Dimension Reduction

Prof. Fei Sha

Dept. of Computer Science
U. of Southern California

IPAM Summer School on Computer Vision
Aug. 6 2013

Dimension reduction

- **Question:**

  How can we discover low dimensional structure in high dimensional data?

- **Motivations:**

  Exploratory data analysis & visualization

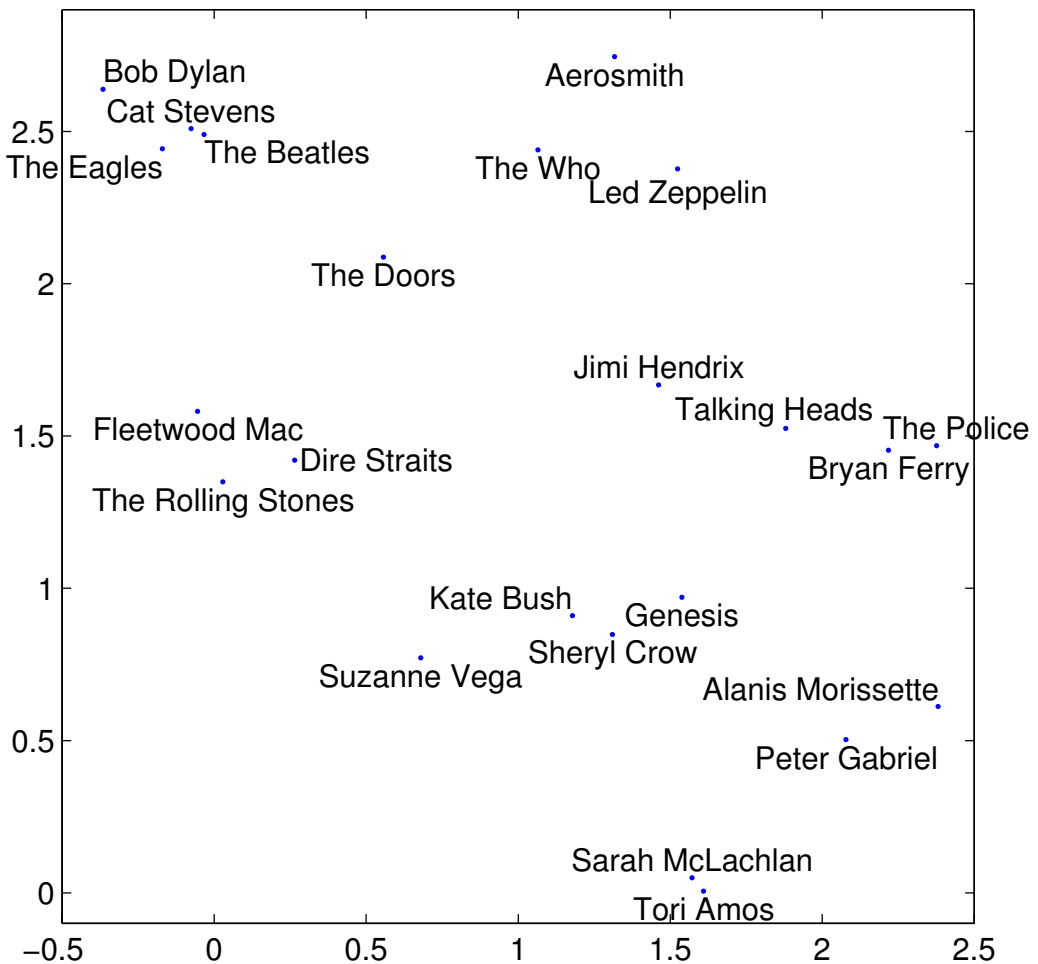
  Data reduction for faster algorithms

  Robust statistical modeling

  Learning new features

Cool applications

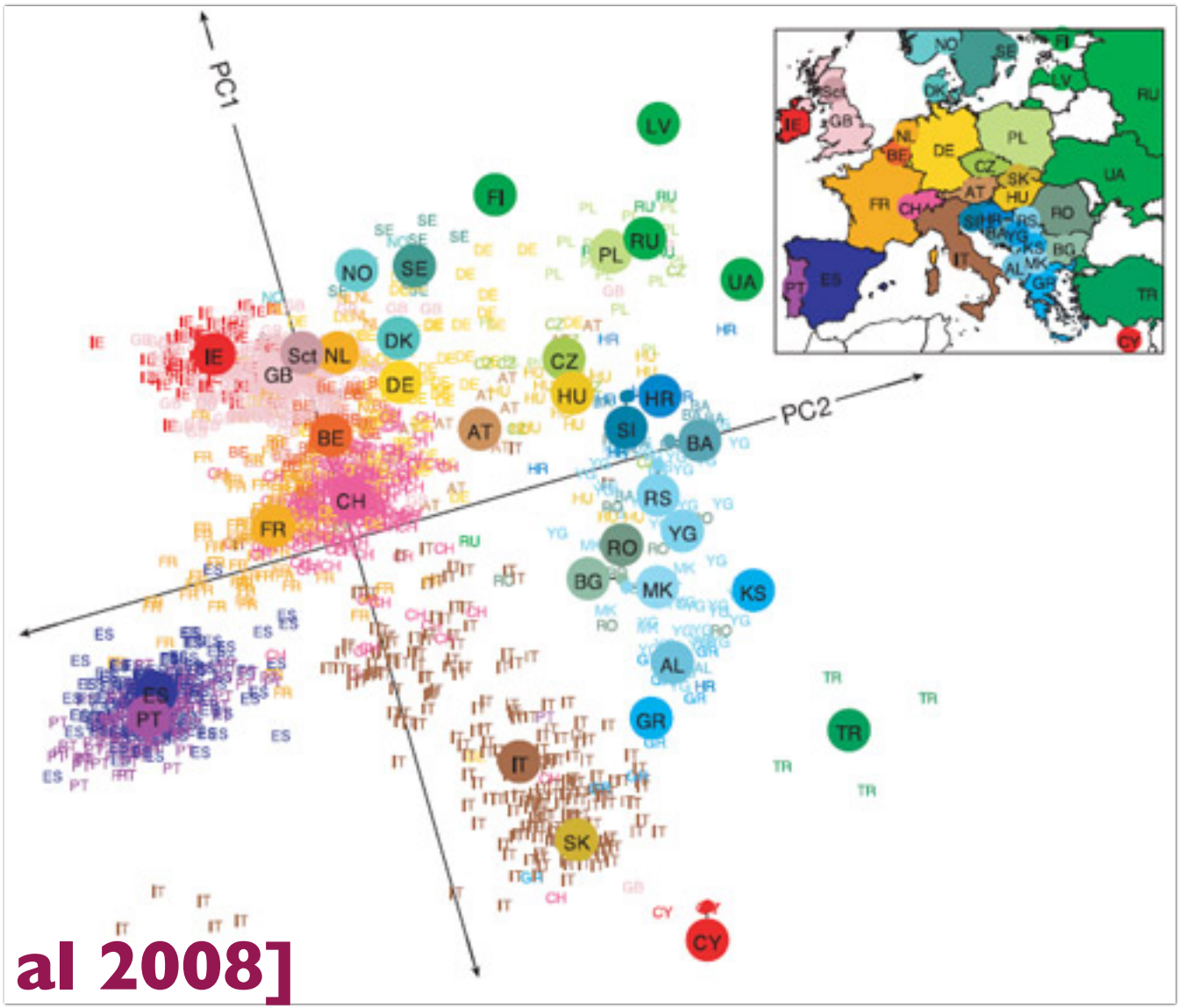
- **Embedding music similarity graph**



[Platt, 2004]

Cool applications

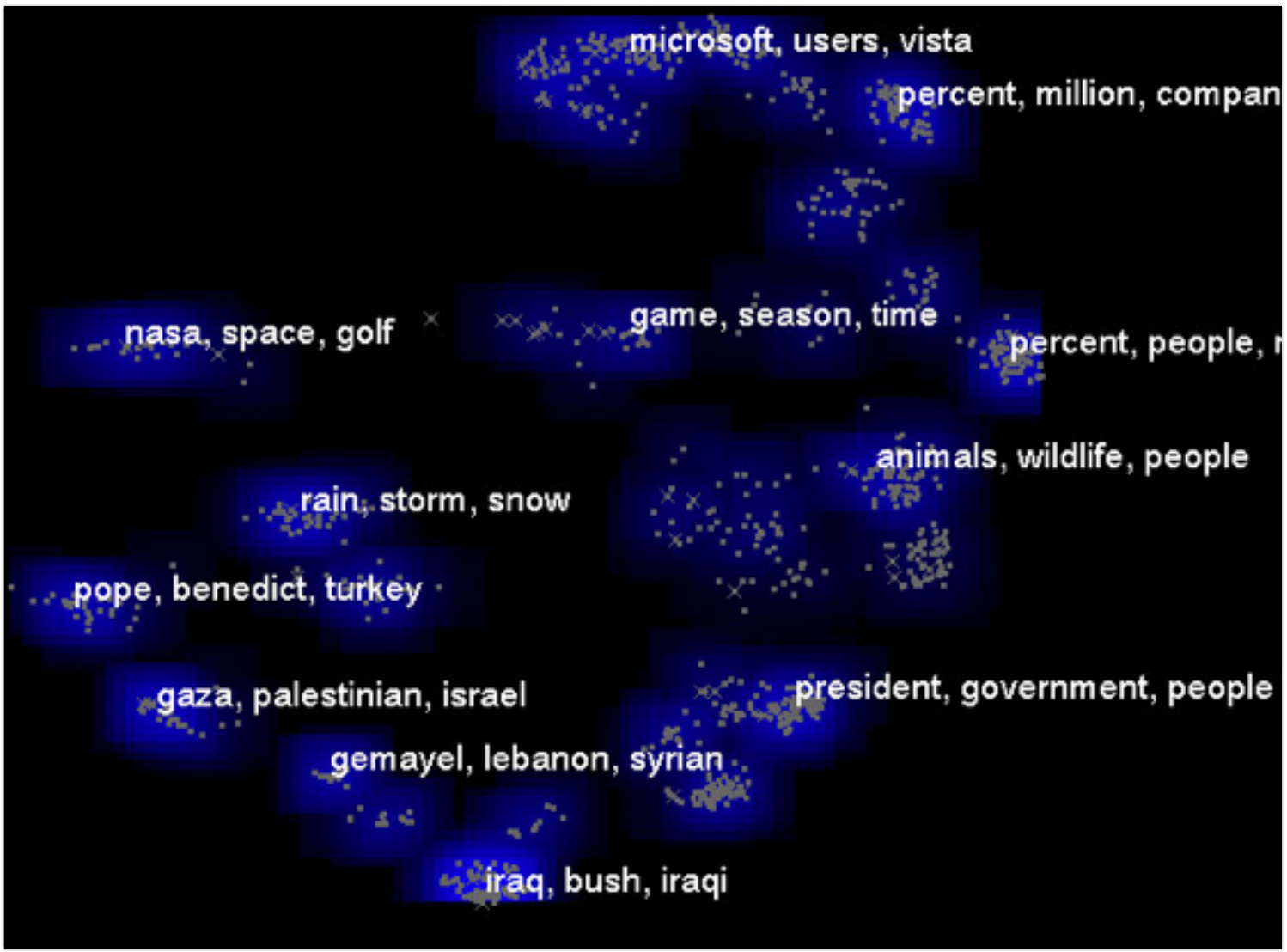
- Gene mirrors Europe geography



[Novembre et al 2008]

Cool applications

- Visualize how papers are clustered



General framework

- **Inputs**

$$\{\boldsymbol{x}_i \in \mathbb{R}^D, i = 1, 2, \ldots, N\}$$

- **Outputs**

$$\{\boldsymbol{y}_i \in \mathbb{R}^M, i = 1, 2, \ldots, n\}$$

- **Constraint**

$$M \ll D$$

- **Goals**

What properties of x should we preserve in y?

