\|^2$$

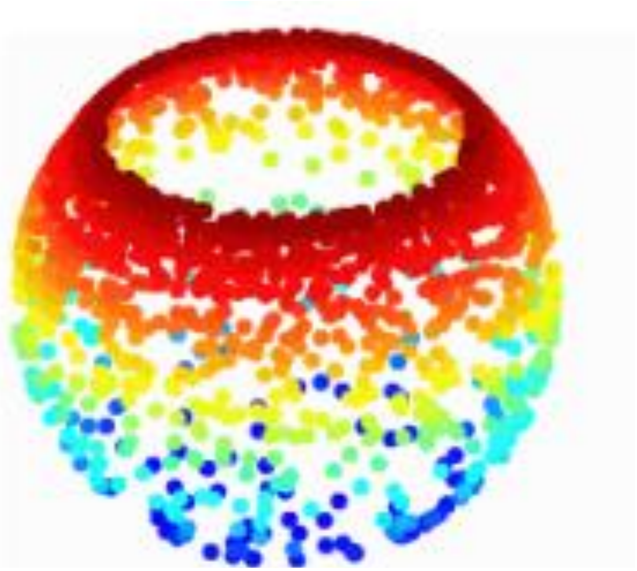
$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \\ | & | & & | \end{bmatrix}_{dxN} = \begin{bmatrix} - & \mathbf{u}_1 & - \\ - & \mathbf{u}_2 & - \\ & \vdots & \\ - & \mathbf{u}_d & - \end{bmatrix}_{dx}$$

Non-isometric maps

Fishbowl dataset : no isomorphic map to plane

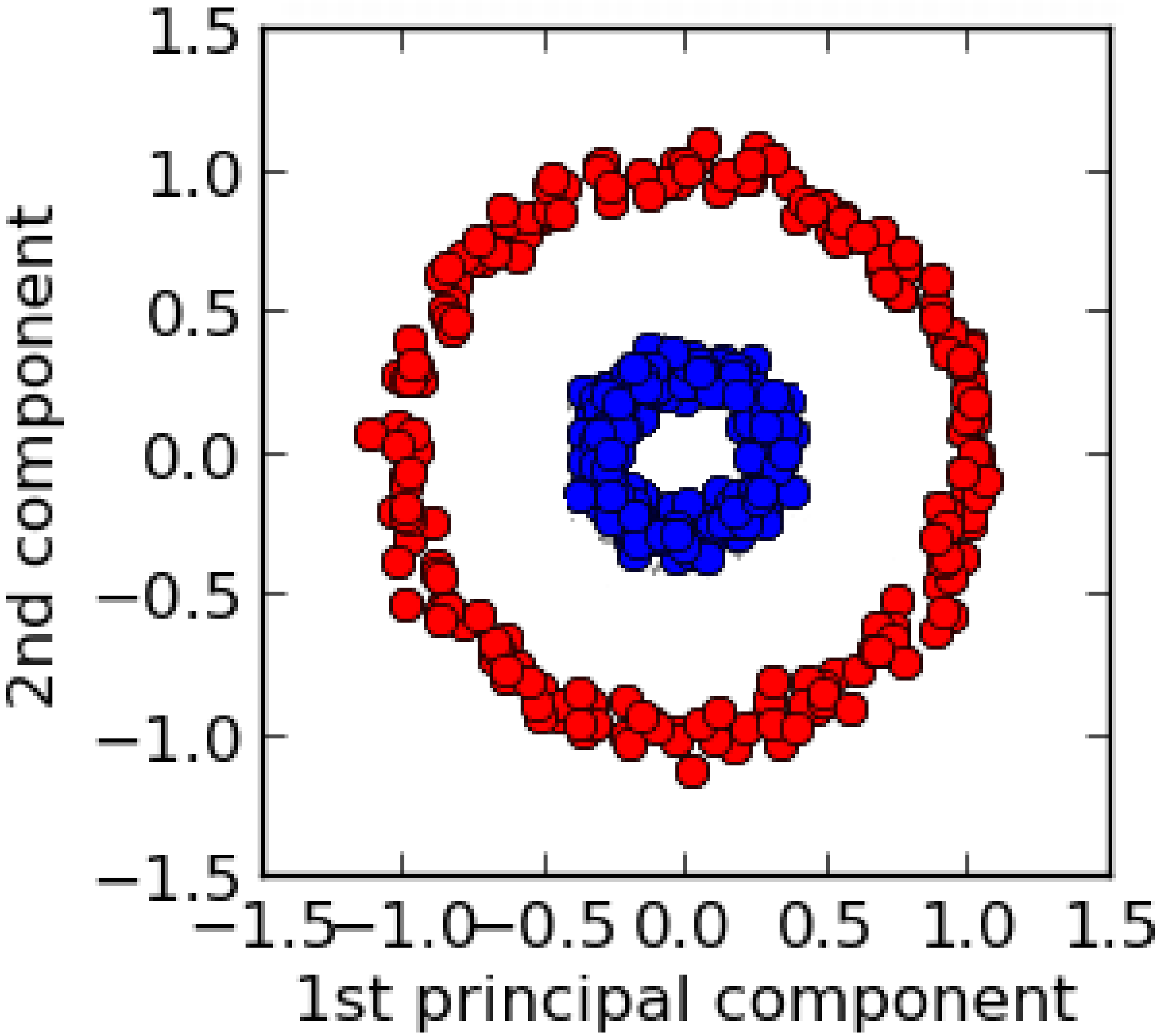
- Conformal mappings: preserve angles,
  not distances

- Assume data is uniformly distributed in low dim

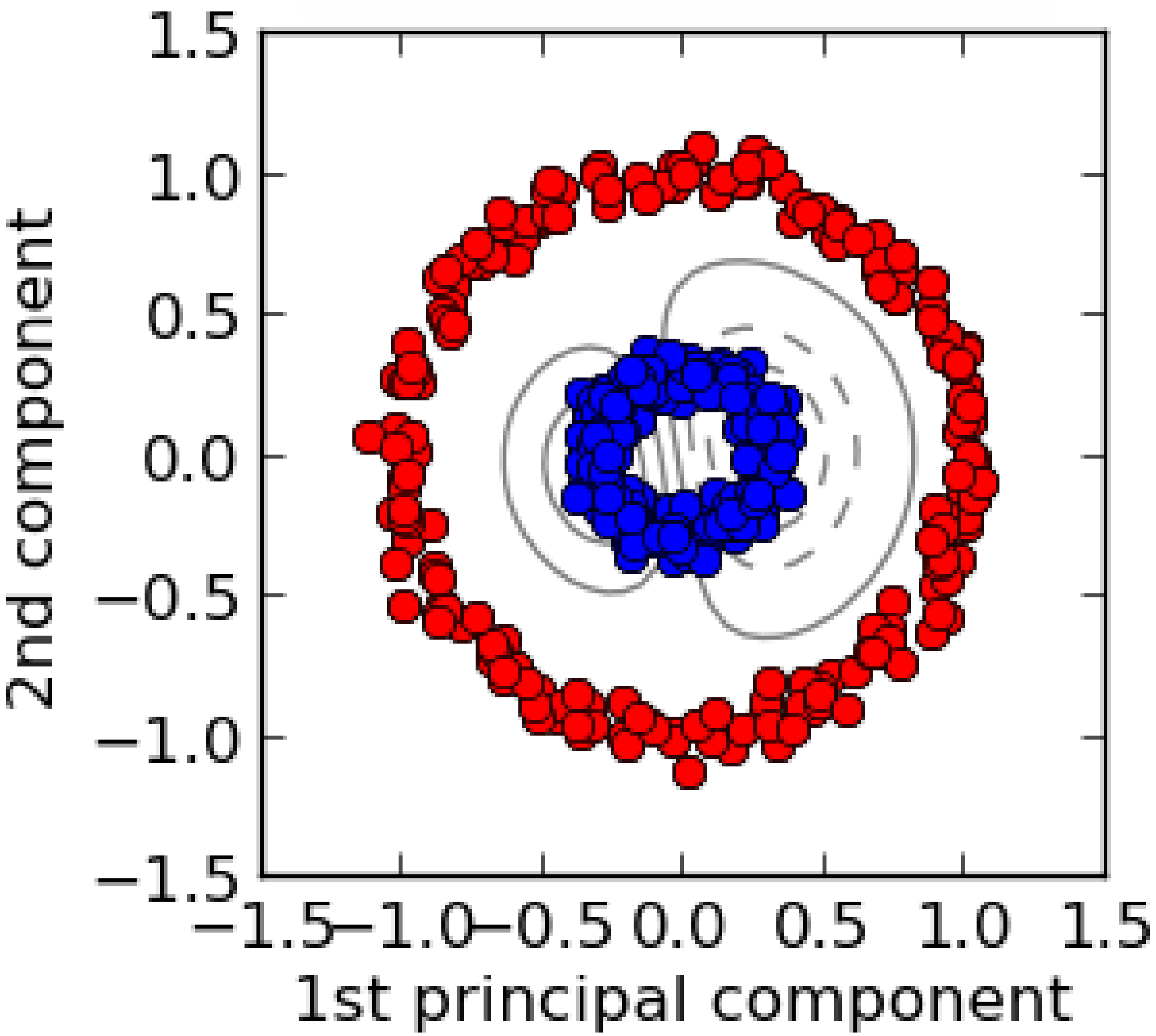


Kernel-PCA

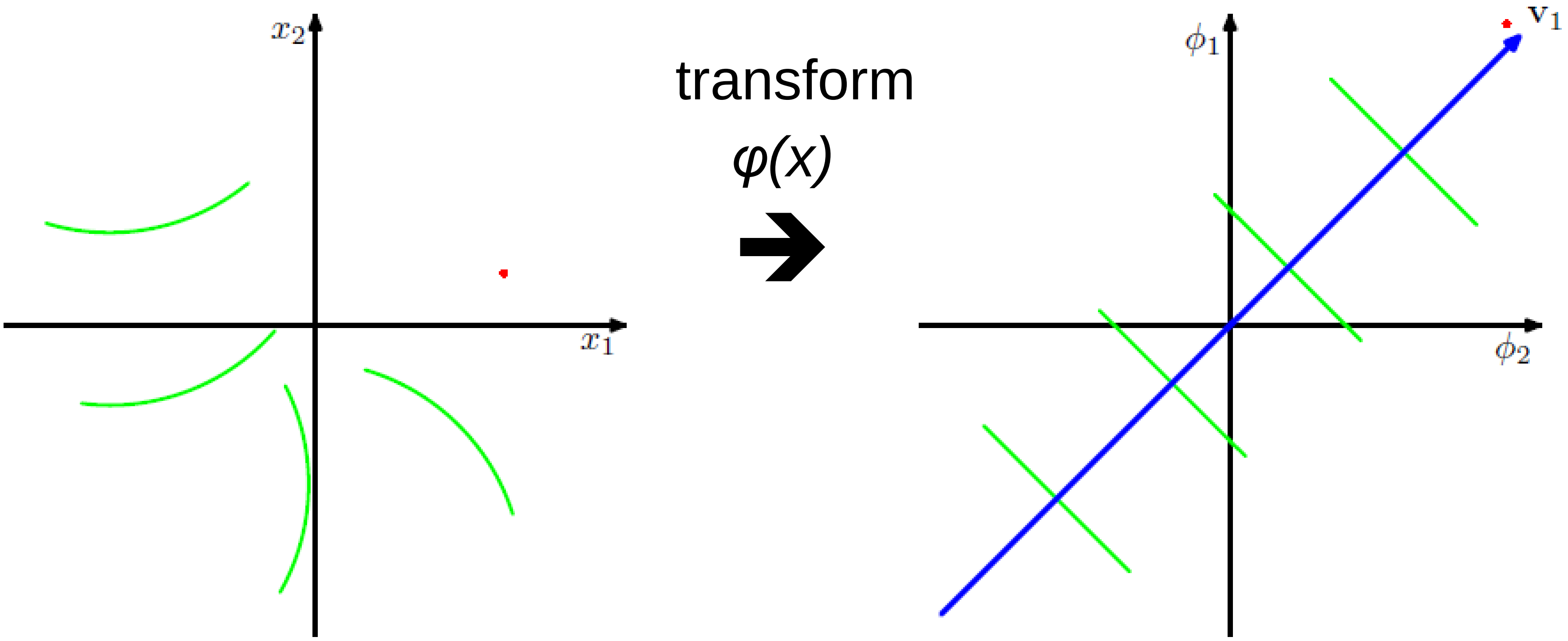
PCA on non-linear data



PCA on non-linear data



Non-linear PCA?



transform

φ(x)

Kernel PCA

PCA: top eigenvectors of covariance matrix $[XX^T]$

Kernel PCA: replace X by $\phi(x)$

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}}$$

Eigenvalue expression $\boldsymbol{C}\boldsymbol{v}_i = \lambda_i\,\boldsymbol{v}_i$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\overbrace{\left\{\phi(\mathbf{x}_n)^{\mathrm{T}}\mathbf{v}_i\right\}}^{\textbf{scalar}} = \lambda_i\mathbf{v}_i$$

To express in terms of kernel fn $k(x_n, x_m) = \varphi(x_n)^T\varphi(x_m)$, substitute

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)$$

Bishop section 12.5

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}}\sum_{m=1}^{N}a_{im}\phi(\mathbf{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(\mathbf{x}_n).$$