Many many algorithms

- **Linear**

  PCA, MDS, random projection, CCA

- **Nonlinear**

  Neural networks, GTM

- **Nonparametric**

  Kernelized version of all linear methods

  Manifold learning

  Gaussian Process

Many many algorithms

- **Linear**

  PCA, MDS, random projection, CCA

- **Nonlinear**

  Neural networks, GTM

- **Nonparametric**

  Kernelized version of all linear methods

  Manifold learning

  Gaussian Process

Outline of this lecture (Part 1)

- **Central questions**

  Discover linear structures

  Generalize to nonlinear structures

- **Algorithms**

  Principal Component Analysis (PCA)

  Metric Multidimensional Scaling (MDS)

  Neural Network Autoencoders

  Kernel PCA

Outline of next lecture (Part 2)

- **Central questions**

  Discover nonlinear structures

  Leverage tractable computation

- **Algorithms**

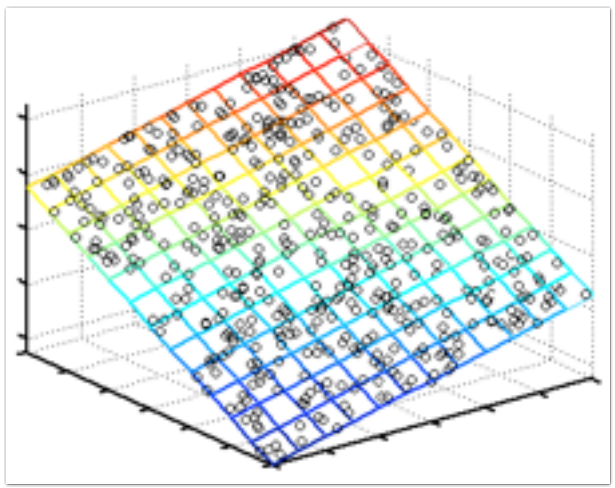
  Isomap

  Locally linear embedding (LLE)

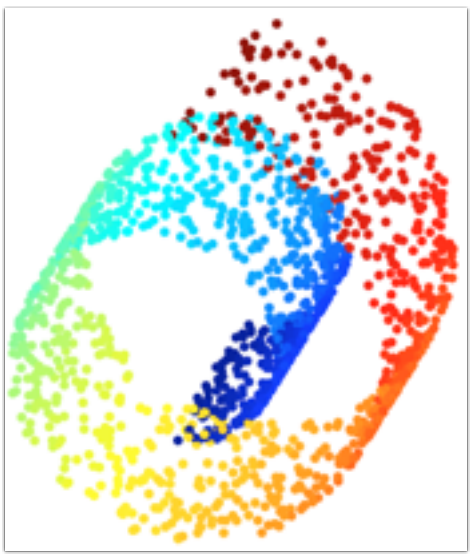
  Maximum variance unfolding (MVU)

Theme: linear vs nonlinear

Linear structure



nonlinear structure



Which one is easier to identify?

Discover linear structures

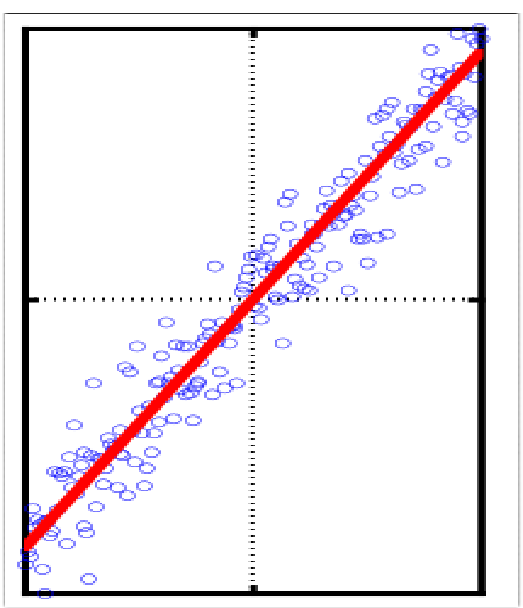
I. Principal Component Analysis (PCA)

Principal components analysis

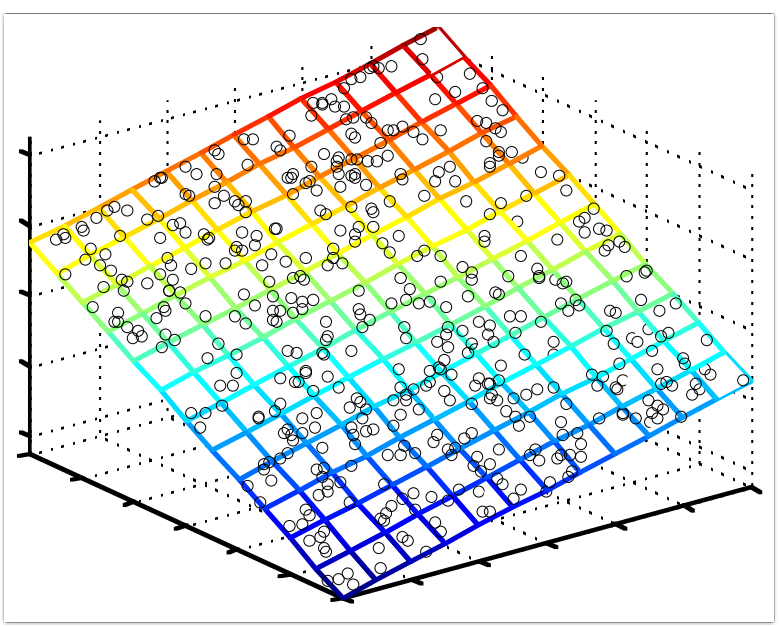
Does the data mostly lie in a subspace?

If so, what is its dimensionality?

D = 2
M = 1



D = 3
M = 2



The framework of PCA

- **Assumptions**

  Centered inputs

$$\sum_i \boldsymbol{x}_i = \mathbf{0}$$

  Linear projection into subspace

$$\boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

$$\boldsymbol{U} \in \mathbb{R}^{D \times M} \quad \textbf{and} \quad U^{\mathrm{T}} U = I$$

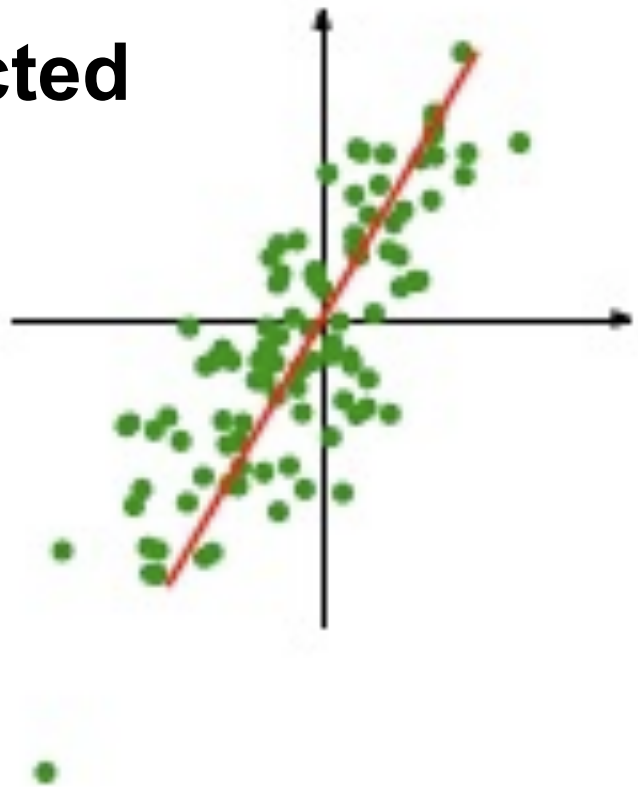
- **Objective**

  maximize variances in y as much as possible

The 1st principal component (PC)

- **Objective**

  maximize variance of points projected on the line



- **Why maximize variance?**

  Zero variance

  ⇒ not interesting data pattern

  High variance

  ⇒ likely interesting data pattern

PCA: solution for the first PC

- **Variance**

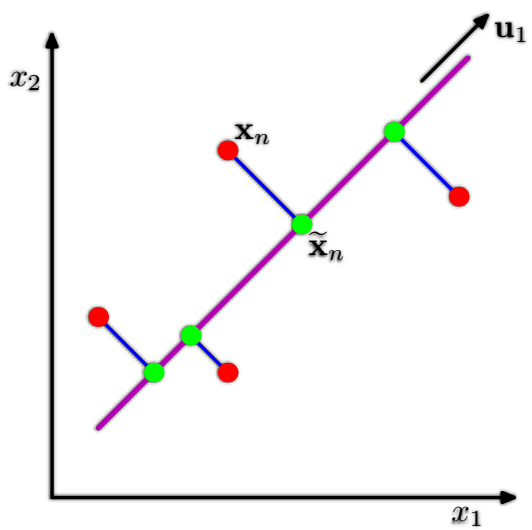
covariance matrix of the data

$$\sum_i y_i^2 = \sum_i \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{u}_1 = \boldsymbol{u}_1^{\mathrm{T}} \left\{ \sum_i \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \right\} \boldsymbol{u}_1 = \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{C}_x \boldsymbol{u}_1$$

- **Constrained optimization**

$$\max \quad \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{C}_x \boldsymbol{u}_1$$

$$\text{such that} \quad \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{u}_1 = 1$$



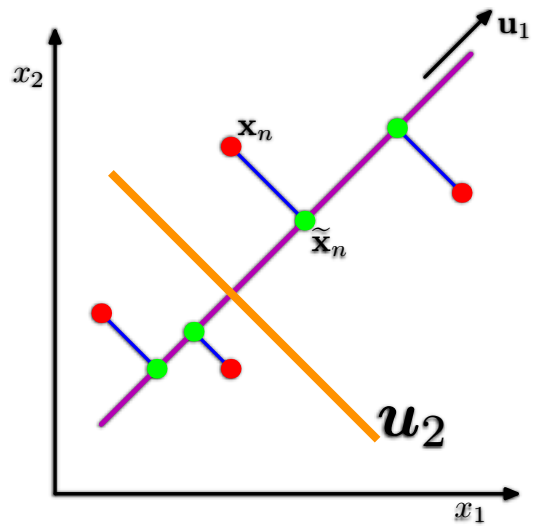
- **Closed-form solution**

Largest eigenvector of the covariance matrix

What about other components?

- **Variance of projections on the *k*-th PC**

$$\sum_i y_i^2 = \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{C}_x \boldsymbol{u}_k$$



- **Constrained optimization**

$$\max \quad \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{C}_x \boldsymbol{u}_k$$

$$\text{such that} \quad \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{u}_k = 1 \quad \textbf{and} \quad \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{u}_{k'} = 0$$

- **Closed-form solution**

The *k*-th largest eigenvector of the covariance matrix

Recipe

- Centralize inputs

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{N}\sum_i \boldsymbol{x}_i$$

- Compute covariance matrix

$$\boldsymbol{C}_x = \frac{1}{N}\sum_i \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}$$

- Decompose spectrally

$$\boldsymbol{C}_x \boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k$$

- Select top M pairs to form projection

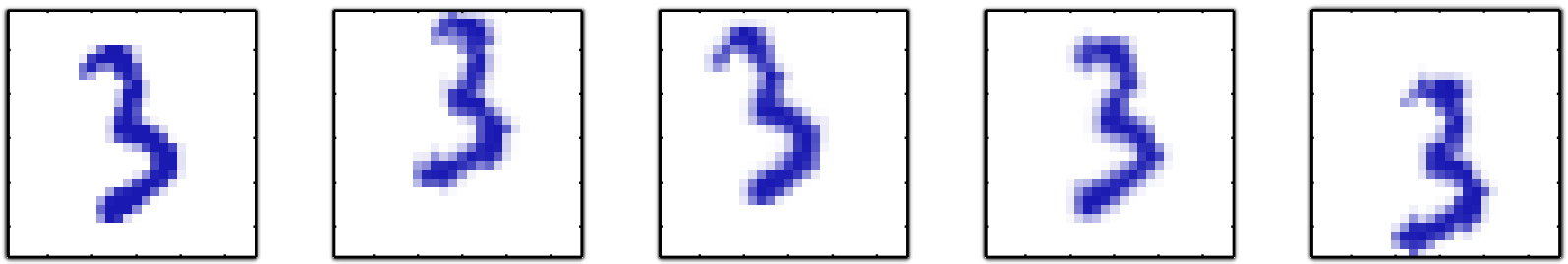
$$(\boldsymbol{u}_d, \lambda_d)$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$$

$$\boldsymbol{U} = (\boldsymbol{u}_1\ \boldsymbol{u}_2\ \cdots\ \boldsymbol{u}_M)$$