Multiply both sides by $\varphi(x_n)^T$

$$\frac{1}{N}\sum_{n=1}^{N}k(\mathbf{x}_l,\mathbf{x}_n)\sum_{m=1}^{N}a_{im}k(\mathbf{x}_n,\mathbf{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}k(\mathbf{x}_l,\mathbf{x}_n).$$

which reduces to

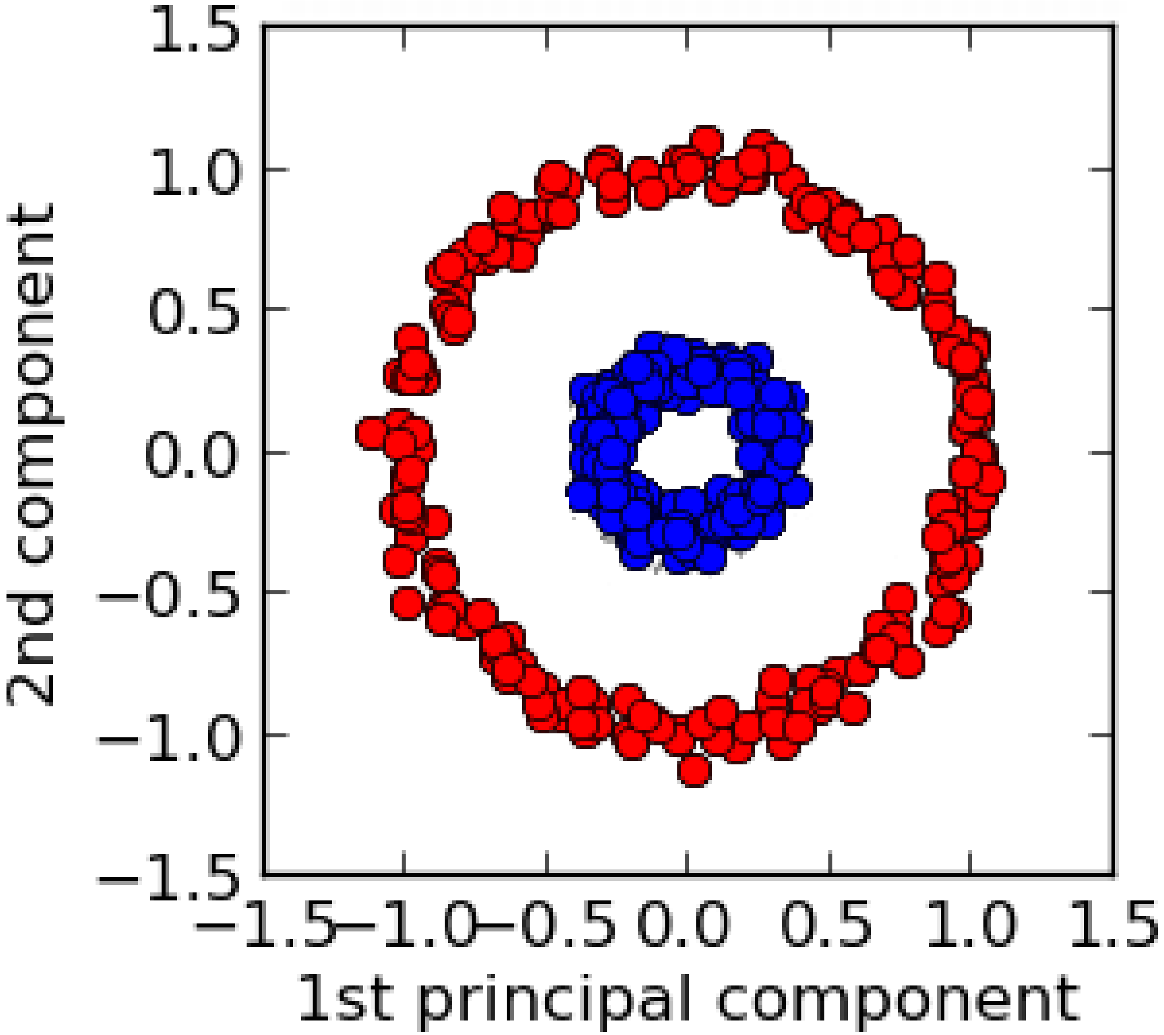
$$\boldsymbol{K}\boldsymbol{a}_i = \lambda_i N \boldsymbol{a}_i$$

(K is semi-positive definite; removing from both sides – affects only zero λ_i).

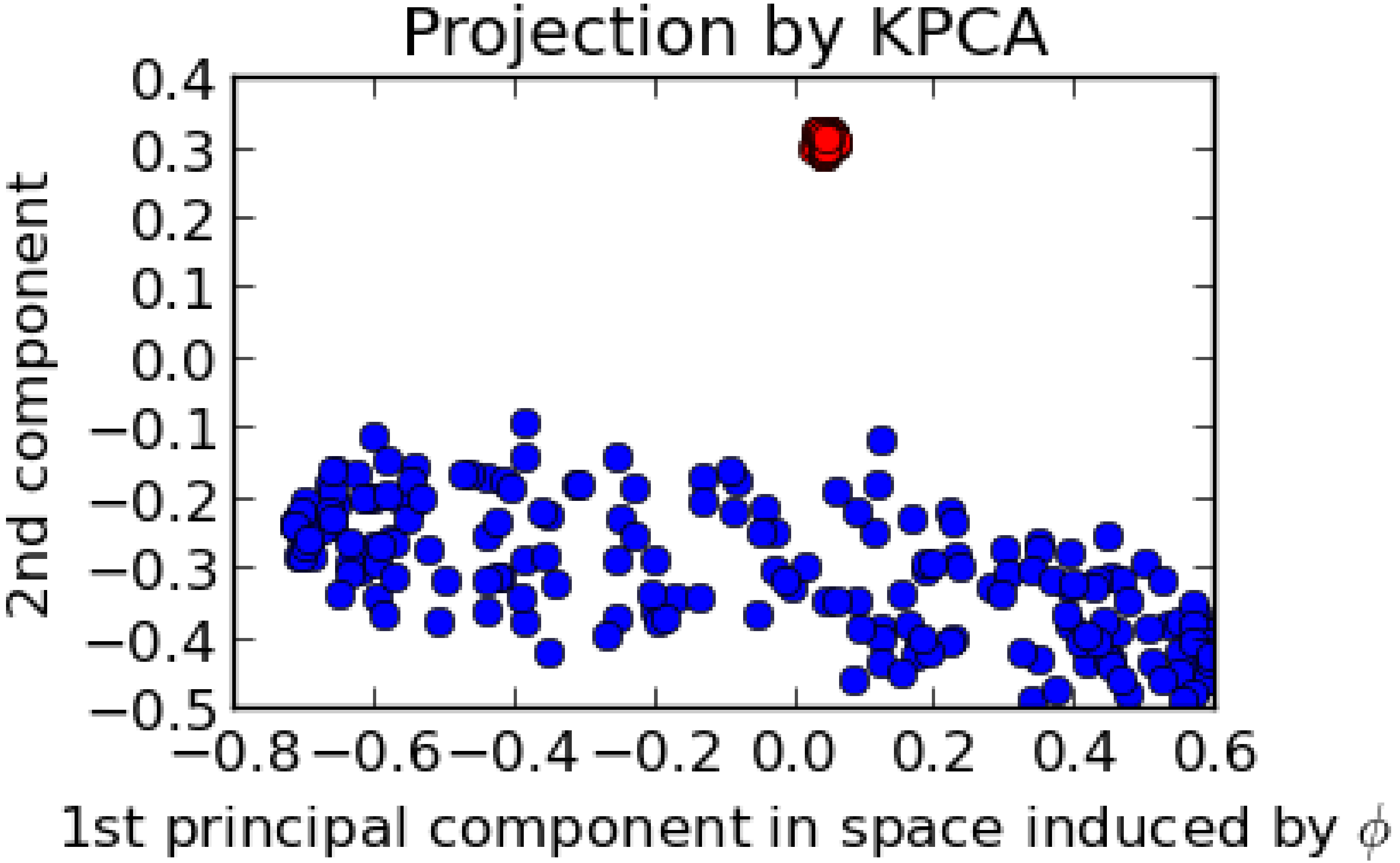
Projections y_i = $\sum_{n=1}^{N}a_{in}k(\mathbf{x},\mathbf{x}_n)$

Q. What happens when we use a linear kernel $k(x, x') = x^T x'$?

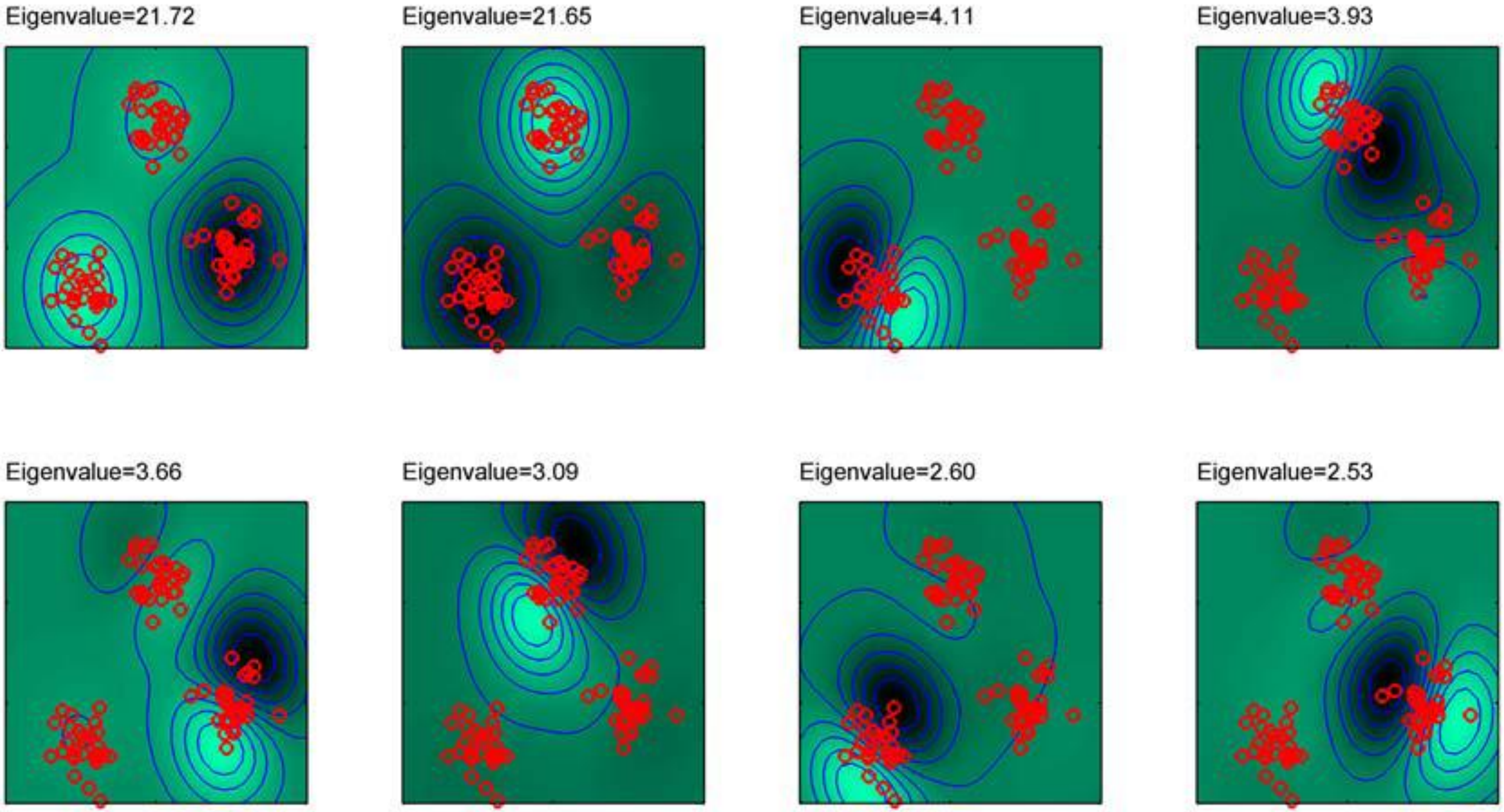
Kernel PCA


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
2nd component
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
1st principal component

Kernel PCA


Projection by KPCA
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
2nd component
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
1st principal component in space induced by ϕ

Kernel PCA : Demonstration



Eigenvalue=21.72

Eigenvalue=21.65

Eigenvalue=4.11

Eigenvalue=3.93

Eigenvalue=3.66

Eigenvalue=3.09

Eigenvalue=2.60

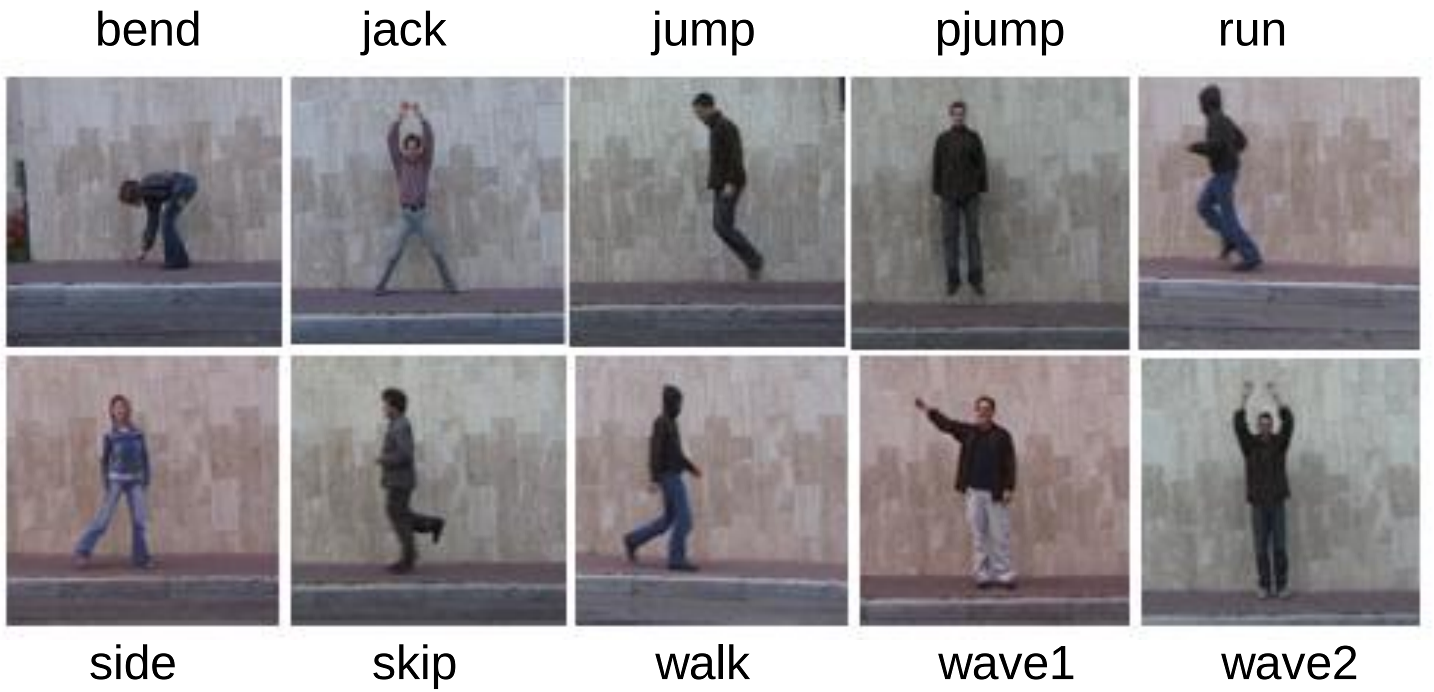
Eigenvalue=2.53

Kernel: $k(x, x') = \exp(-|x - x'|^2 / 0.1)$

[Scholkopf 98]