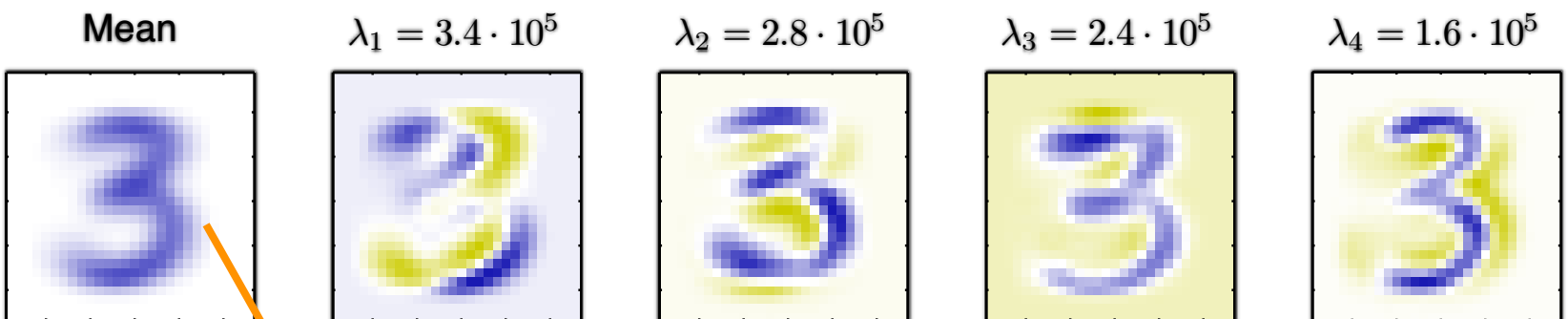
Examples of running PCA

- **Original data**



- **Principal components**

they look like blurred original images



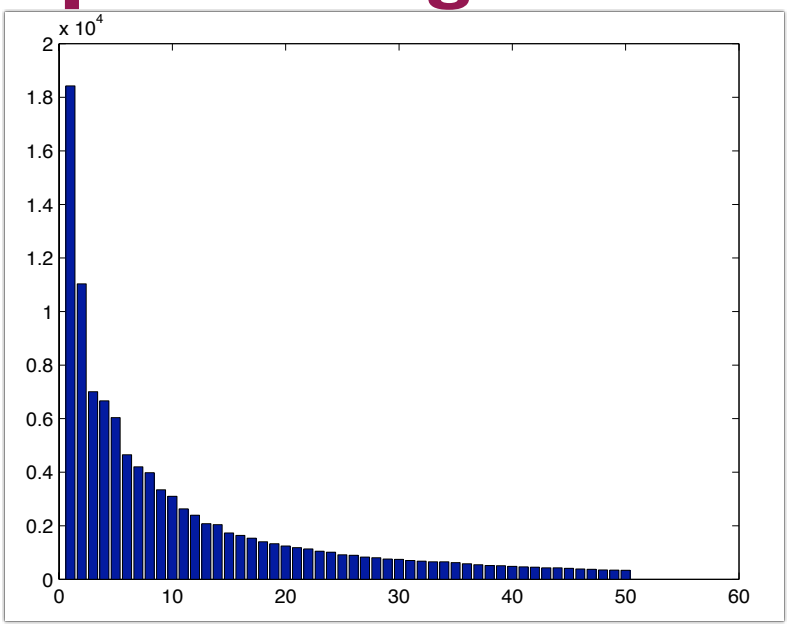
Used to centralize inputs

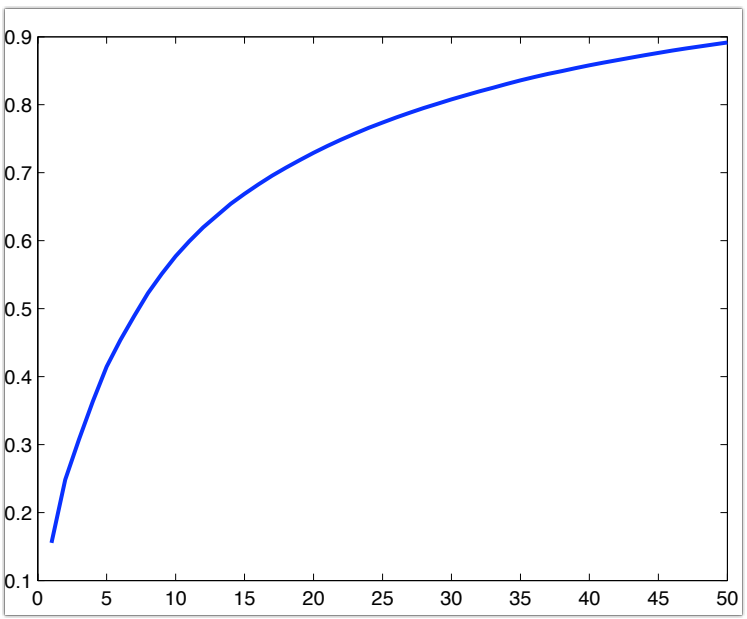
Choose the dimensionality

common choice is 95% or 90%

$$\frac{\sum_{d=1}^{M} \lambda_d}{\sum_{d=1}^{D} \lambda_d} \geq \text{Threshold}$$

bar-plot of eigenvalues





Interpreting PCA

- **Principal components**

  Basis of a subspace that maximally preserve variance in data

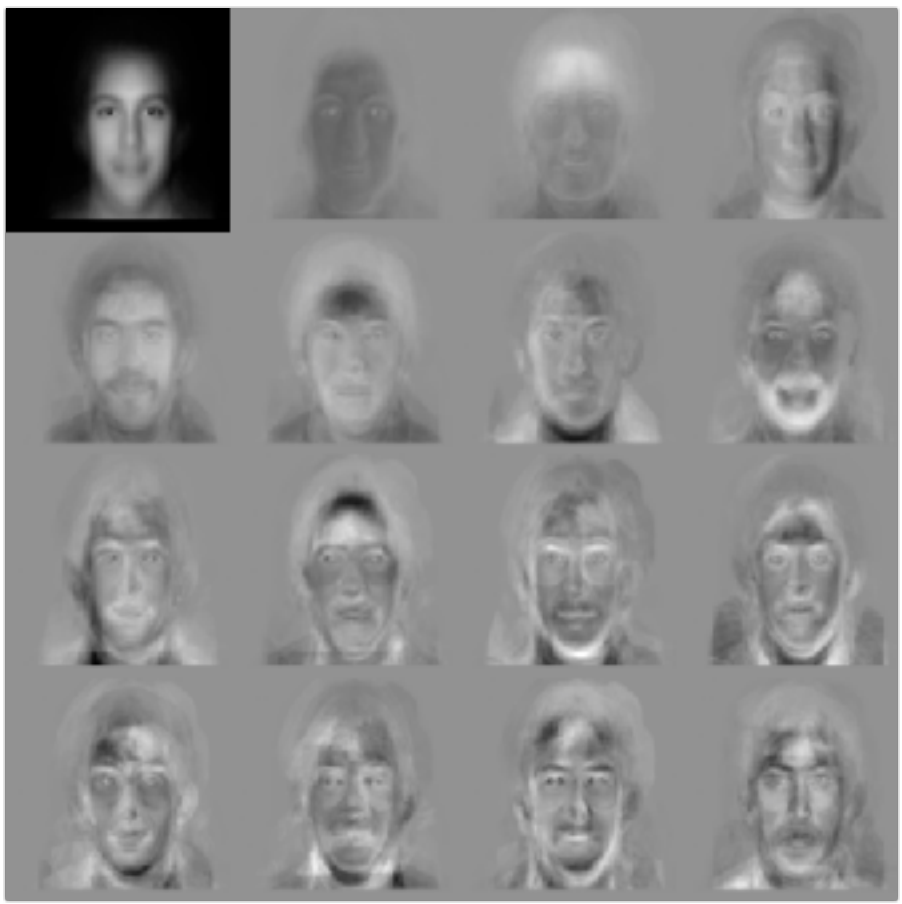
- **Eigenvalues**

  Variances of linearly projected inputs on the basis

- **Estimated dimensionality**

  Number of significant eigenvalues, whose sum explain the total variance in data above a threshold.

Another example



Eigenfaces (principal components) from 7562 face images

Top left image's low-dimensional representation is thus 15 numbers, each a projection of that image onto an eigenface

Meaning of low-dimensional representation?

- **An alternative objective**

  Linear projection (as before)

$$\boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

  Reconstruct the original input

$$\boldsymbol{x}'_i = \boldsymbol{U}\boldsymbol{y}_i = \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

  Minimize reconstruction error

$$\min \quad \sum_i \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|_2^2$$

Equivalent to maximizing variance

- **Maximize variance**

$$\max \quad \sum_i \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i \boldsymbol{x}_i \boldsymbol{U}^{\mathrm{T}}$$

- **Minimize reconstruction error**

$$\min \quad \sum_i \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|_2^2$$

- **Yield the same solution**

  U is the top eigenvectors of the covariance matrix

Interpreting PCA (again)

- **Principal components**

  Basis of a linear subspace that is closest to data points

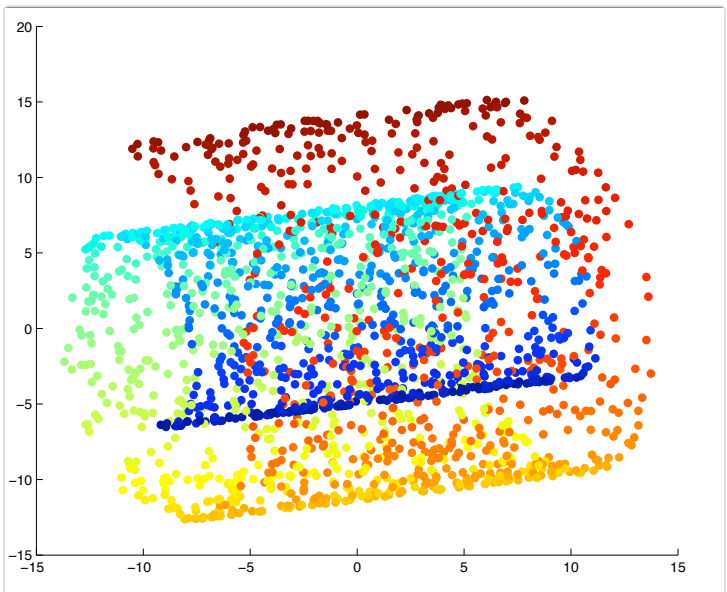
- **Low-dimensional representation**

  Coordinates (ie, combination coefficients of the basis) of the closet data point in that subspace

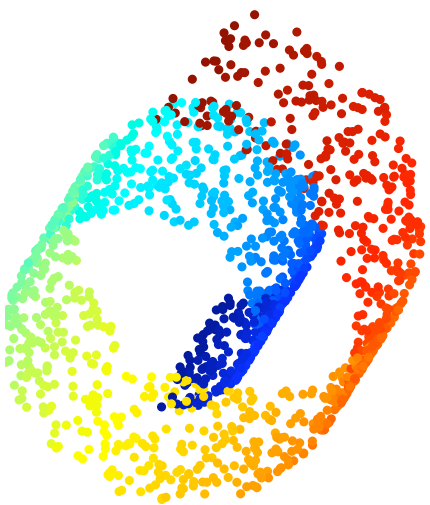
There are other interpretations!

Linear method is not enough!

Apply PCA

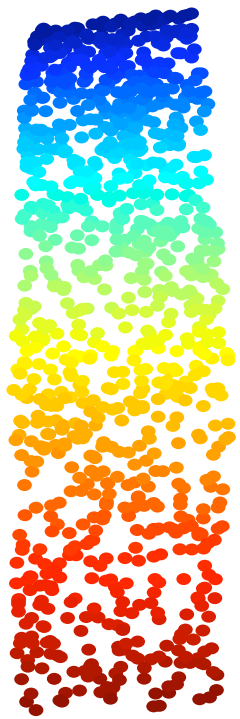


We need nonlinear mapping!



nonlinear projection

nonlinear reconstruction



Discover nonlinear structures

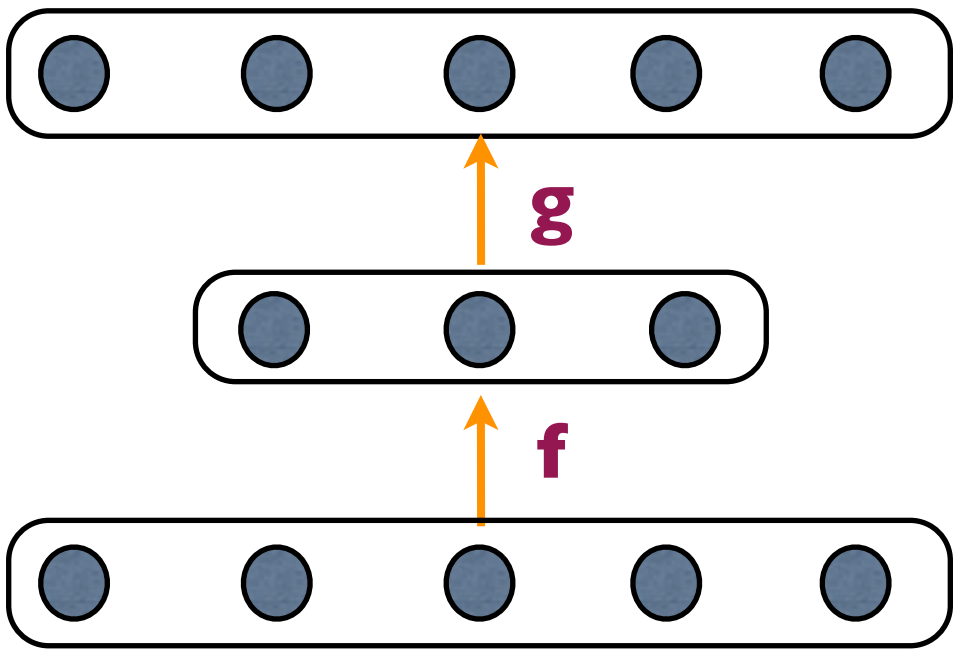
1. Neural networks autoencoders

Autoencoder

- **Neural network**

  Implement nonlinear functions for both projection and reconstruction

  Reconstruct inputs at output



- **Minimize reconstruction error**

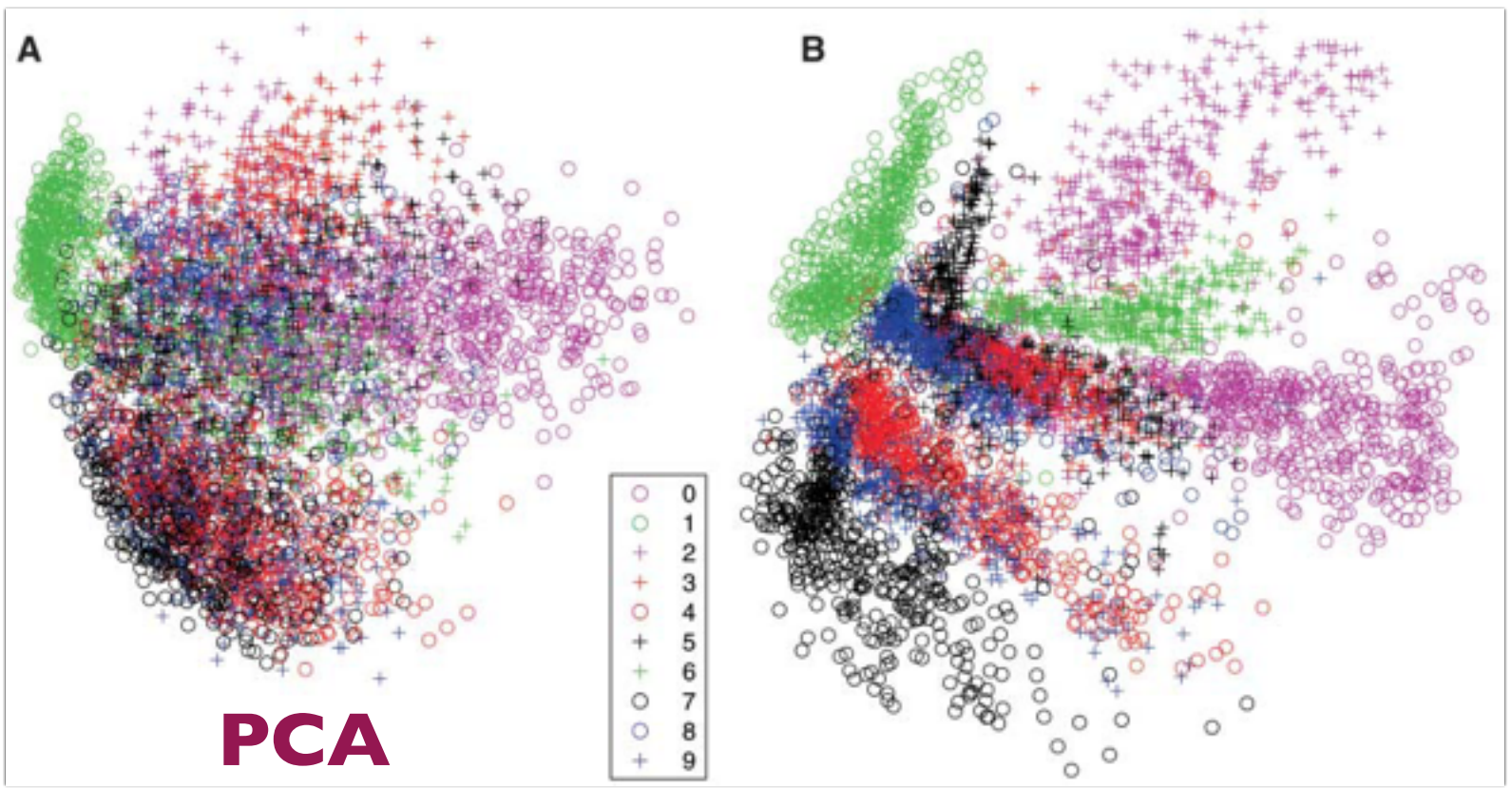
reduce to PCA if f and are linear

$$\min \sum_i \|\boldsymbol{x}_i - g(f(\boldsymbol{x}_i))\|_2^2$$

Example: embedding images of digits

[Hinton & Salakhutdinov, 2006]

(Deep) neural networks



PCA

Contrast PCA to N.N. Autoencoders

- **PCA**

  Identify linear structures only

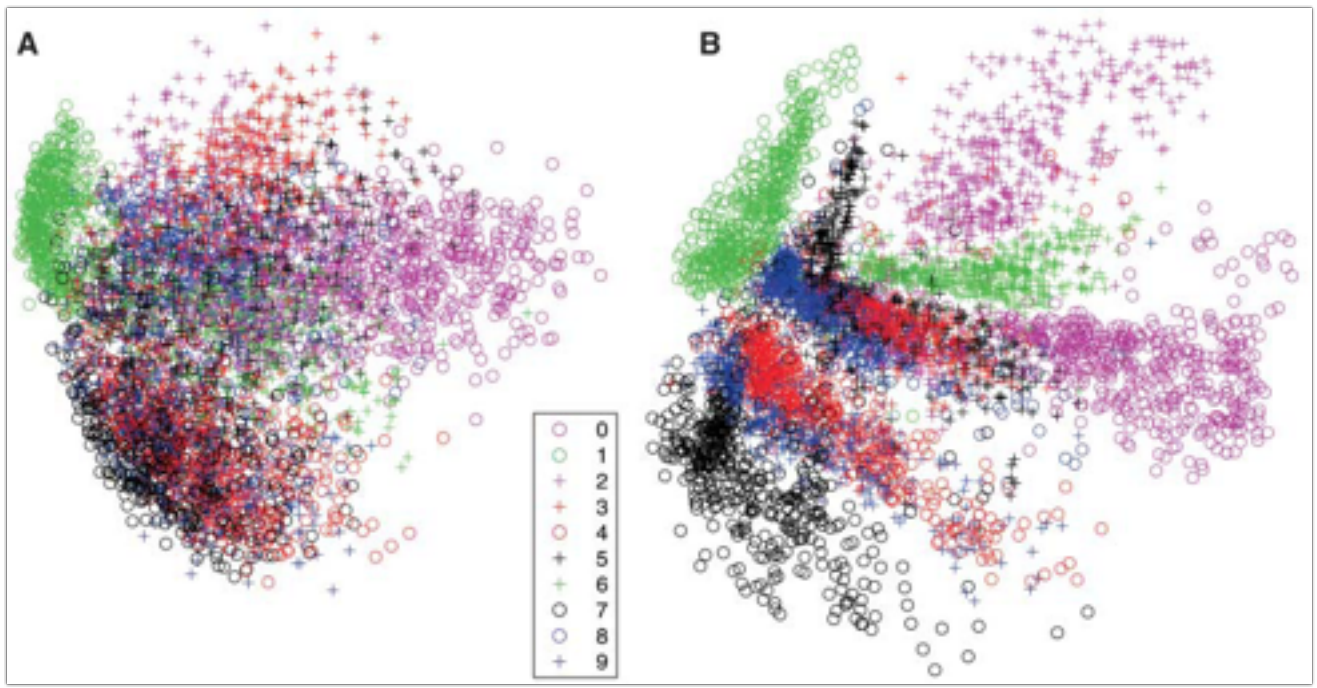
  Easy to solve, no local optima, no tuning parameters

- **Neural network**

  Identify nonlinear structures

  Iterative procedure, local optima, need to tune many parameters (network size, learning rate)

Linear vs nonlinear



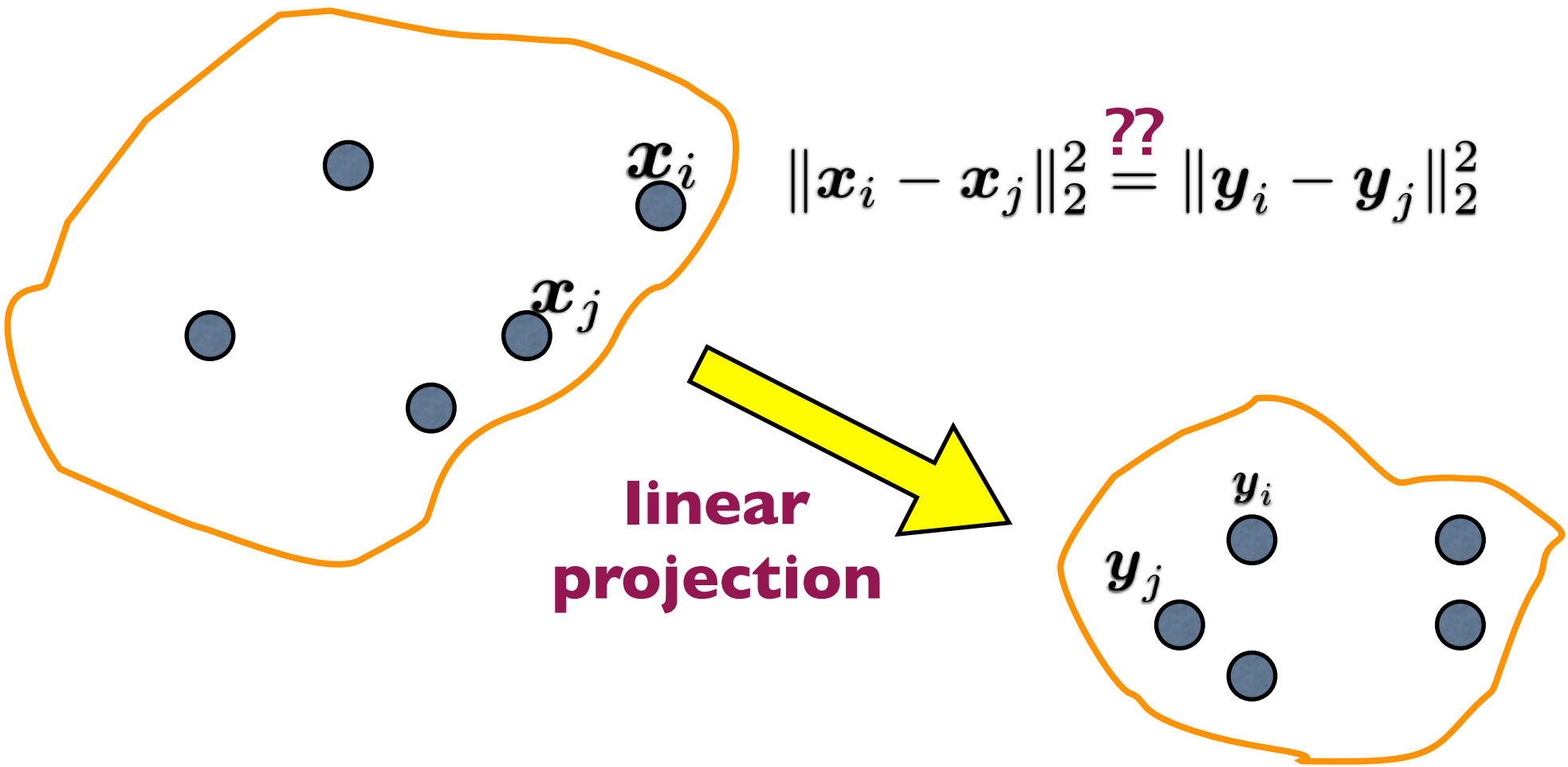
Can we do better?

Discover linear structures

3. Metric Multidimensional Scaling (MDS)

Alternative criteria

- Preserve pairwise distances?



Preserve all pairwise distances

- **Minimize distortion on distances**

$$\min \sum_{ij} \left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2 \right]^2$$

- **Resort to a more tractable formulation**

Preserve pairwise inner products (of centered inputs)

$$\min \sum_{ij} \left[\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j - \boldsymbol{y}_i^{\mathrm{T}} \boldsymbol{y}_j \right]^2$$

Intuition

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_i + \boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{x}_j - 2\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

MDS: match Gram matrix

- **Gram matrix**

  Matrix composed of pairwise inner products

$$\boldsymbol{G} = [G_{ij}] = [\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j] \in \mathbb{R}^{N \times N}$$

- **Optimal embedding**

$$\min \quad \sum_{ij} [G_{ij} - \boldsymbol{y}_i^{\mathrm{T}} \boldsymbol{y}_j]^2$$

No constraints on y!

Solution to MDS

- **Decompose Gram matrix spectrally**

  Compute top M eigenvalues and eigenvectors

$$\boldsymbol{G}\boldsymbol{v} = \lambda \boldsymbol{v} \longrightarrow \{(\boldsymbol{v}_d, \lambda_d)\}_{d=1}^{M}$$

  Note eigenvector is N-dimensional

- **Embed by using eigenvectors as coordinates**

$$y_{id} = \sqrt{\lambda_d}\, v_{di} \qquad d = 1, 2, \ldots, M$$

Especially useful if only know distances

We convert distance matrix

$$\boldsymbol{D} = [D_{ij}^2] \qquad D_{ij}^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

to Gram matrix

$$\boldsymbol{G} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}\boldsymbol{H}$$

with centering matrix

$$\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}$$

PCA vs MDS

- Data matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \cdots \\ \boldsymbol{x}_N \end{bmatrix} \in \mathbb{R}^{N \times D}$$

- PCA decompose covariance matrix

$$\boldsymbol{C}_x = \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \in \mathbb{R}^{D \times D}$$

- MDS decompose Gram matrix

$$\boldsymbol{G} = \boldsymbol{X} \boldsymbol{X}^{\mathrm{T}} \in \mathbb{R}^{N \times N}$$

Are they really that different?