Manifolds in video

Dimensionality of Actions

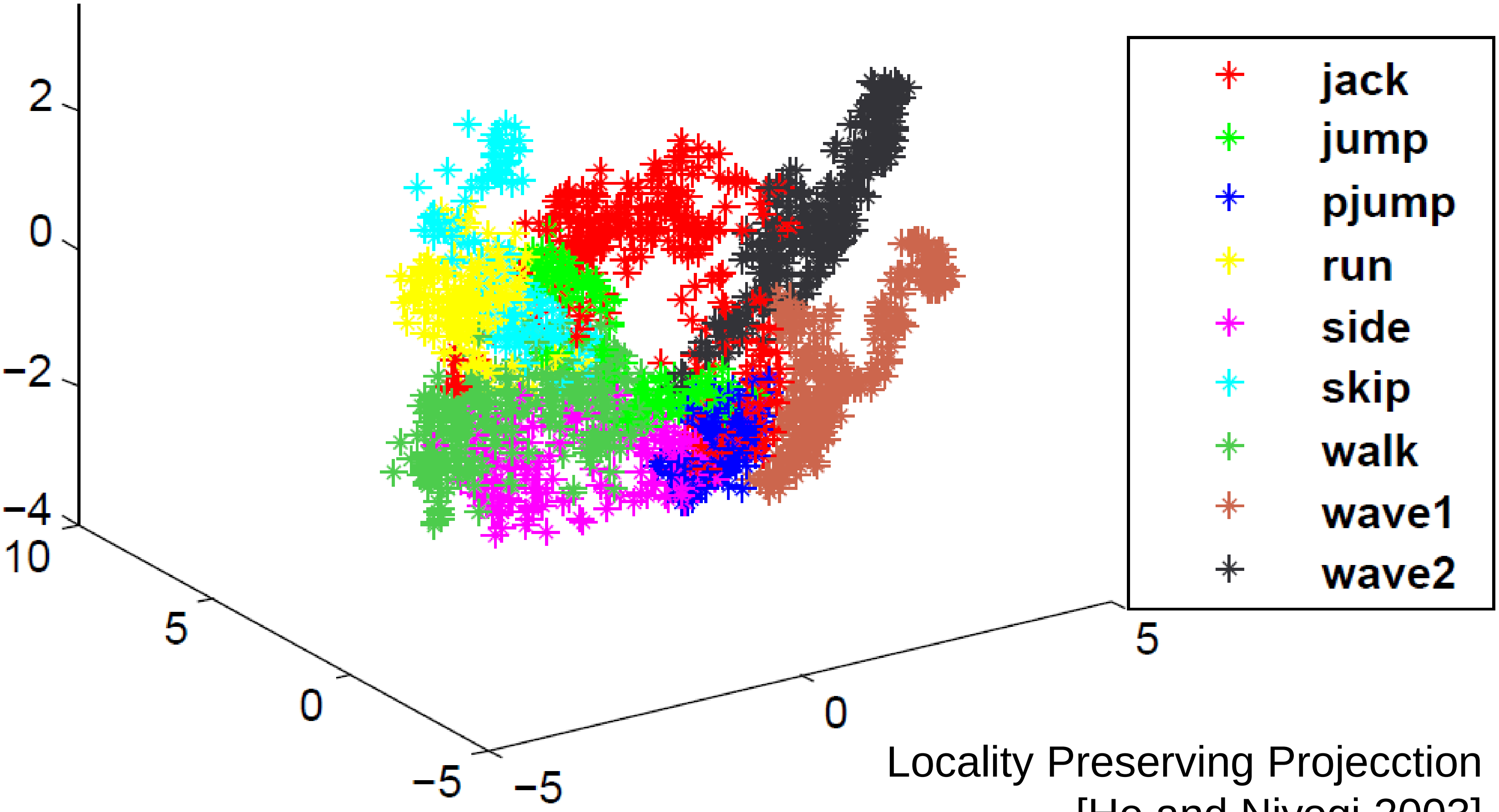


Weizmann activity dataset:

videos of 10 actions by 12 actors

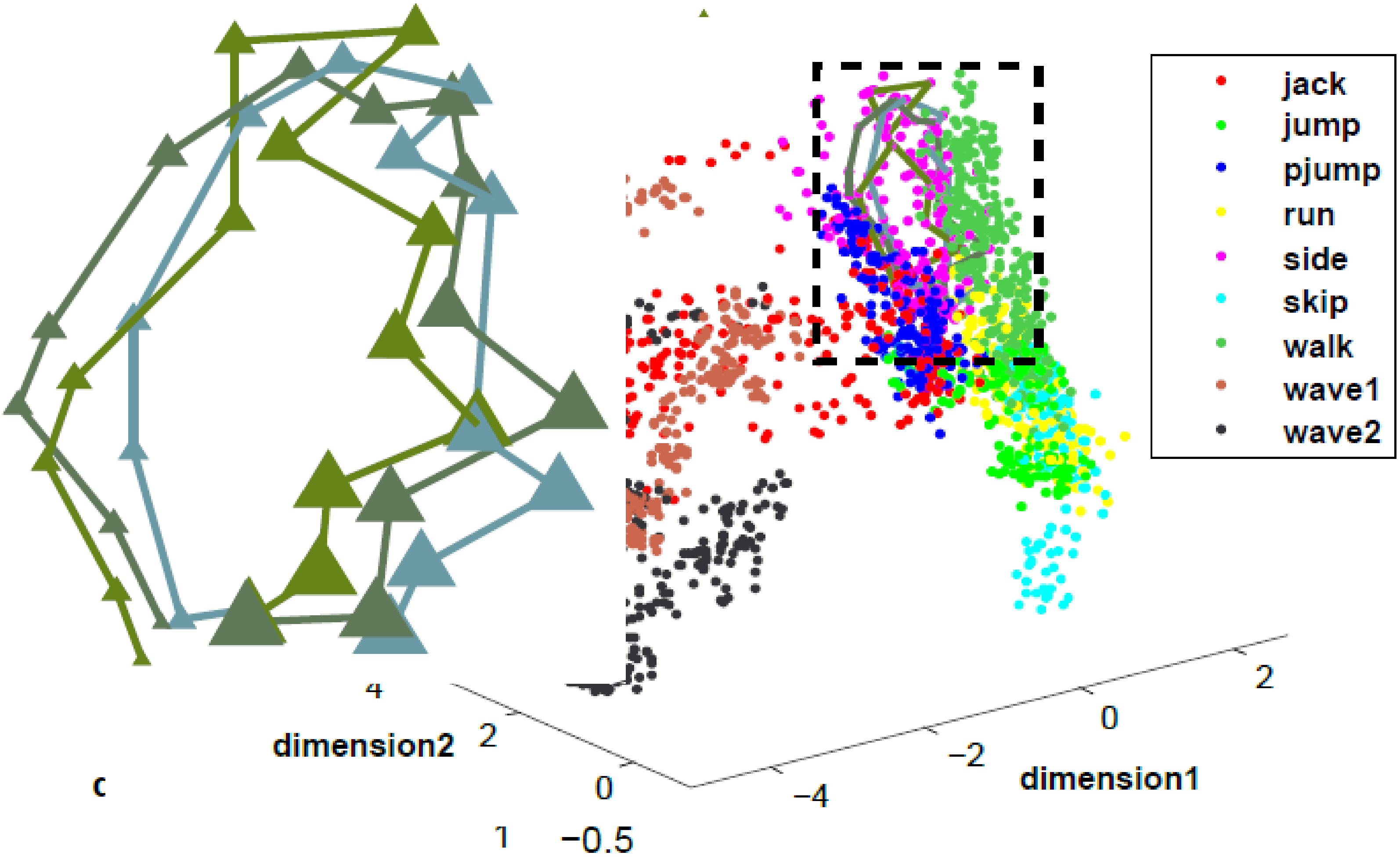
[Gorelick / Blank / Irani : 2005 / 07]

Reduced dimensionality

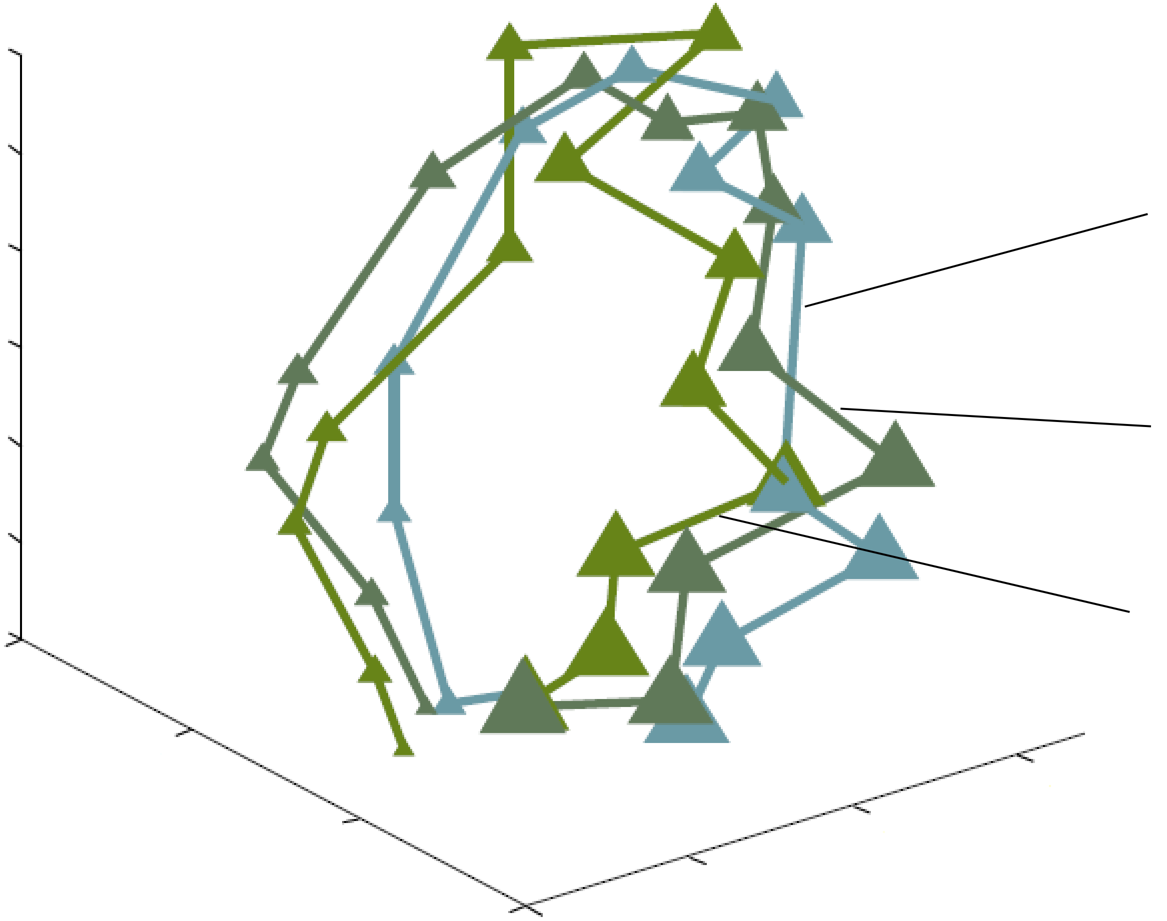


Locality Preserving Projecction
[He and Niyogi 2003]

Gestures in low dimensions



Recognizing gestures

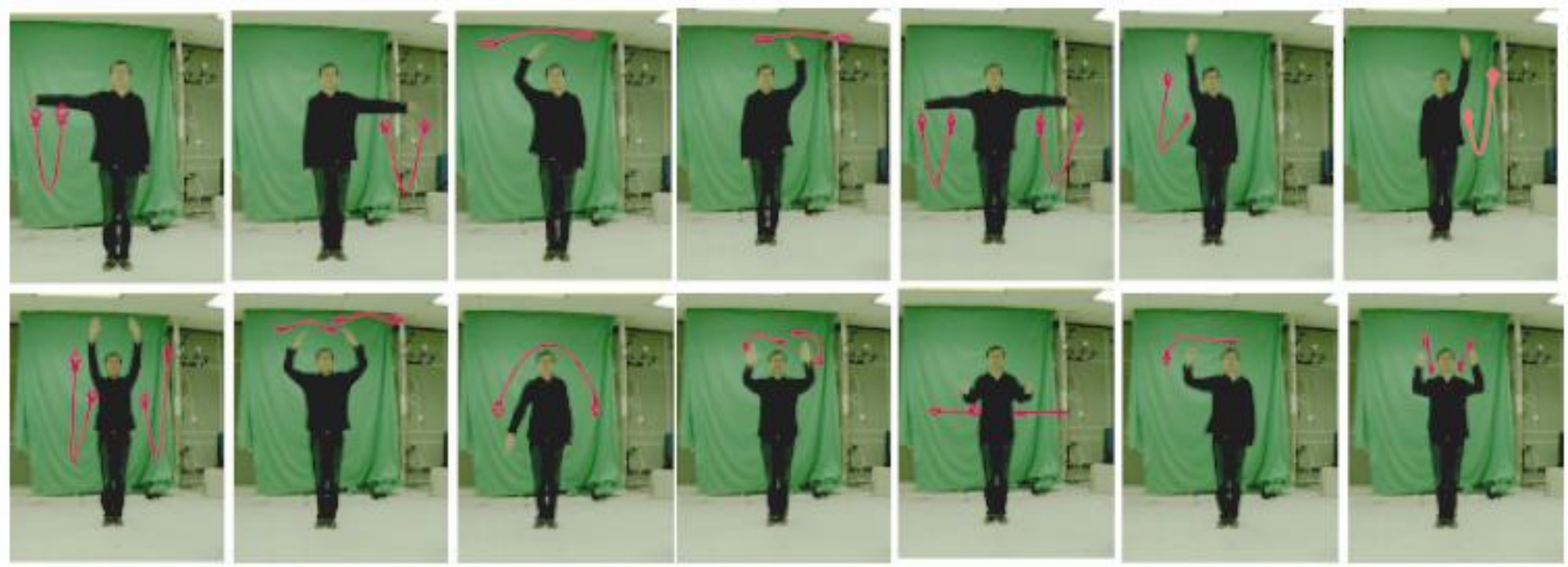


HMM
1

HMM2

HMM3

Recognizing gestures



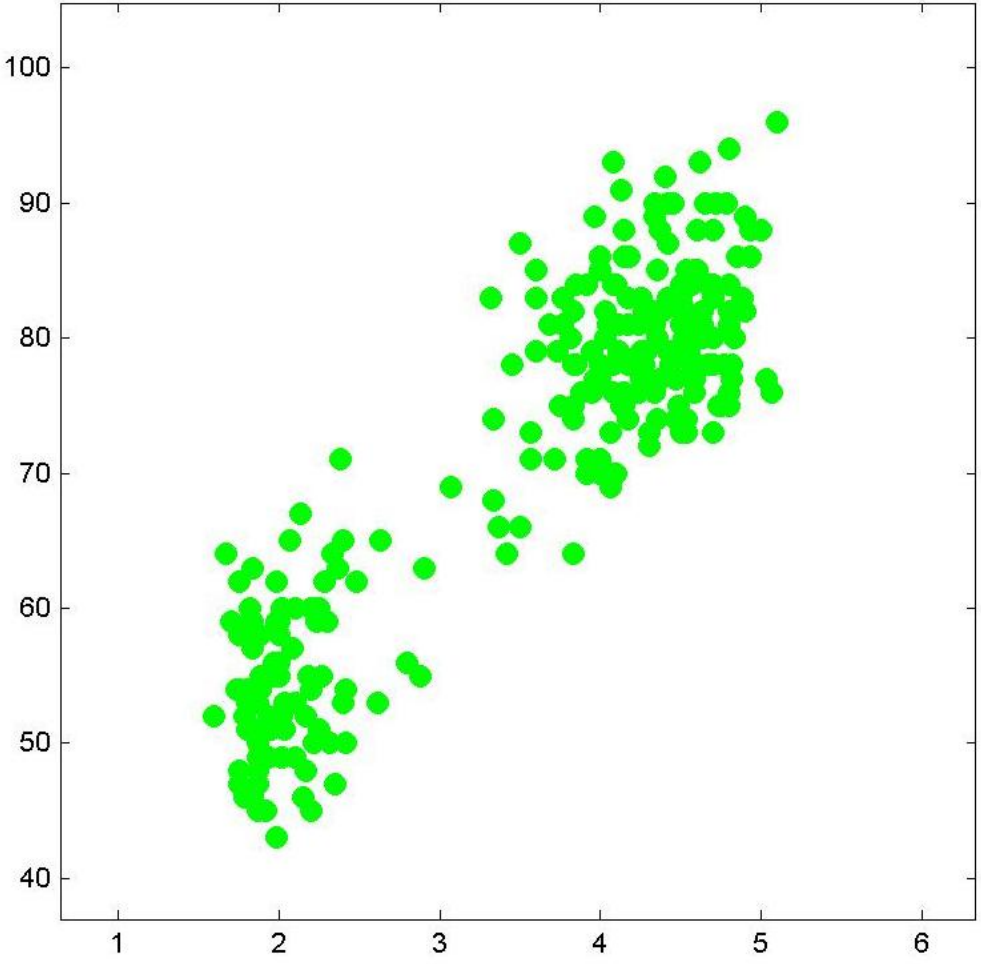
Keck gesture dataset

Expectation Maximization

Old Faithful Geyser

Slides: Christopher M. Bishop

Old Faithful Data Set



Time between eruptions (minutes)