- Same set of eigenvalues

$$\frac{1}{N}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{u} = \lambda\boldsymbol{u} \rightarrow \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\frac{1}{N}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{u} = N\lambda\frac{1}{N}\boldsymbol{X}\boldsymbol{u}$$

PCA **MDS**

- Similar low dimensional representation
- Different computational cost

  PCA scales quadratically in D

  MDS scales quadratically in N

Big win for MDS when D is much greater than N !

PCA vs MDS

- **PCA**

  Linear method, parametric

  \+ Fast, easy to compute

  \- Does not unfold nonlinear structures

- **MDS**

  Dual to PCA

  \+ Can be faster

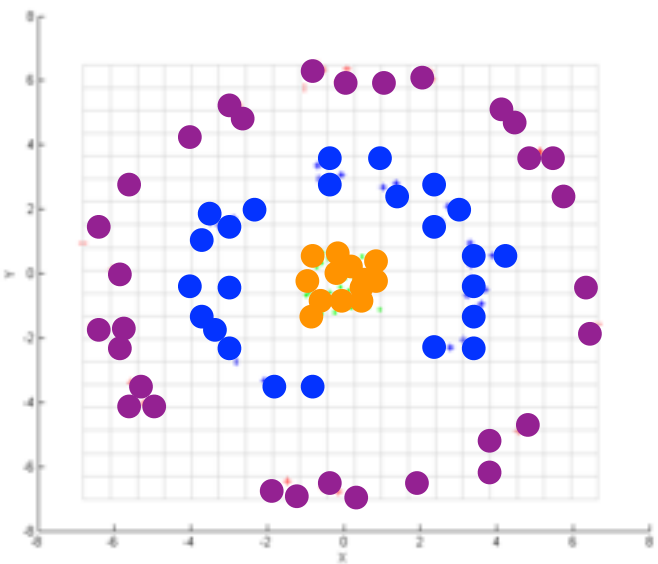
  \+ Need only distances or inner products

Discover nonlinear structures

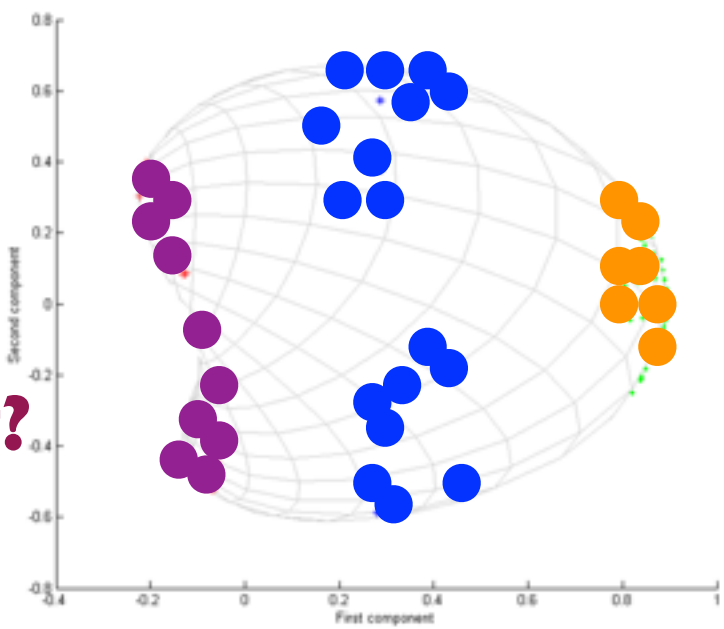
4. kernel PCA

Central challenge

Linear method is not good enough



after nonlinear mapping , linear method works!



How to identify this mapping?

Kernel method

- **Kernel functions**

  Avoid explicitly finding a nonlinear mapping

  $$\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x})$$

  Define implicitly through a kernel function

  $$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)$$

- **Common kernels**

  Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/\sigma^2\right\}$

  Polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$

Kernel PCA

- **Main idea**

  Map data to kernel feature space; then run MDS

- **Recipe**

  Choose a kernel function

  Compute kernel matrix $\boldsymbol{K} = [K(\boldsymbol{x}_i, \boldsymbol{x}_j)]$

  Centralize kernel matrix $\tilde{\boldsymbol{K}} \leftarrow \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}$

  Decompose spectrally $\tilde{\boldsymbol{K}} = \sum_d \lambda_d \boldsymbol{v}_d \boldsymbol{v}_d^{\mathrm{T}}$

  Project

$$\boldsymbol{y} \propto \tilde{\boldsymbol{K}}\boldsymbol{v}$$

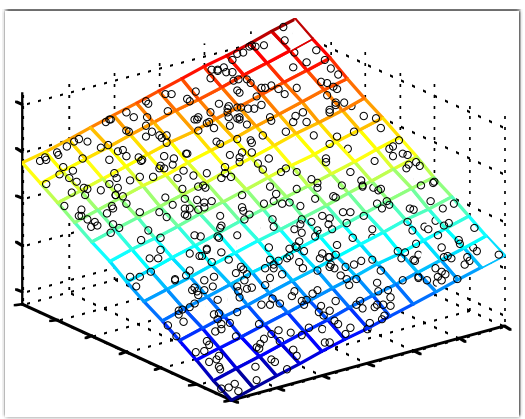
(there are some scaling constants)

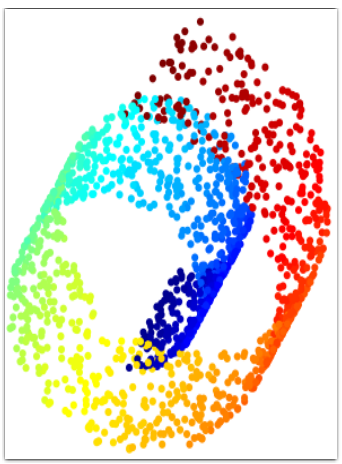
Does it work?

- Demo

Next lecture

How to discover nonlinear structures?





Going from linear to nonlinear is not that difficult!

Dimension Reduction

Prof. Fei Sha

Dept. of Computer Science
U. of Southern California

IPAM Summer School on Computer Vision
Aug. 6 2013

Outline of previous lecture (Part 1)

- **Central questions**

  Discover linear structures

  Generalize to nonlinear structures

- **Algorithms**

  Principal Component Analysis (PCA)

  Metric Multidimensional Scaling (MDS)

  Neural Network Autoencoders

  Kernel PCA

Outline of this lecture (Part 2)

- **Central questions**

  Discover nonlinear structures

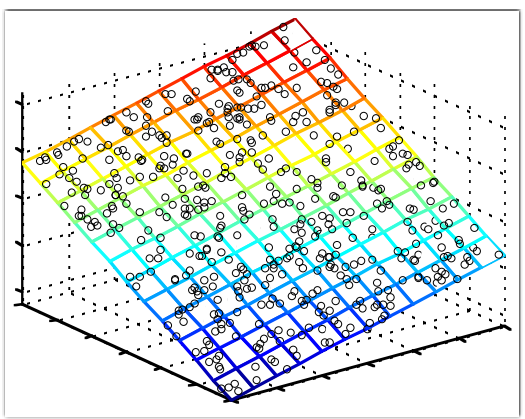
- **Algorithms**

  Isomap

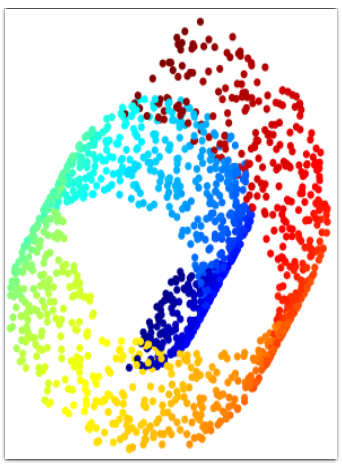
  Locally linear embedding (LLE)

  Maximum variance unfolding (MVU)

What does it take to discover nonlinear structures?

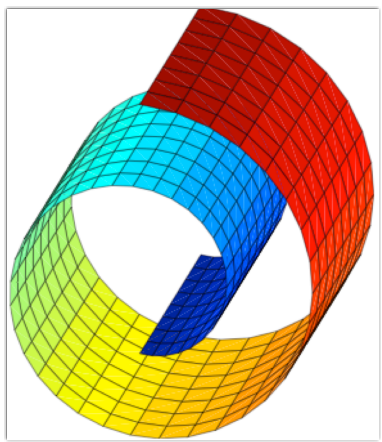


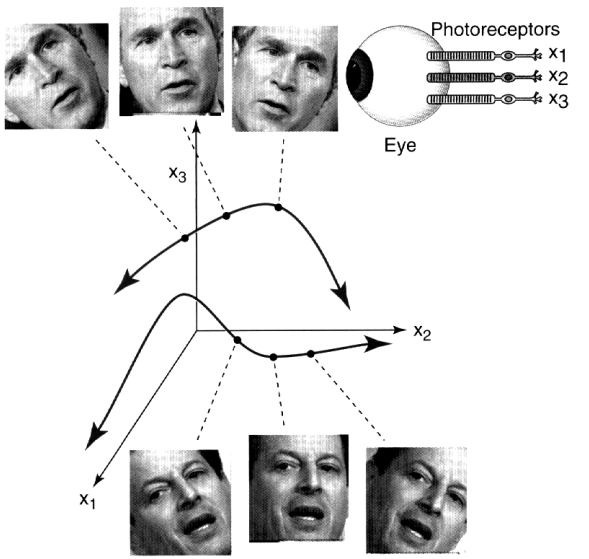
Being myopic!



Nonlinear structures

- Manifolds such as

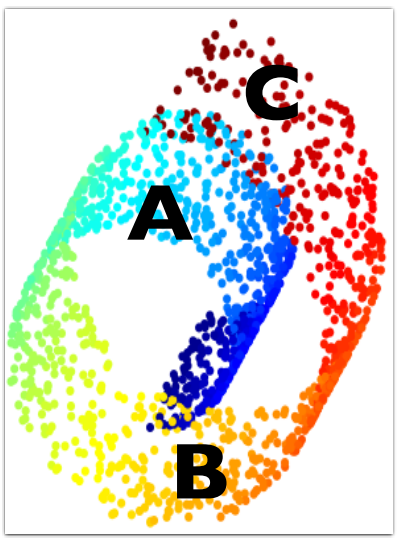




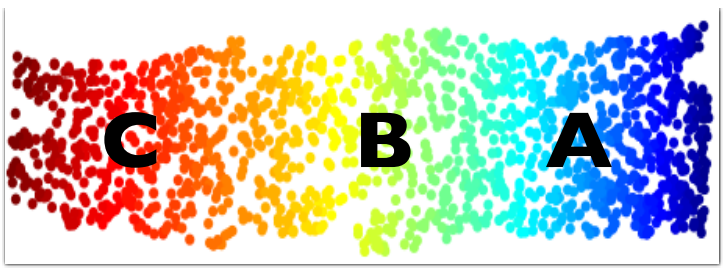
- can be approximately locally with linear structures.

Why good to be myopic?

Euclidean distance is not appropriate measure of proximity between points on nonlinear manifold.



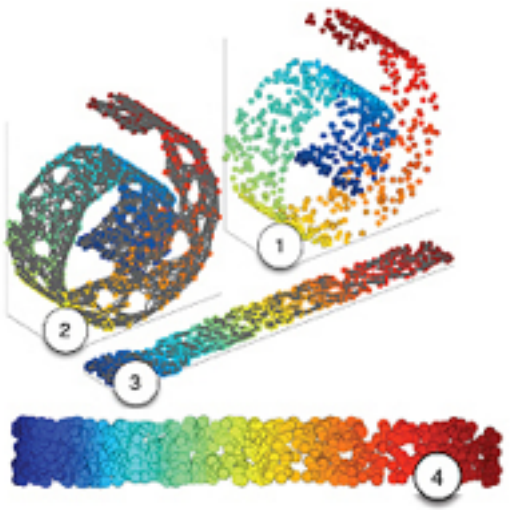
A closer to C in Euclidean distance



A closer to B once unrolled

Nonlinear methods: manifold learning

Preserve geometrical properties on manifolds



Local symmetry and proximity

- Locally linear embedding (Roweis & Saul ' 00)
- Laplacian eigenmap (Belkin & Niyogi' 03)

Local isometry and angle

- Local Tangent Space Alignment (LTSA) (Zhang 04)
- hessian LLE (Donoho & Grime '03)
- Maximum variance unfolding (MVU) (Weinberger & Saul 04)
- Conformal component analysis (Sha & Saul 05)

(Estimated) geodesic distances

- IsoMap (Tenenbaum et al 00)

Diffusion distances

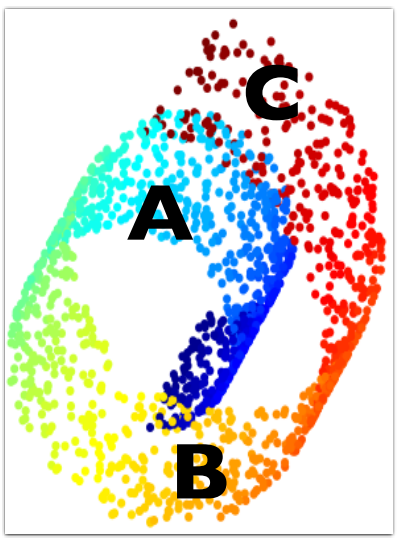
- Diffusion map (Coifman et al 06)

Manifold learning

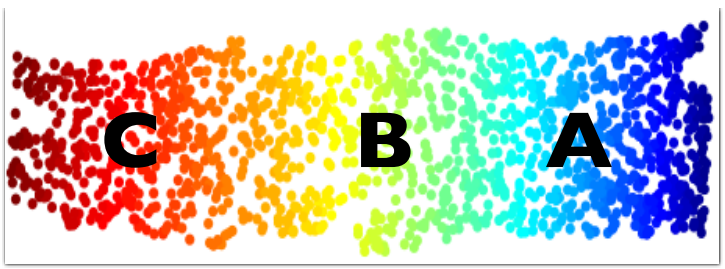
1. Isomap

IsoMap: Preserving geodesic distances

Geodesic distance is more appropriate measure of proximity between points on nonlinear manifold.



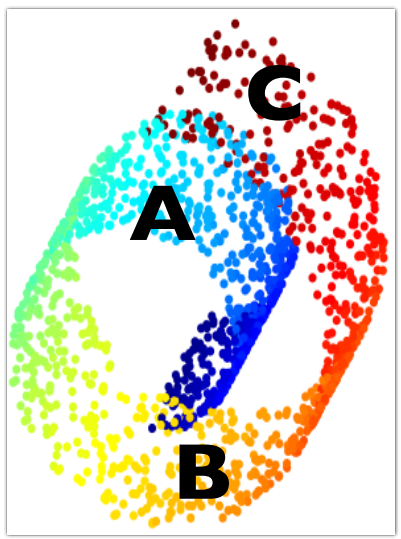
A closer to C in Euclidean distance



A closer to B in geodesic distance

Caveat

Without knowing the shape of the manifold, how to estimate the geodesic distance?



The tricks will unfold next....

Step 1. Build adjacency graph

- **Graph from nearest neighbor**

  Vertices represent inputs

  Edges connect nearest neighbors

- **How to choose nearest neighbor**

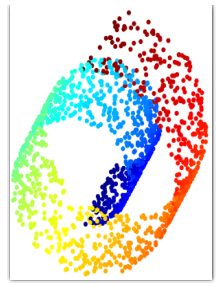
  k-nearest neighbors

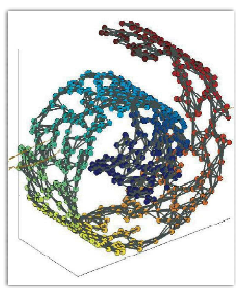
  Epsilon-radius ball

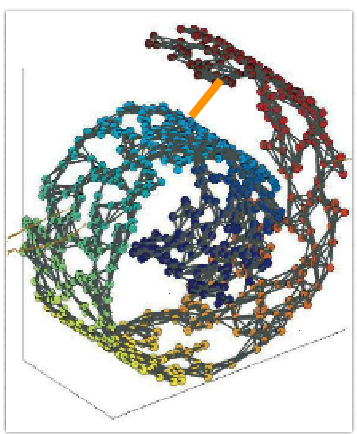
- **Assumptions**

  Graph is connected

  No “shortcuts” that connect wrongfully





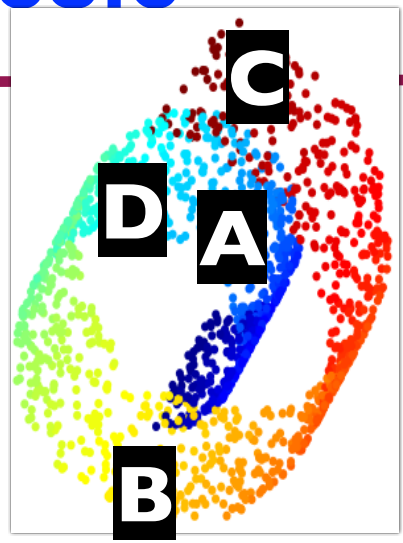


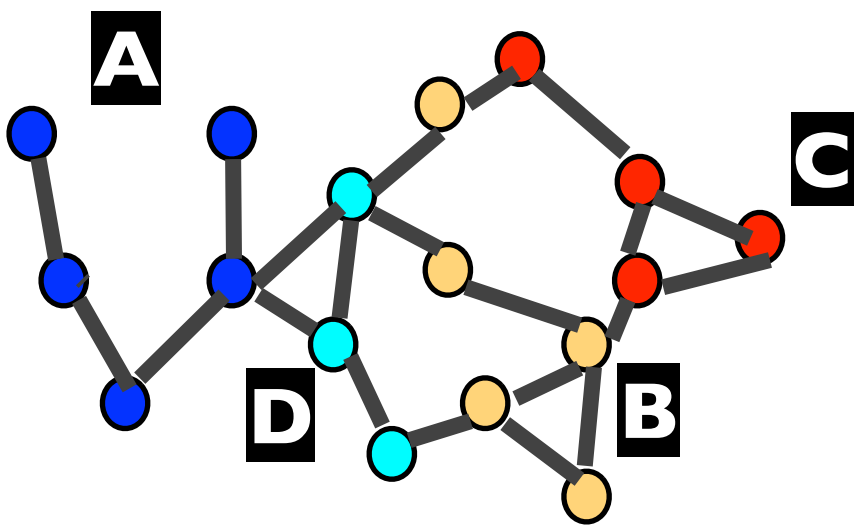
From local distance to geodesic

- **Approximation**

  Distances on manifold are modeled as shortest paths on the graph

  all pairwise shortest paths can be computed efficiently



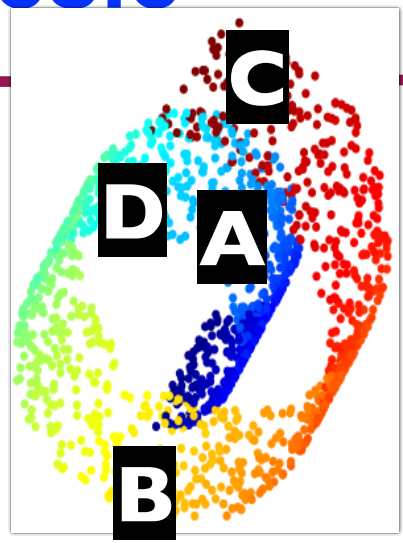


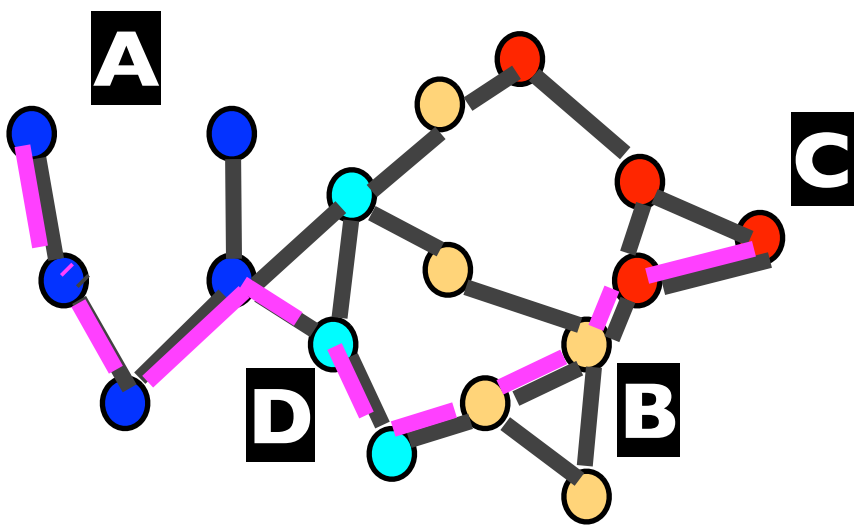
From local distance to geodesic

- **Approximation**

  Distances on manifold are modeled as shortest paths on the graph

  all pairwise shortest paths can be computed efficiently





Step 2. Construct geodesic distance matrix

- **Geodesic distances**

  Weight edges by local Euclidean distance

  Approximate geodesic by shortest paths

- **Computational cost**

  Require all pair shortest paths (Djikstra's algorithm: $O(N^2 \log N + N^2 k)$)

  Require dense sampling to approximate well

  (very intensive for large graph)

Step 3. Apply MDS

- **Convert geodesic matrix to Gram matrix**

  Pretend the geodesic matrix is from Euclidean distance matrix

- **Diagonalize the Gram matrix**

  Gram matrix is a dense matrix, ie, no sparsity

  Can be intensive if the graph is big.

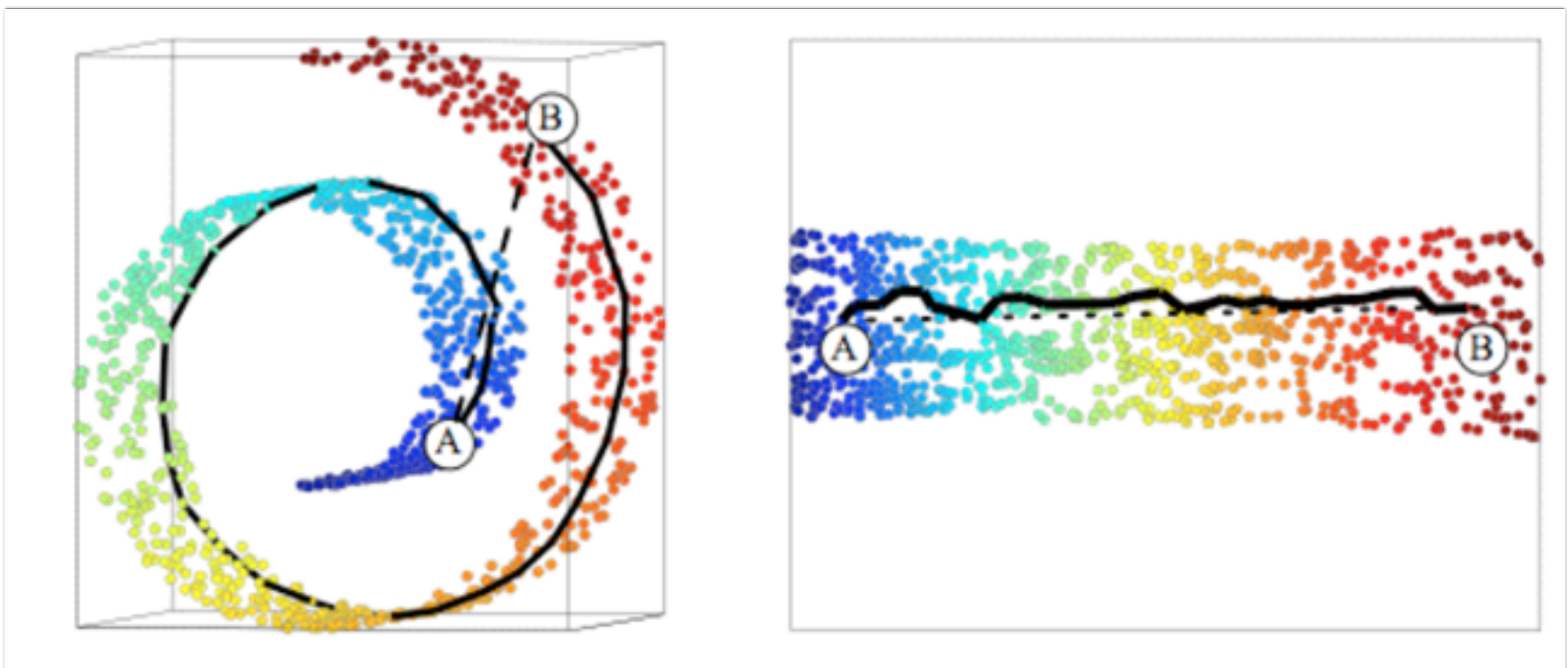
- **Embedding**

  Number of significant eigenvalues yield estimate of dimensionality

  Top eigenvectors yield embedding.