Duration of eruption (minutes)

Expectation Maximization

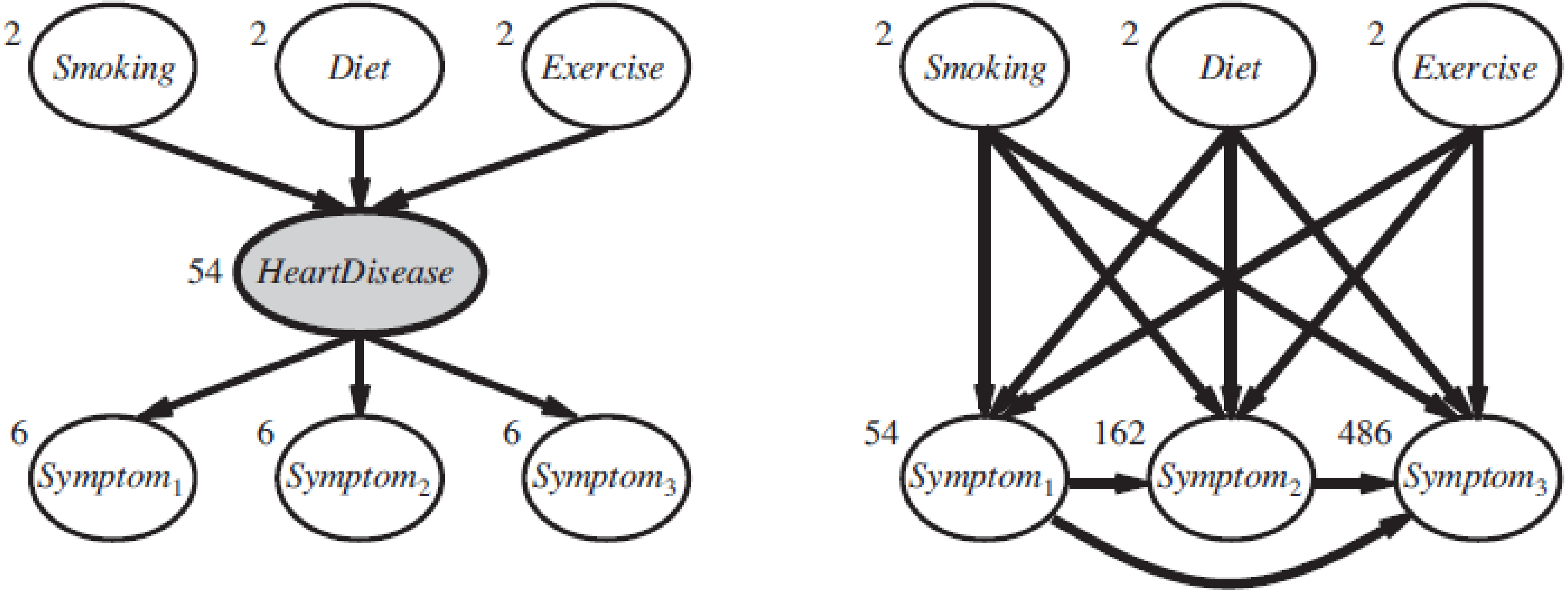
- Select a prototype (model) – e.g. *k* spherical clusters

- E-step: represent the data by assigning it to the nearest model . Compute the Expectation of the data for this assignment.

- M-step: Identify the parameters for the model so as to maximize the likelihood of the parameters

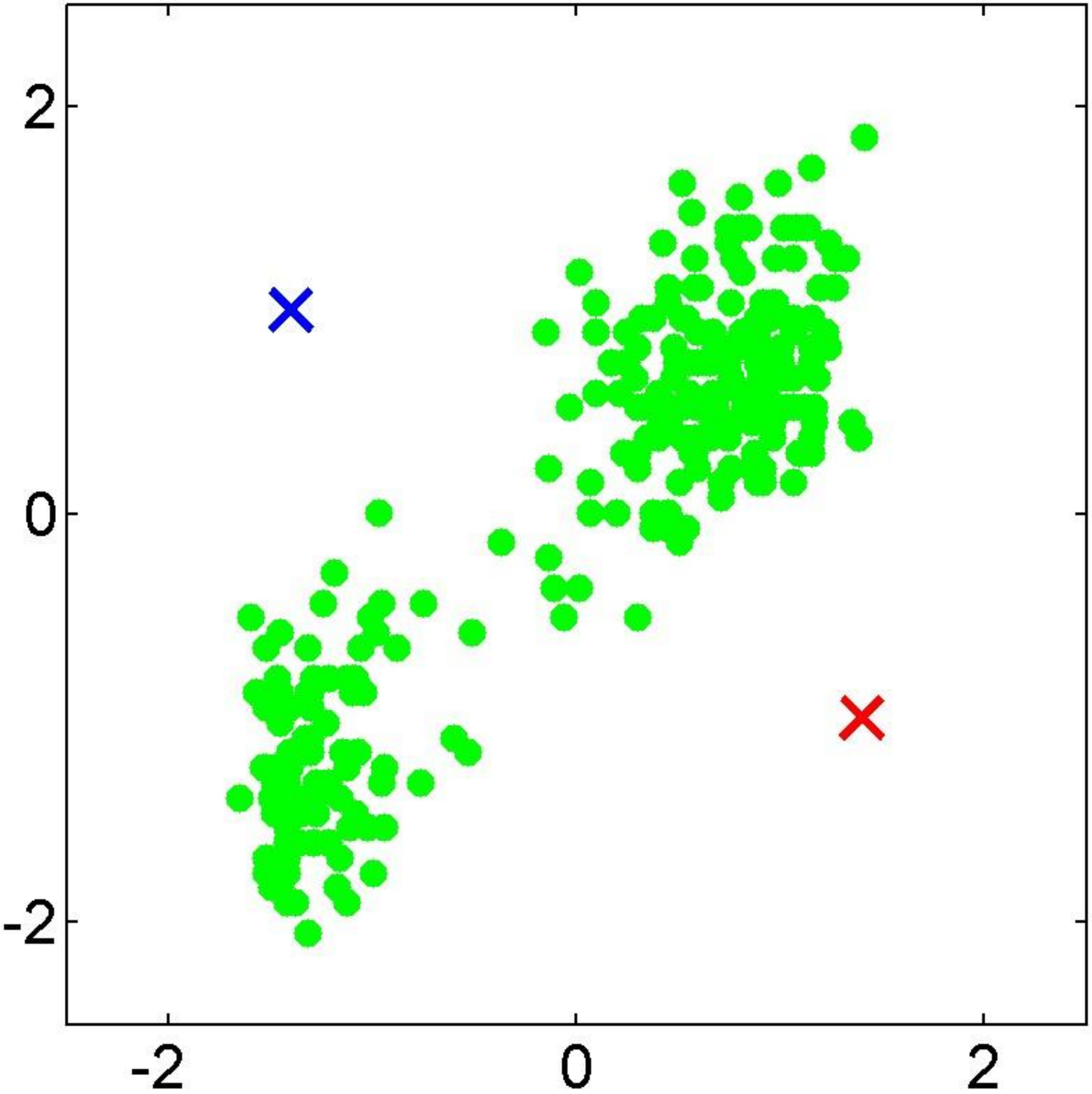
- How to minimize the number of parameters?