Examples

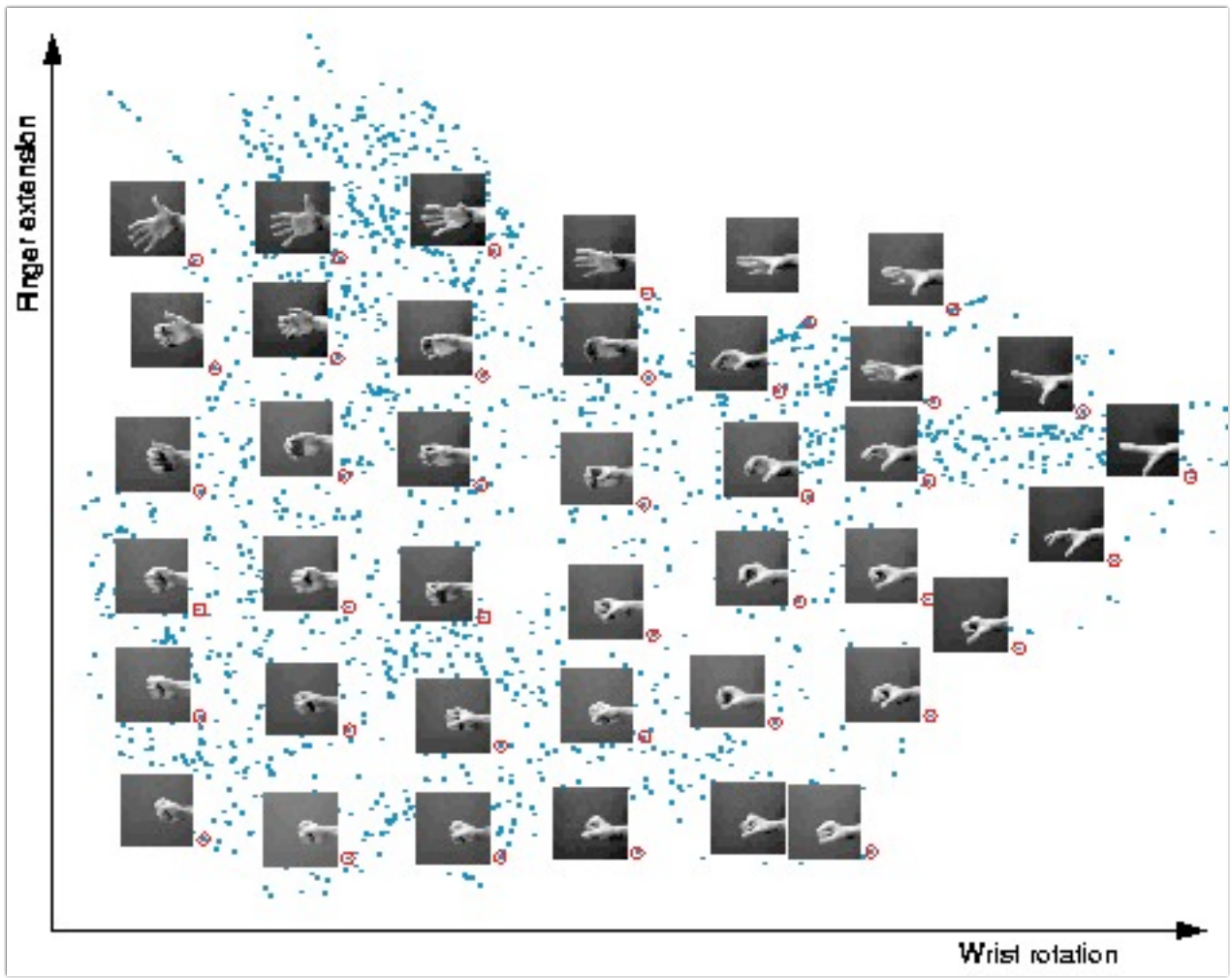
- Swiss roll



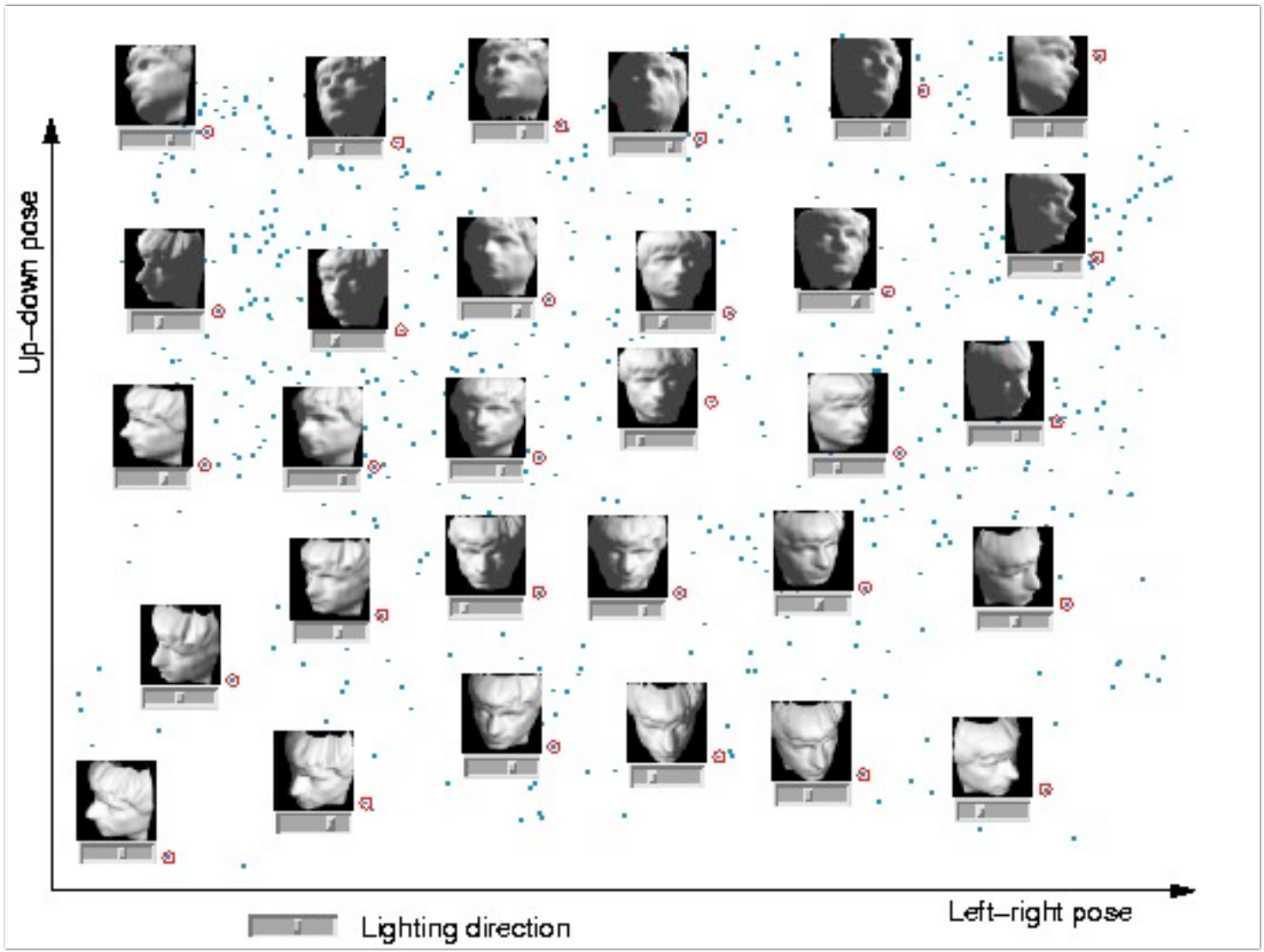
N=1024, k=12

- **Wrist images**

N=200
k=6
D = 4096



- **Face images**



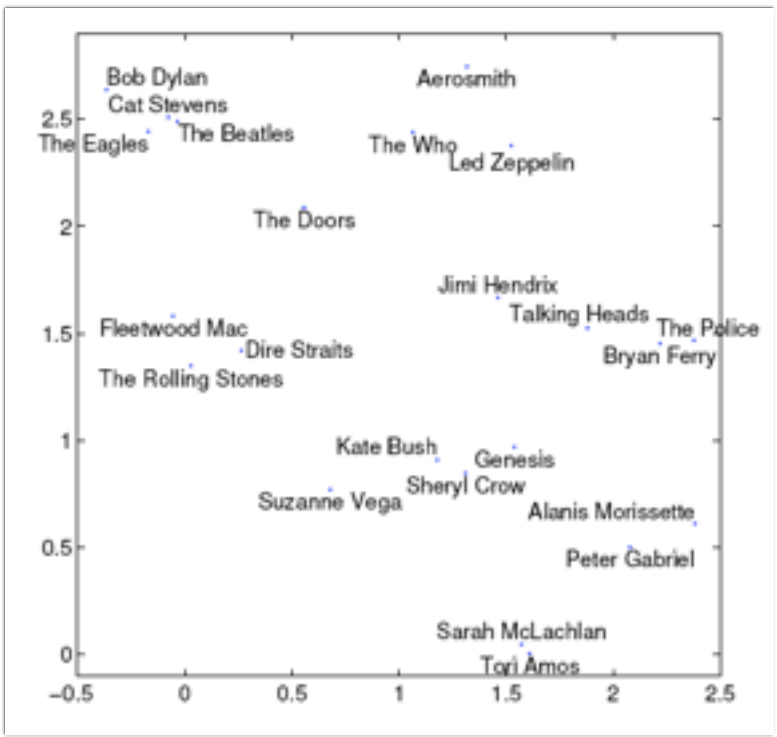
Overcome computational bottleneck

- **Embed music similarity graph**

  Landmark MDS

  avoid computing geodesic distances between all pairs

N= 267,000
E = 3,200,000



[Platt, 2004]

Summary of Isomap

- **Recurring theme**

  Construct (sparse) graph from kNN

  Discretize manifold with graphs

  Formulate matrix-based optimization from graph weights

  Preserve properties derived from those weights

  Derive embedding from decomposing the matrix

  Reduce dimensionlity with MDS

Manifold learning

2. Locally linear embedding (LLE)

Locally linear embedding (LLE)

- **Intuition**

  Better off being myopic and trusting only local information

- **Steps**

  Define locality by nearest neighbors

  Encode local information **Least square fit locally**

  Minimize global objective to preserve local information **Think globally**

Step 1. Build adjacency graph

- **Graph from nearest neighbor**

  Vertices represent inputs

  Edges connect nearest neighbors

- **How to choose nearest neighbor**

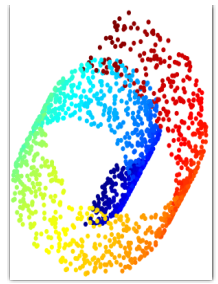
  k-nearest neighbors

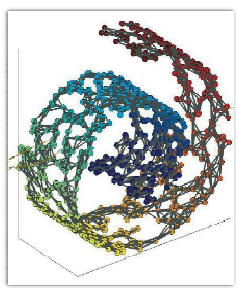
  Epsilon-radius ball

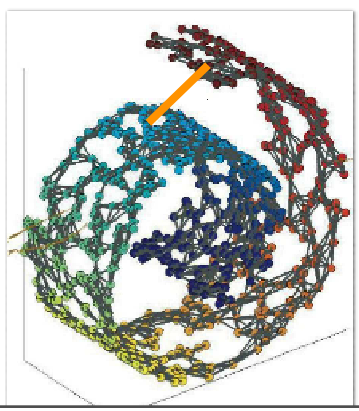
- **Assumptions**

  Connected graph

  No short-circuit



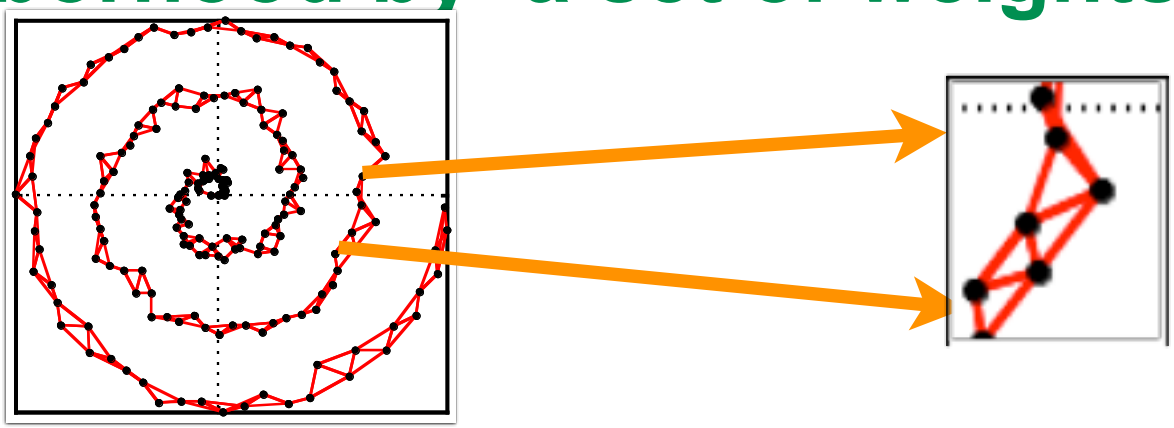




Large k would cause this problem

Step 2. Least square fits

- Characterize local geometry of each neighborhood by a set of weights

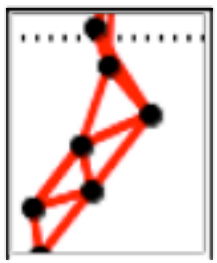


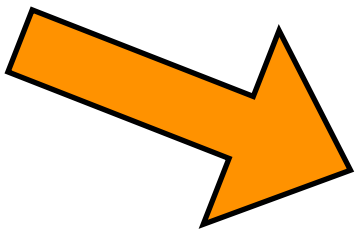
- Compute weights by reconstructing each input linearly from its neighbors

$$\Phi(\boldsymbol{W}) = \sum_i \|\boldsymbol{x}_i - \sum_k \boldsymbol{W}_{ik}\boldsymbol{x}_k\|^2$$

subject to $\sum_k W_{ik} = 1$

Weighted adjacency graph





W =

Vertex

0.00				
	0.00			
0.2	0.00	0.00	0.9	-0.1
			0.00	
				0.00

weights for i_{th} vertex

Vertex

What are these weights for?