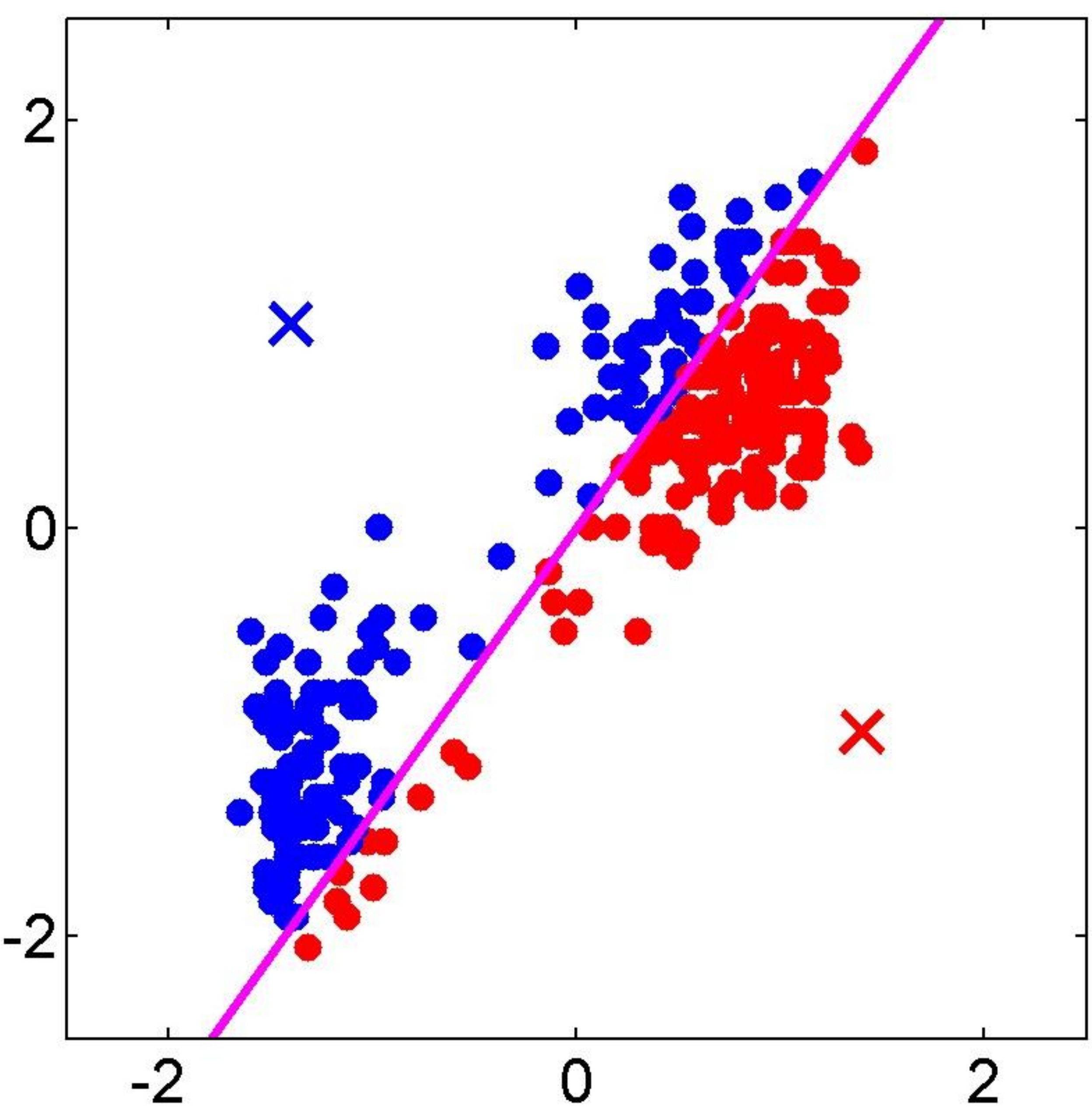
Assume latent state

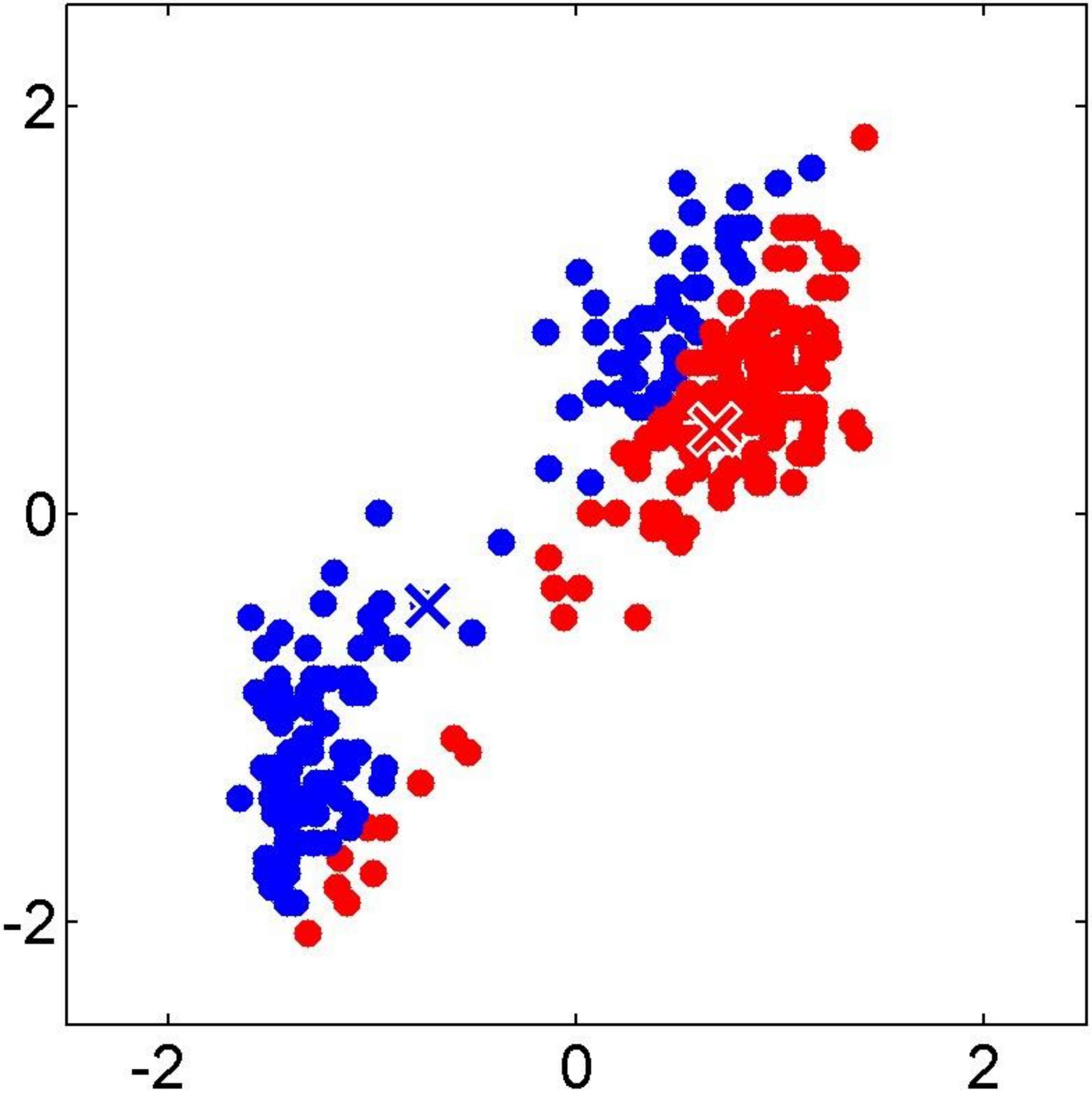


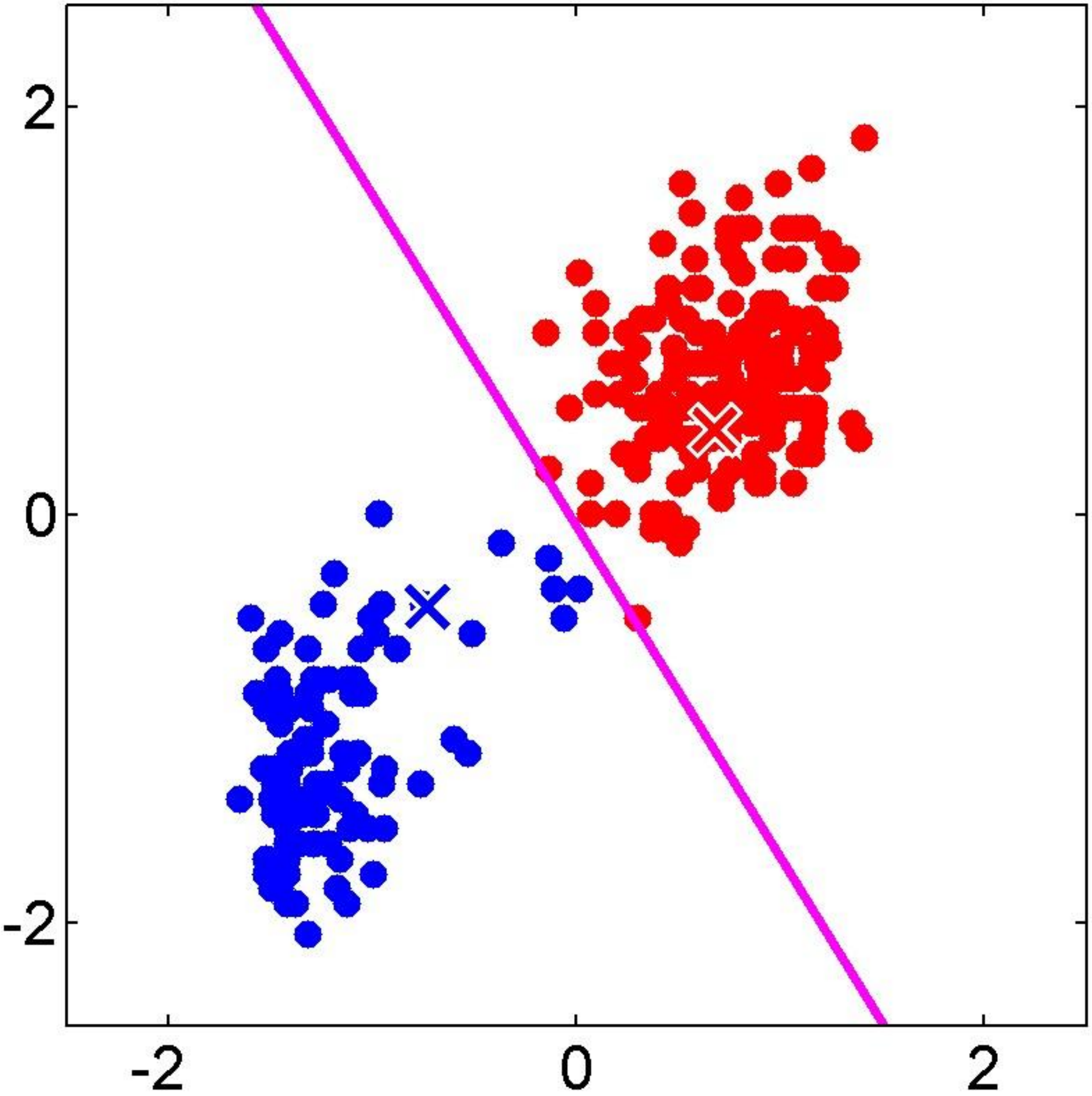
K-means Algorithm

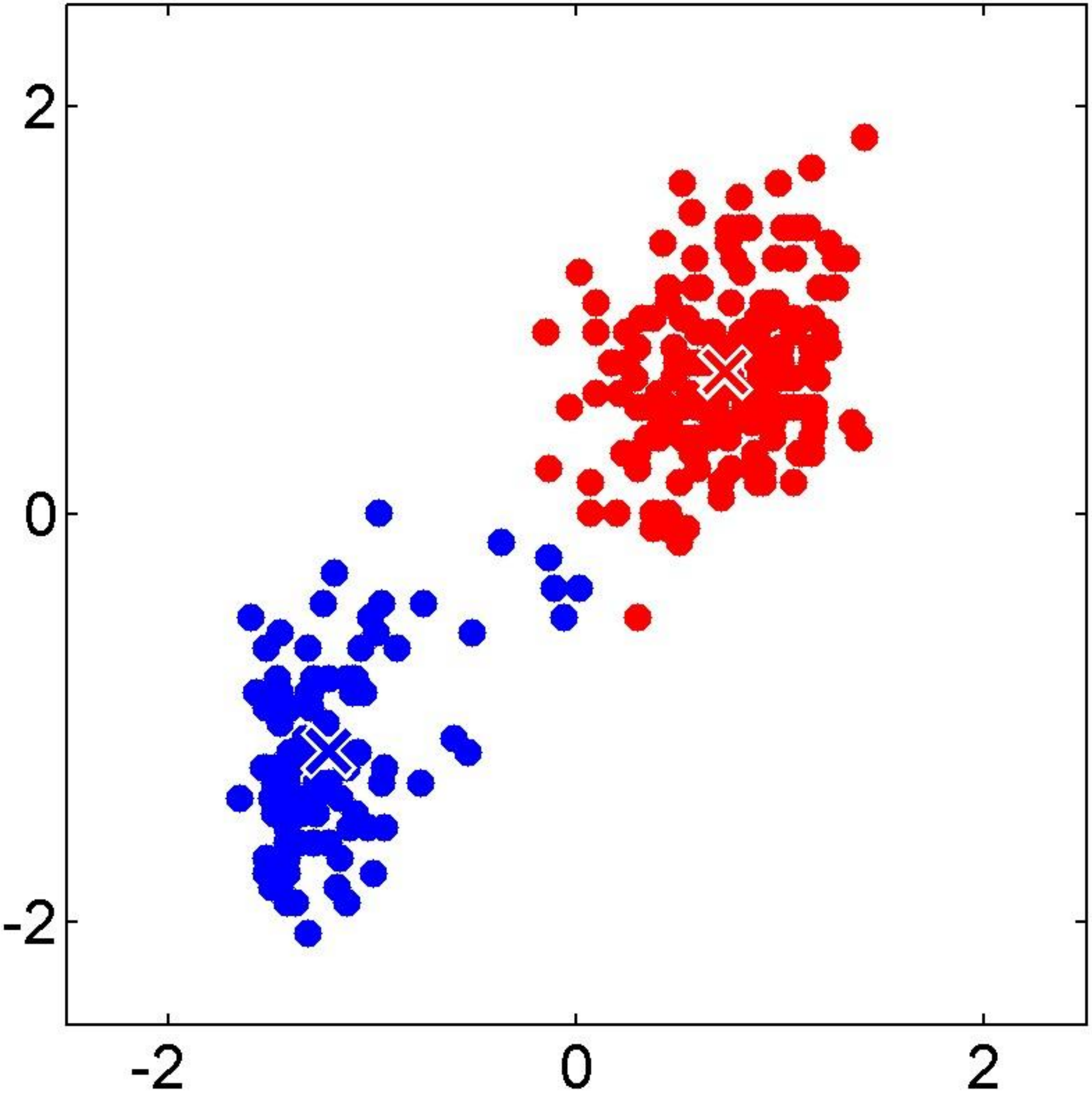
- Represent data set by K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
  - E-step: assign each data point to nearest prototype
  - M-step: update prototypes to be the cluster means
- Simplest version is based on Euclidean distance
  - re-scale Old Faithful data

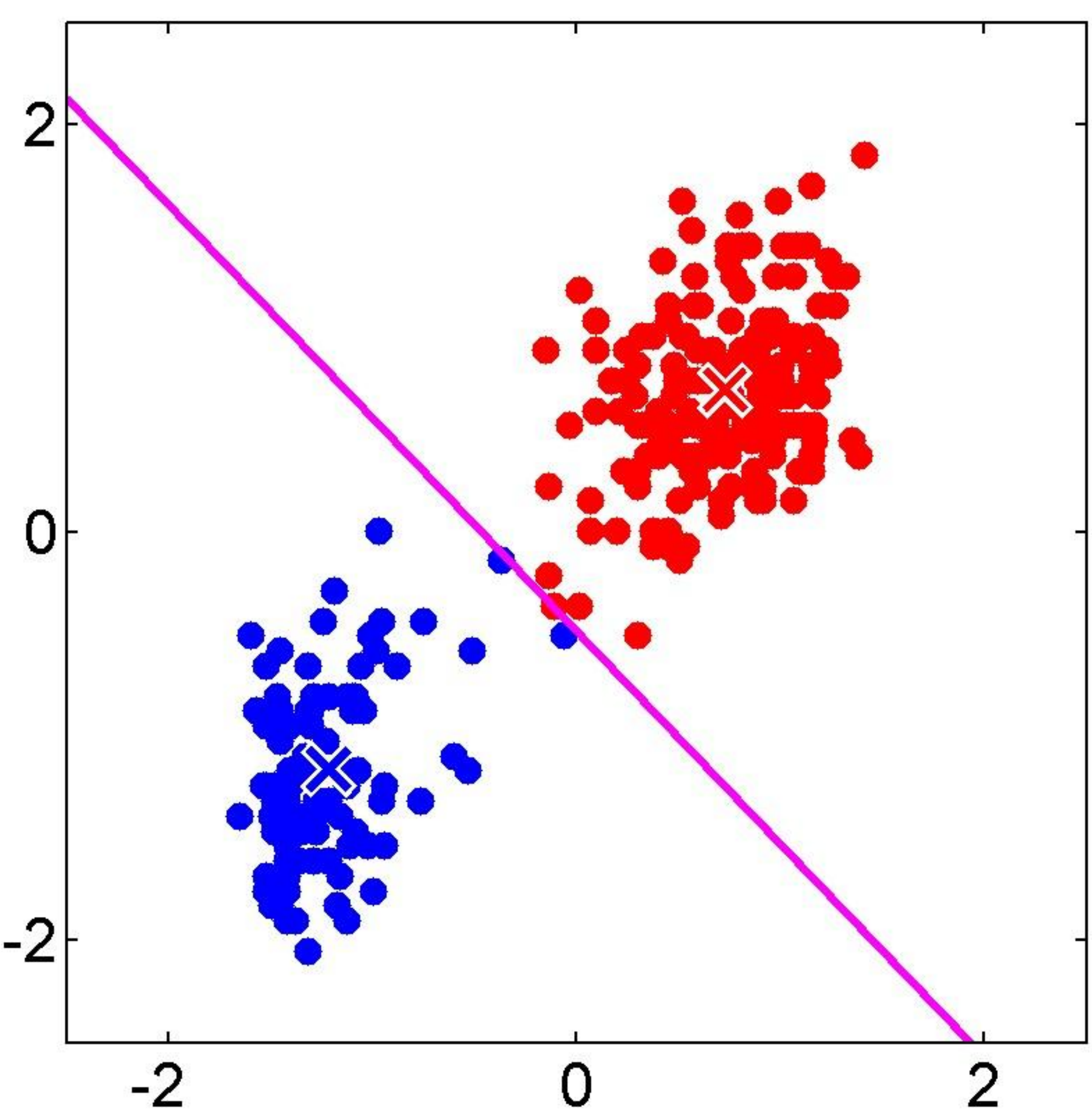

2
0
-2
-2
0
2

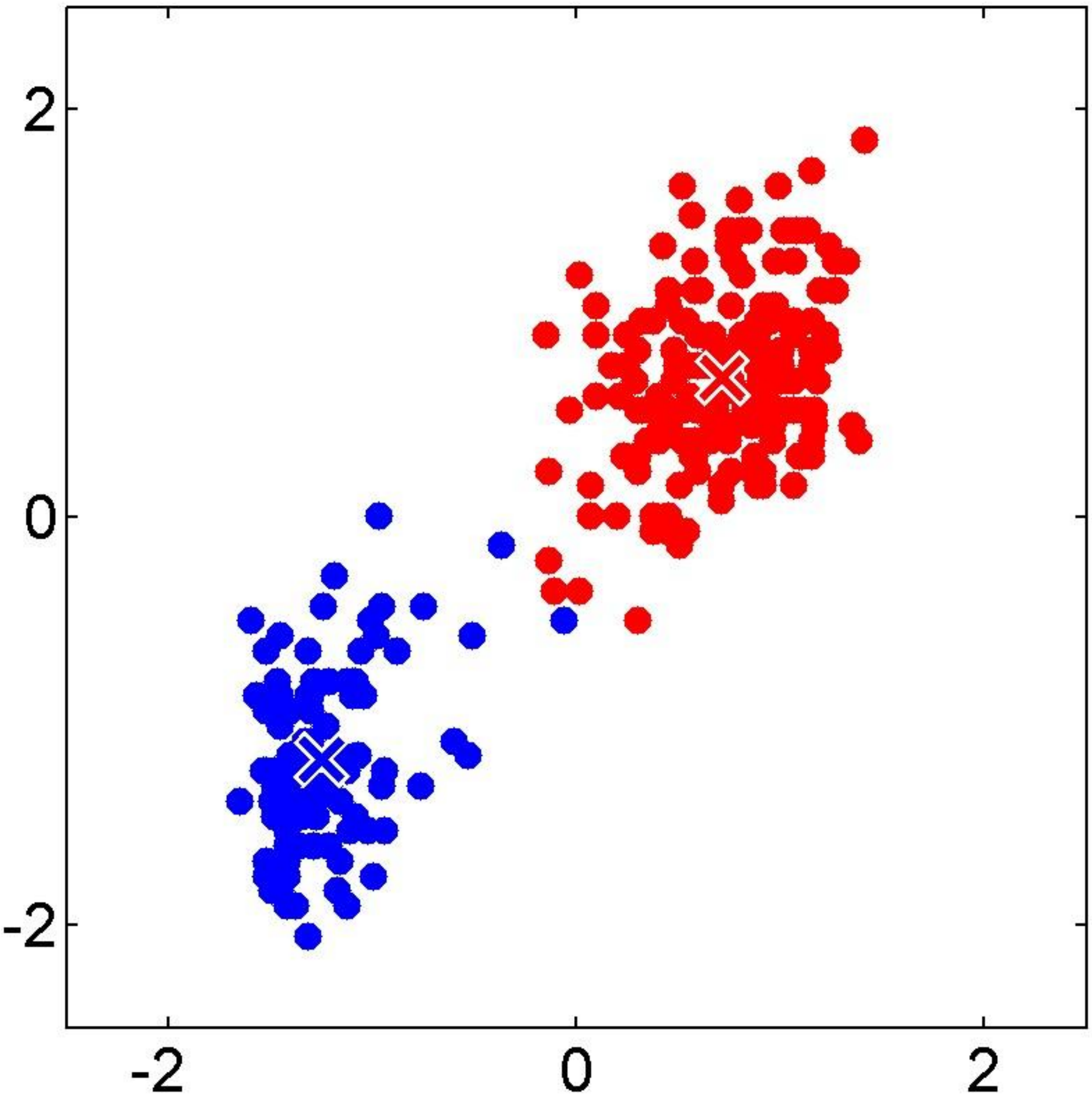

2
0
-2
-2
0
2

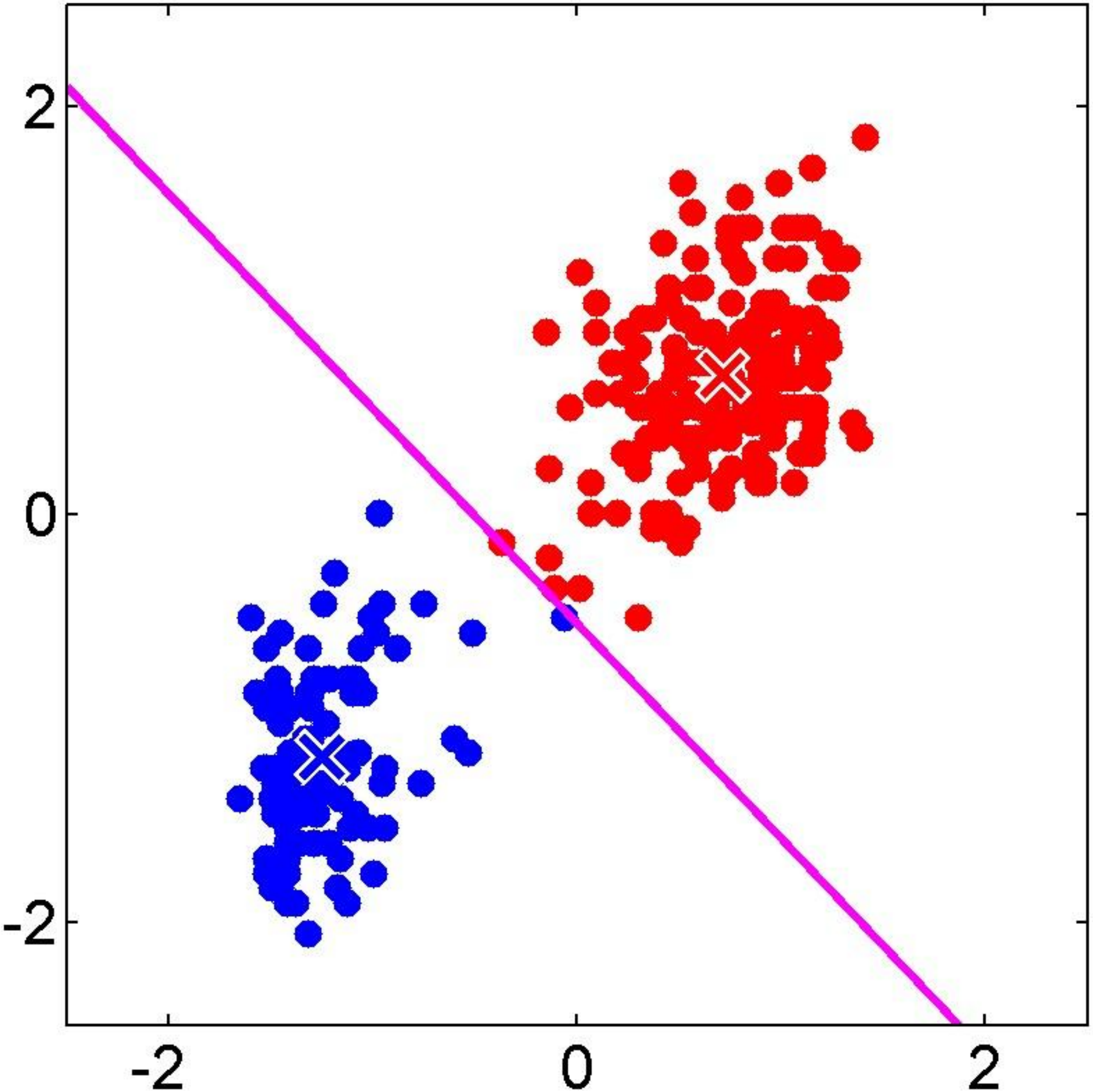

2
0
-2
-2
0
2

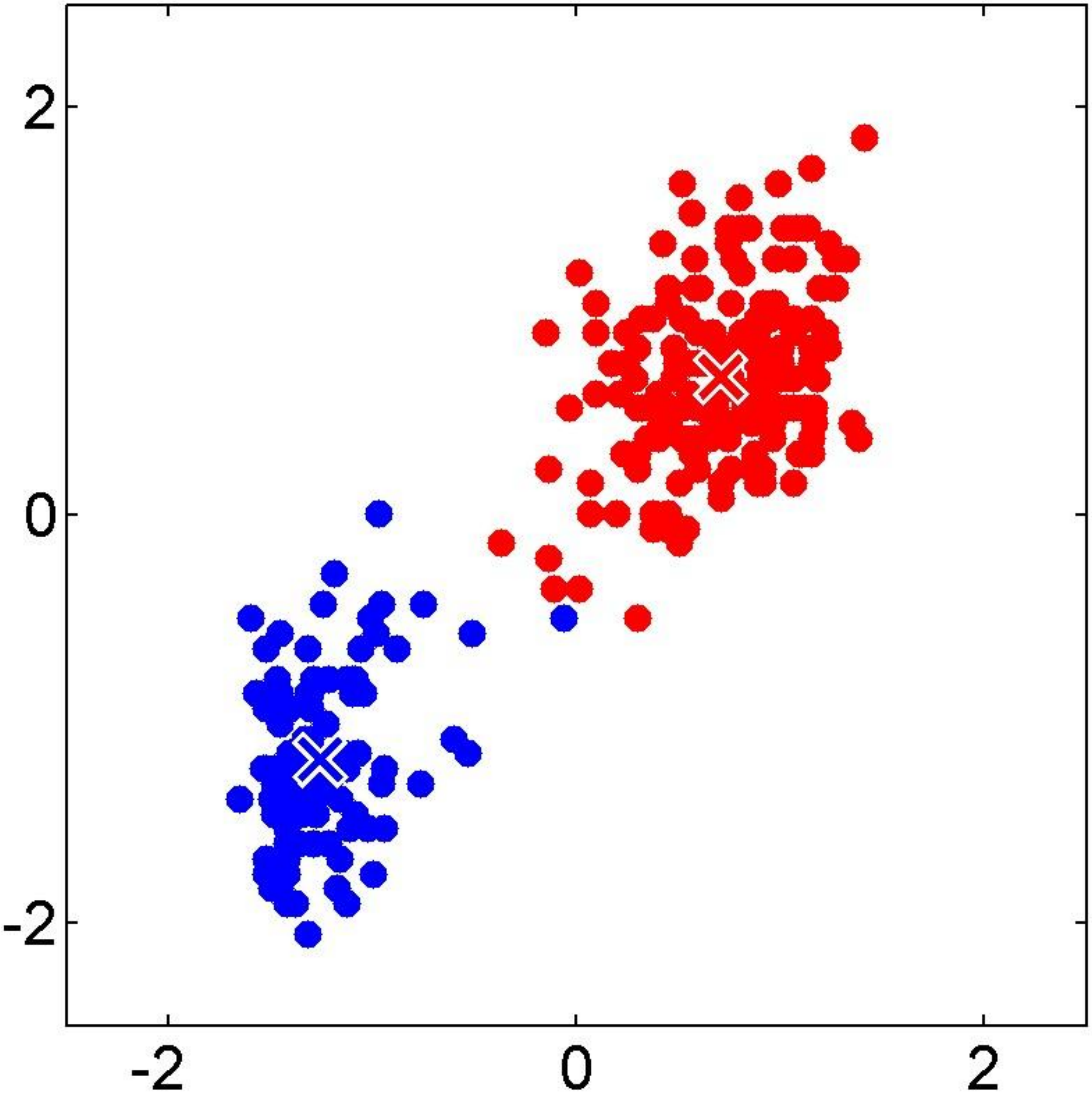

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

Responsibilities

- *Responsibility matrix:* assign data points to clusters

  $$r_{nk} \in \{0, 1\}$$

  such that

  $$\sum_k r_{nk} = 1$$

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

K-means Cost Function

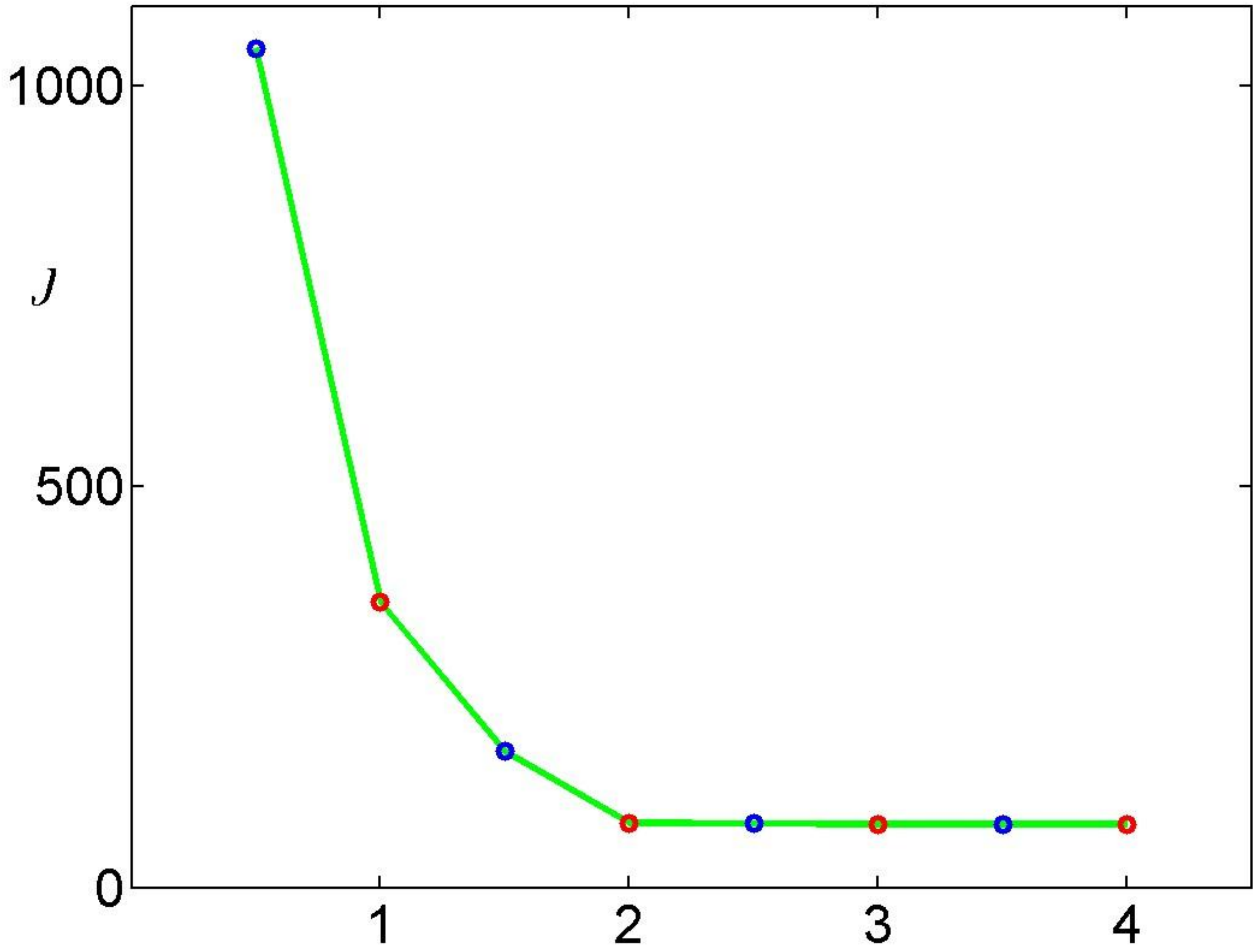
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \underset{\text{responsibilities}}{r_{nk}} \| \overset{\text{data}}{\mathbf{x}_n} - \underset{\text{prototypes}}{\boldsymbol{\mu}_k} \|^2$$

Minimizing the Cost Function

- E-step: minimize Jw.r.t. r_{nk}
  - assigns each data point to nearest prototype
- M-step: minimize J w.r.t $\boldsymbol{\mu}_k$
  - gives

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{kn}\mathbf{x}_n}{\sum_n r_{kn}}$$

  - each prototype set to the mean of points in that cluster
- Convergence guaranteed since there is a finite number of possible settings for the responsibilities


J
1000
500
0
1
2
3
4

Limitations of K-means

- Hard assignments of data points to clusters – small shift of a data point can flip it to a different cluster
- Not clear how to choose the value of K
- Solution: replace ‘hard’ clustering of K-means with ‘soft’ probabilistic assignments
- Represents the probability distribution of the data as a *Gaussian mixture model*

Mixture of Gaussians

- Each class is a mixture of k Gaussians.
- Each gaussian has covariance, in addition to mean

The Gaussian Distribution

- Multivariate Gaussian

$$\mathcal{N}(\mathbf{x}|\underset{\text{mean}}{\boldsymbol{\mu}}, \underset{\text{covariance}}{\boldsymbol{\Sigma}}) = \frac{1}{(2\pi|\boldsymbol{\Sigma}|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

- Define precision to be the inverse of the covariance

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$$

- In 1-dimension

$$\tau = \frac{1}{\sigma^2}$$

Likelihood Function

- Data set

$$D = \{\mathbf{x}_n\} \quad n = 1, \ldots, N$$

- Assume observed data points generated independently

$$p(D|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Viewed as a function of the parameters, this is known as the *likelihood function*

Maximum Likelihood

- Set the parameters by maximizing the likelihood function
- Equivalently maximize the log likelihood

$$\begin{aligned}\ln p(D|\boldsymbol{\mu},\boldsymbol{\Sigma}) \;&=\; -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\ln(2\pi) \\ &\quad -\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n-\boldsymbol{\mu})\end{aligned}$$

Maximum Likelihood Solution

- Maximizing w.r.t. the mean gives the *sample mean*

$$\boldsymbol{\mu}_{\mathsf{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\boldsymbol{\Sigma}_{\mathsf{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{\mathsf{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\mathsf{ML}})^{\mathsf{T}}$$

- Maximizing w.r.t covariance gives the *sample covariance*

Gaussian Mixtures

- Linear super-position of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

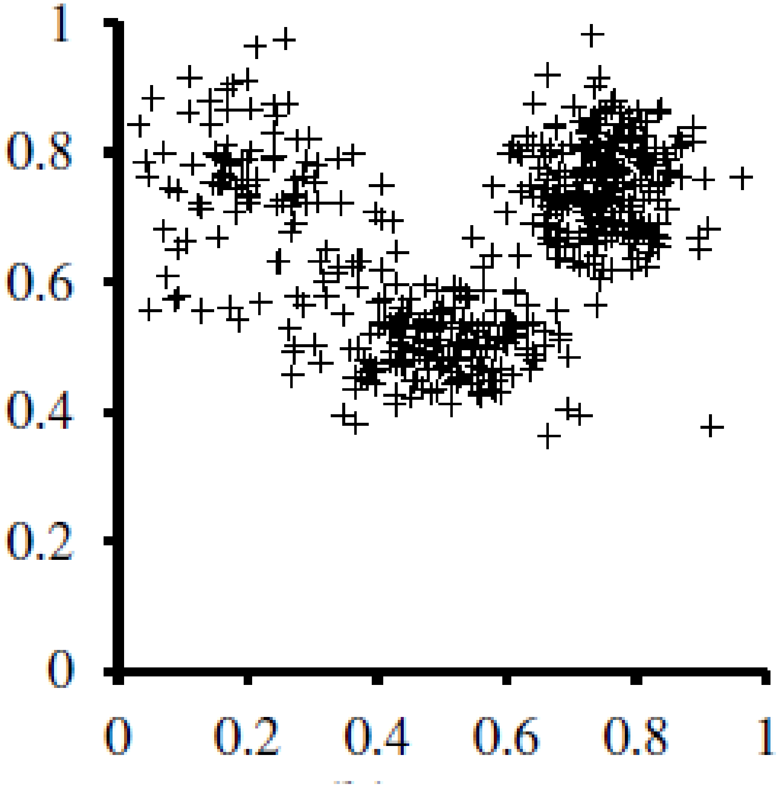
- Normalization and positivity require

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

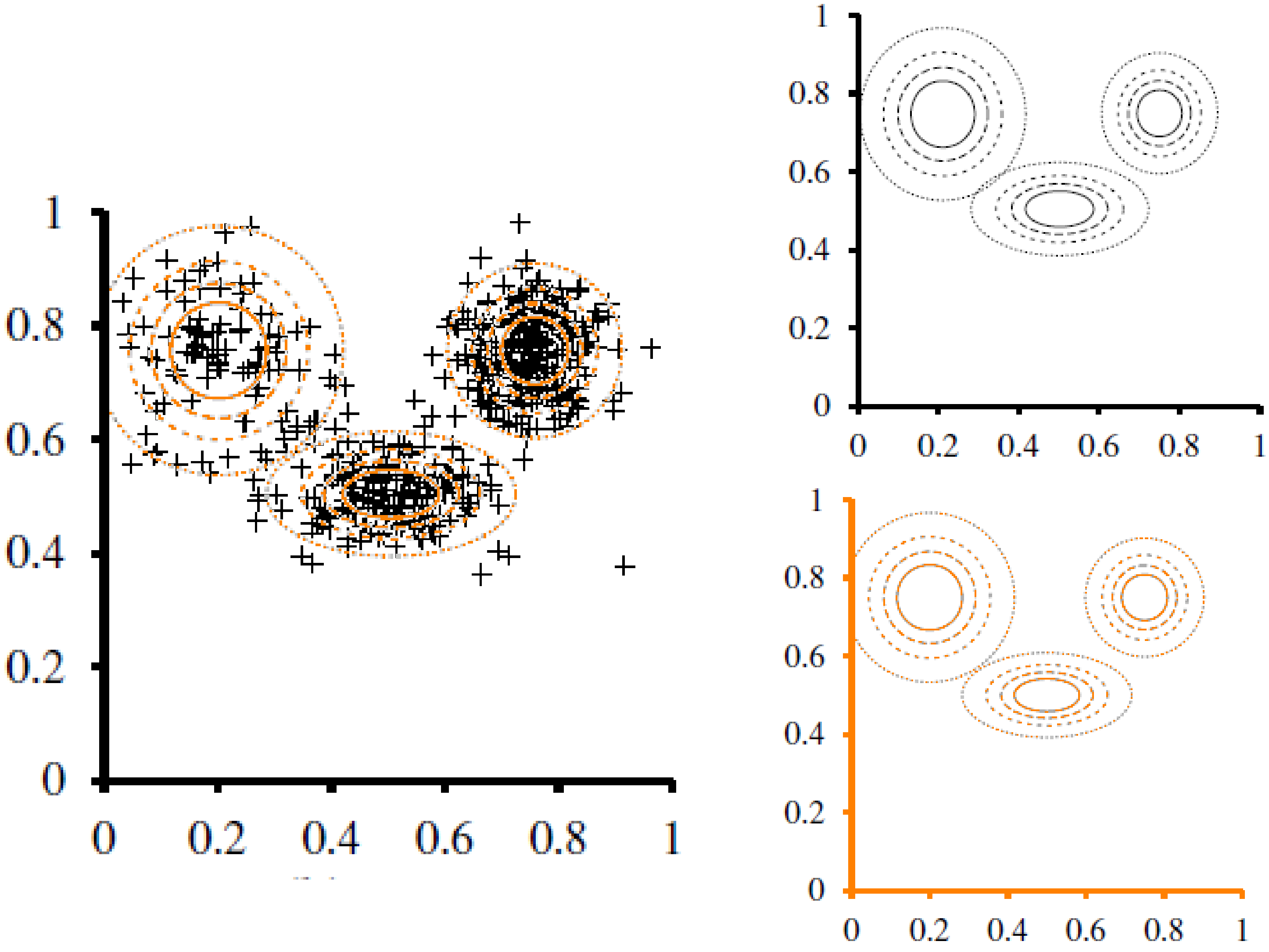
- Can interpret the mixing coefficients as prior probabilities

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k)$$

Single Gaussians model



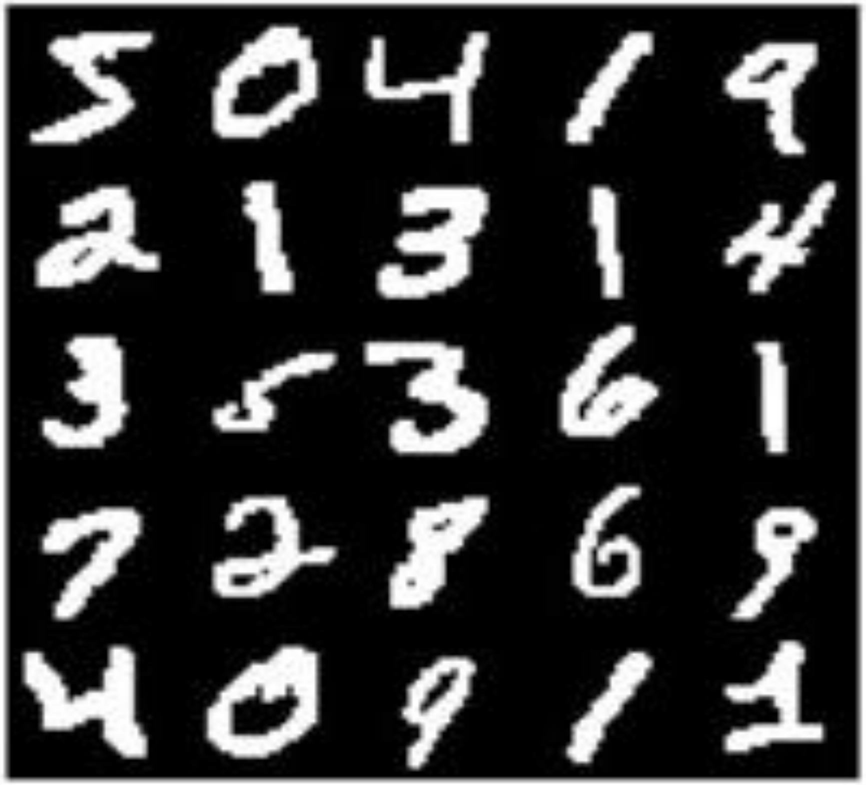
Single Gaussians model



The value of a good metric

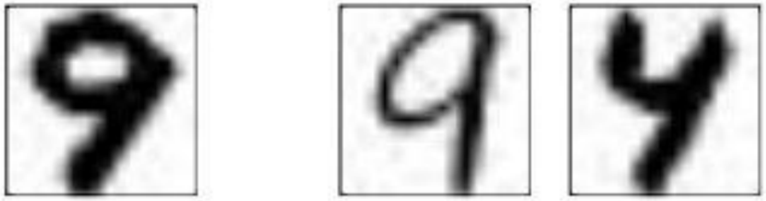
Learning representations: Handwritten Digits

handwrittten numerals (MNIST)

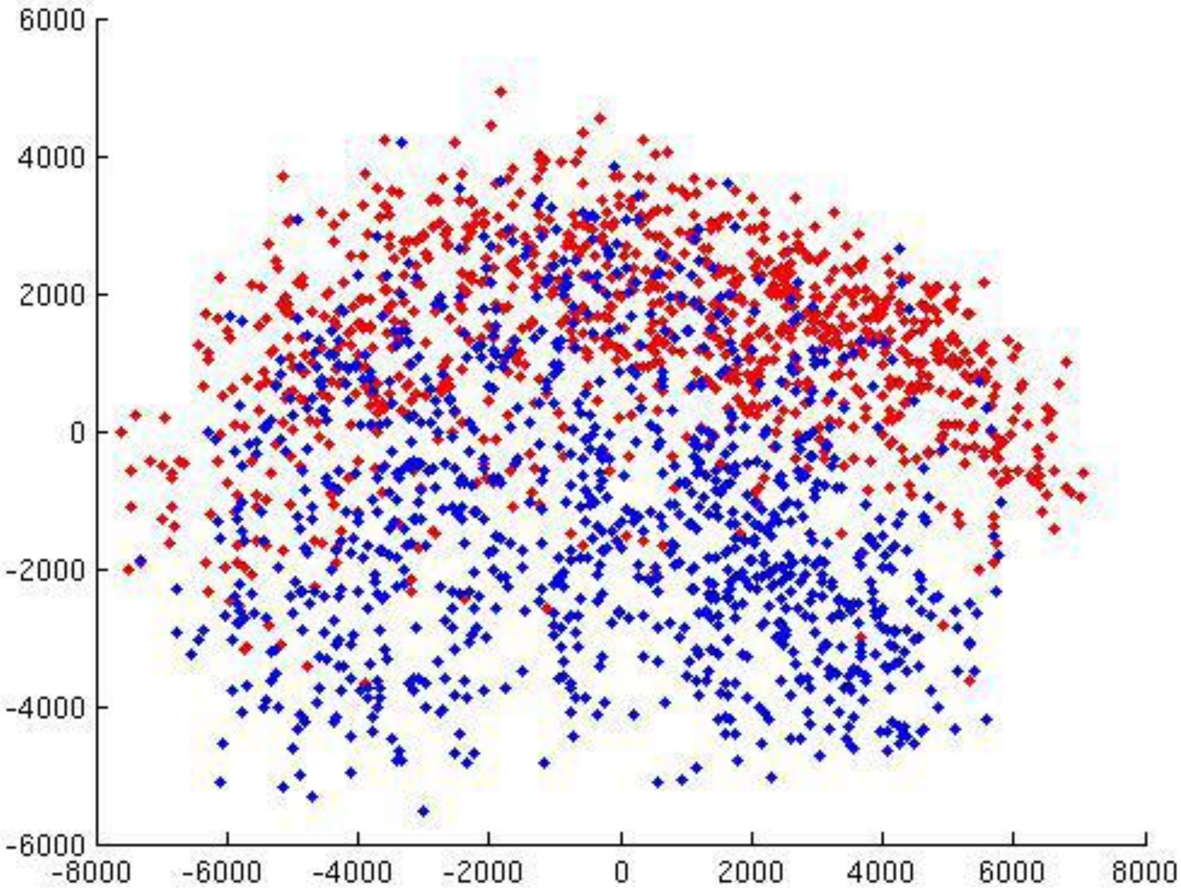


Modified NIST digits database: 60K + 10K 28x28 images

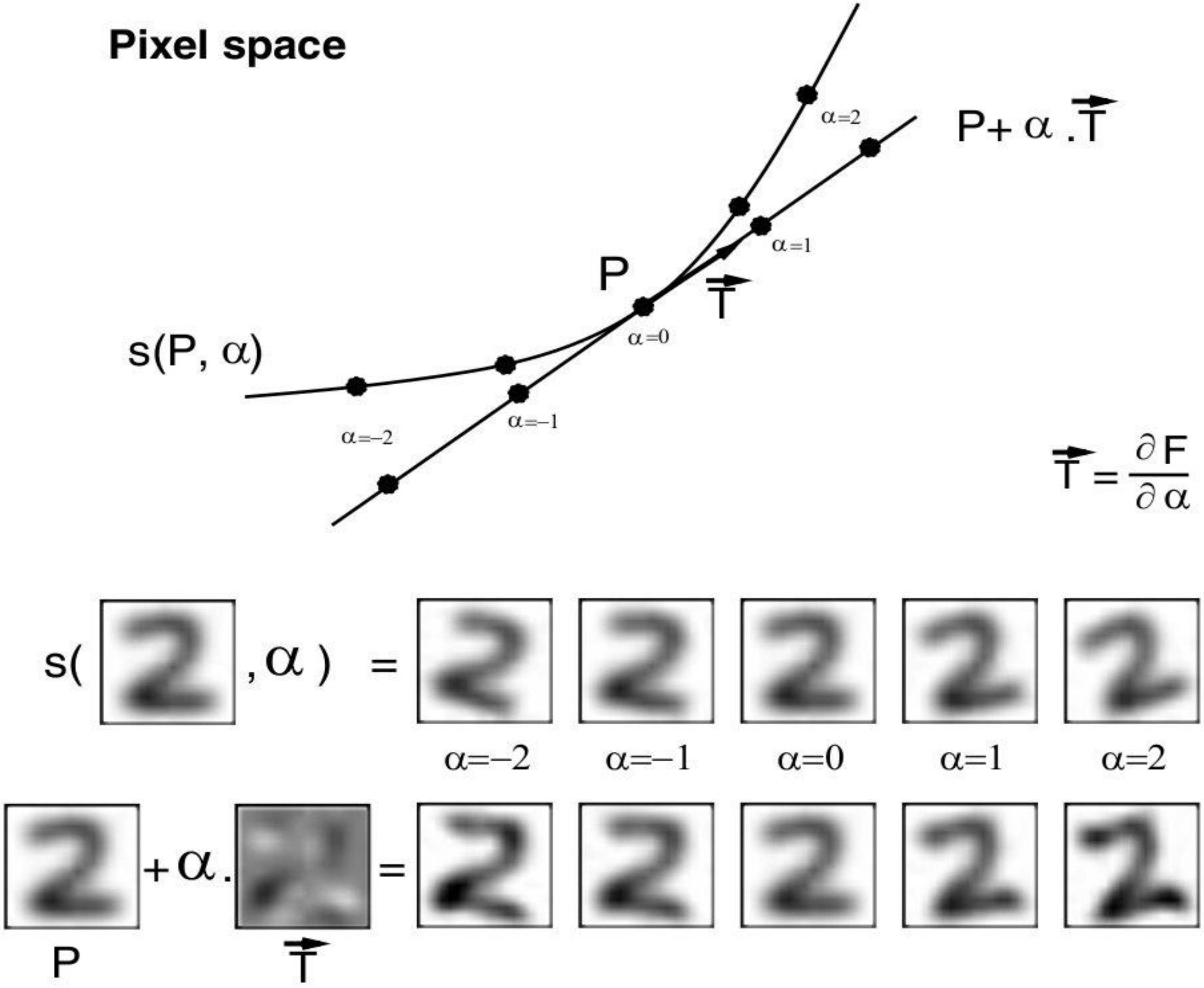
Importance of choosing a metric



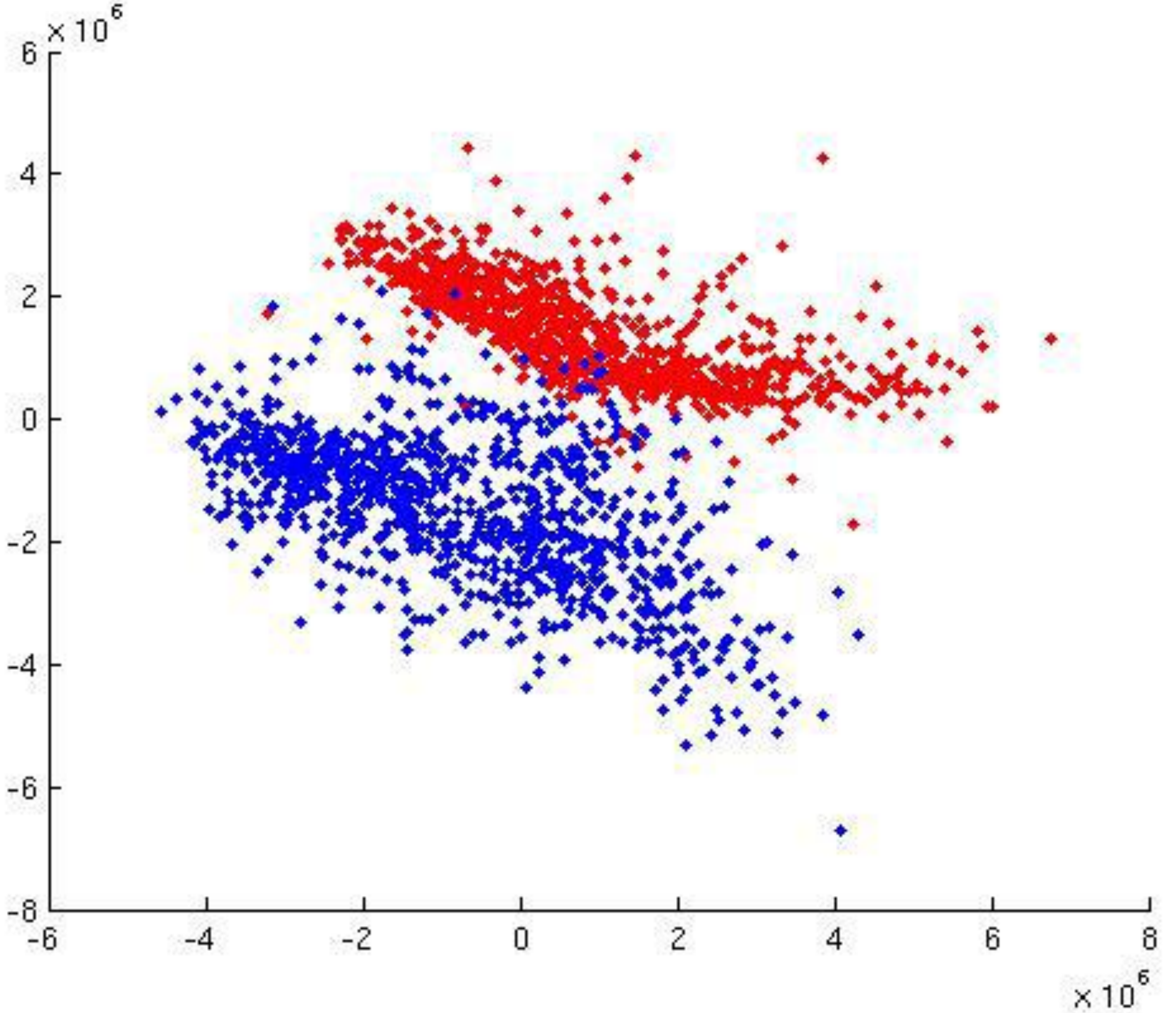
Manifold mapping with Euclidean Distance

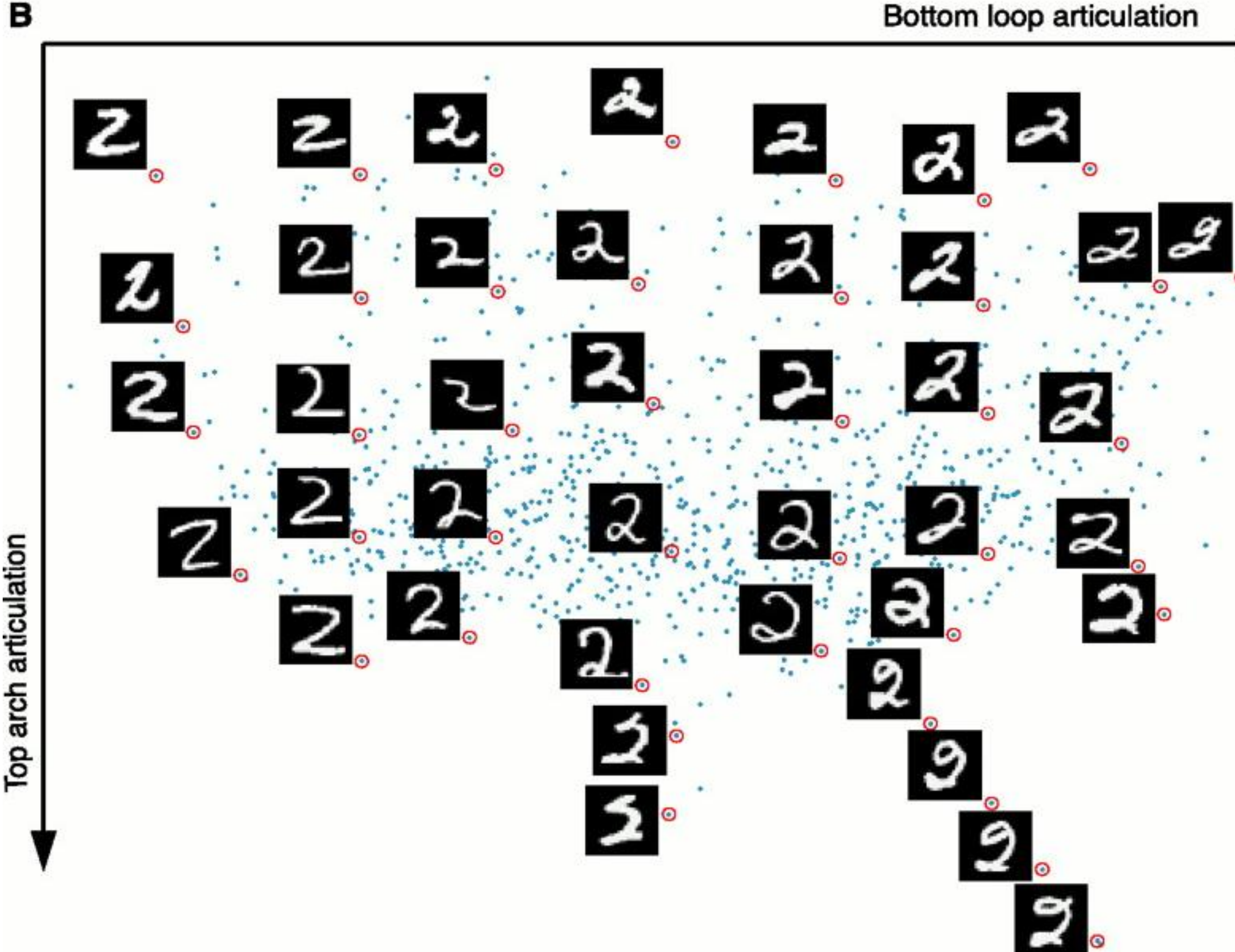


“tangent distance”

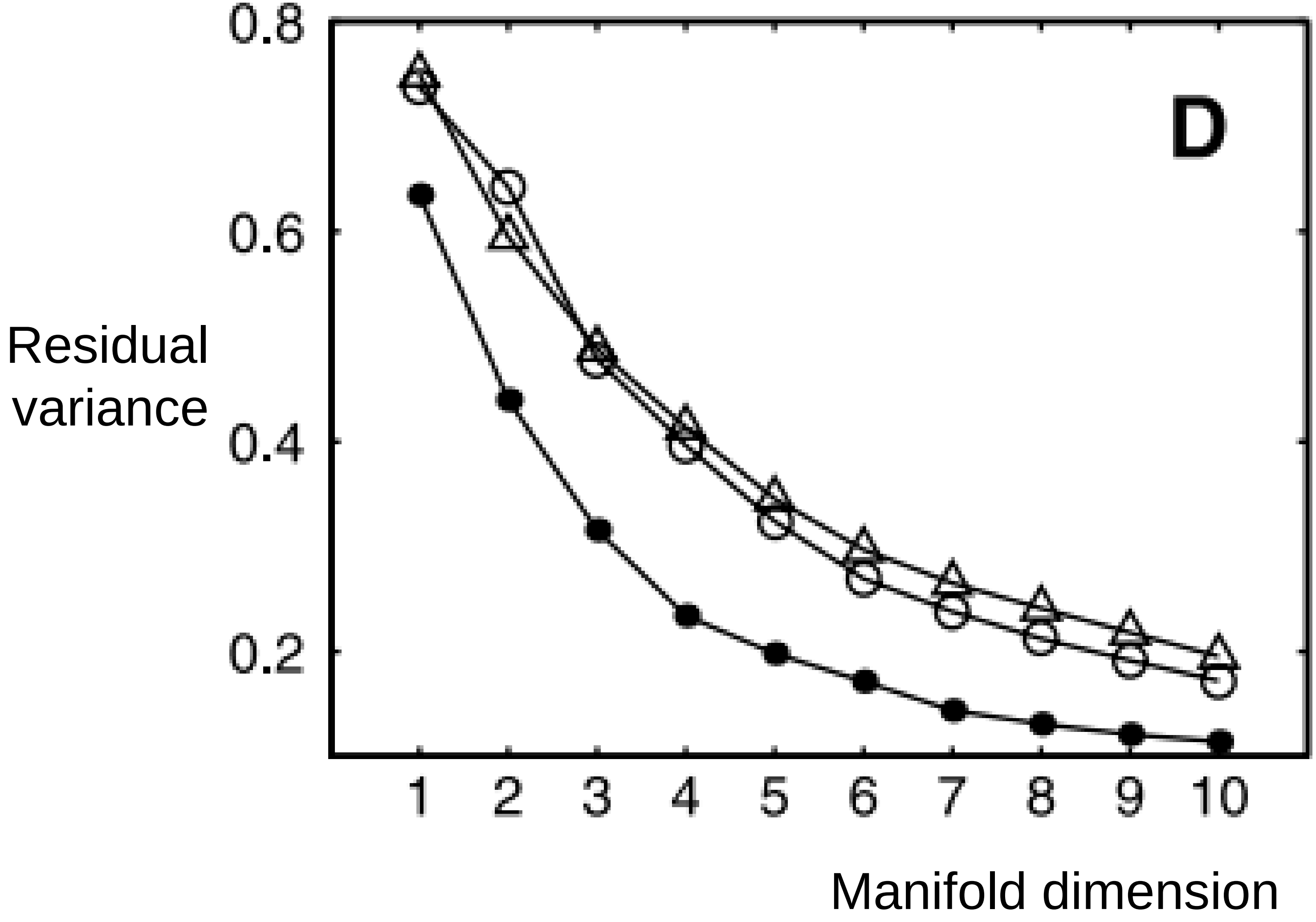


Manifold mapping with Euclidean Distance




B
Bottom loop articulation
Top arch articulation

Dimensionality: handwritten digits


Residual variance
D
Manifold dimension