0				
	0			
0.2	0	0	0.9	-0.1
			0	
				0

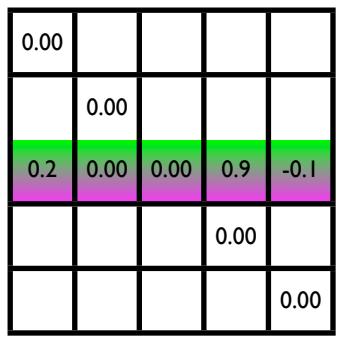
They are shift, rotation, scale invariant.

The head should sit in the middle of left and right finger tips.

Step 3. Preserve local information

- The embedding should follow same local encoding

$$\boldsymbol{y}_i \approx \sum_k \boldsymbol{W}_{ik} \boldsymbol{y}_k$$



- Minimize a global reconstruction error

$$\Psi(\boldsymbol{Y}) = \sum_i \|\boldsymbol{y}_i - \sum_k \boldsymbol{W}_{ik} \boldsymbol{y}_k\|^2$$

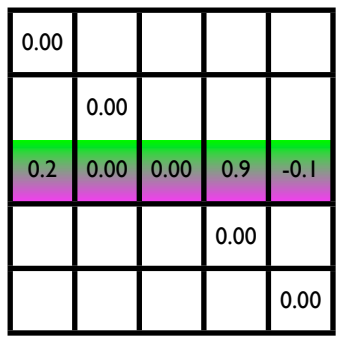
subject to $\sum \boldsymbol{y}_i = \boldsymbol{0}$

$$\frac{1}{N} \boldsymbol{Y} \boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}$$

Sparse eigenvalue problem

- Minimize a global reconstruction error

$$\Psi(\boldsymbol{Y}) = \sum_i \|\boldsymbol{y}_i - \sum_k \boldsymbol{W}_{ik}\boldsymbol{y}_k\|^2$$



subject to $\sum \boldsymbol{y}_i = \boldsymbol{0}$

$$\frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}$$

- Solution

Bottom eigenvectors of $(\boldsymbol{I} - \boldsymbol{W})^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{W})$

Relates to kernel PCA

Summary of LLE

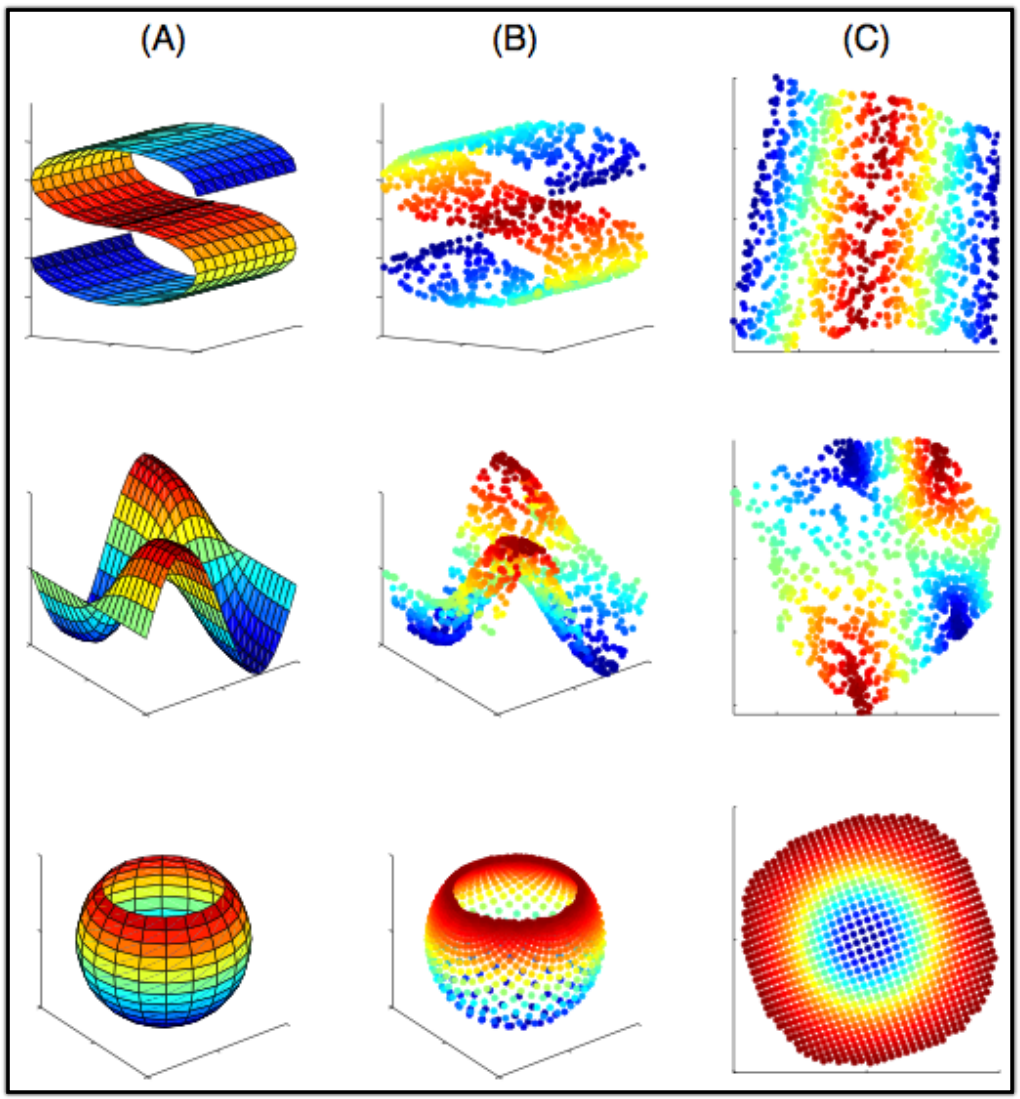
- Build k-nearest neighbor graph
- Solve linear least square fit for each neighbor

  Closed-form solution
- Solve a sparse eigenvalue problem

Every step is relatively trivial, however the combined effect is quite complicated.

Examples

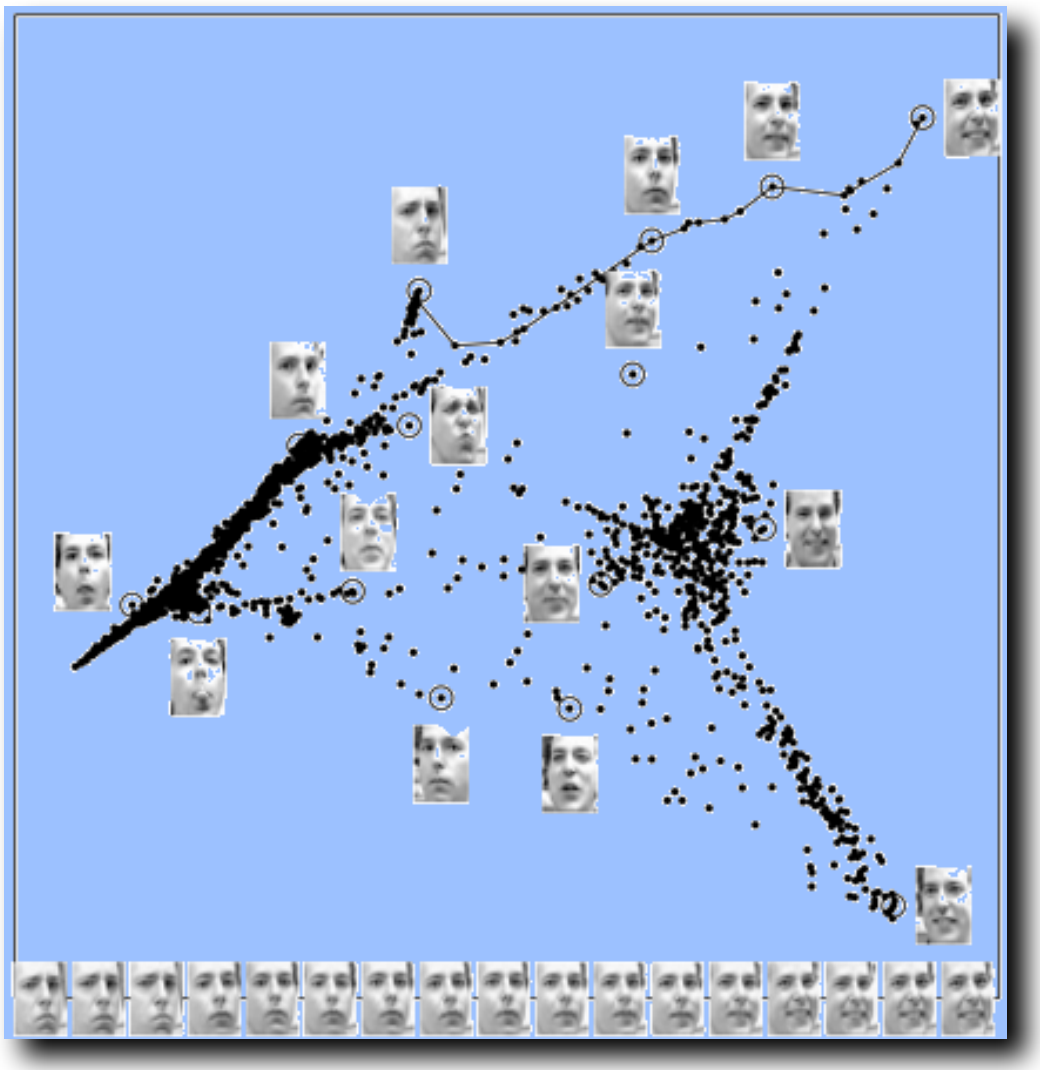
N = 1000
k = 8
D = 3
d = 2



Examples of LLE

- Pose and expression

N = 1965
k = 12
D = 560
d = 2



LLE vs Isomap

- **Similarity**

  Graph-based

  Spectral decomposition of matrix

- **Differences**

  Dense (Isomap) vs Sparse (LLE)

  Top (Isomap) vs Bottom (LLE) eigenvectors

  Preserve distances (Isomap) vs local geometry (LLE)

Manifold learning

3. Maximum variance unfolding (MVU)

What if we preserve distances explicitly?

- **But, we will get MDS!**

  True, though

- **what if we preserve local distances?**

  We trust local distances anyway.

  In contrast to Isomap, we would not even be bothered by estimating global (geodesic distances)

Preserve local distances

- **Quadratic programming**

$$\max \quad \sum_i \|\boldsymbol{y}_i\|^2$$

unfolding

$$\sum_i \boldsymbol{y}_i = 0$$

centering

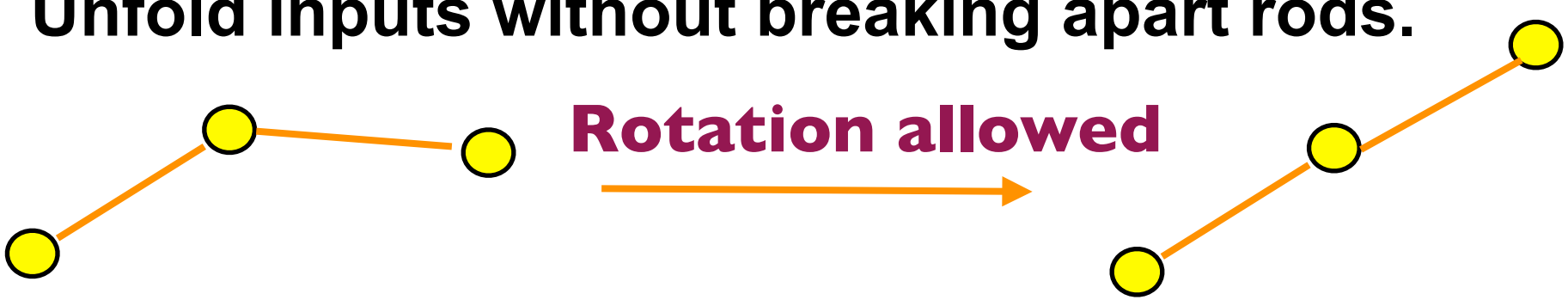
$$\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

only if i and j are nearest neighbor!

- **Intuition**

Nearby points are connected with rigid rods

Unfold inputs without breaking apart rods.



Maximum variance unfolding

(Weinberger & Saul 2003, Weinberger, & Saul 2004)

- **Change of variables**

$$K_{ij} = \boldsymbol{y}_i^{\mathrm{T}} \boldsymbol{y}_j$$

- **Semidefinite programming (SDP)**

$$\max \quad \sum_{ii} K_{ii}$$

unfolding objective

$$\sum_{ij} K_{ij} = 0$$

See this trick before?

$$K_{ii} + K_{jj} - 2K_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

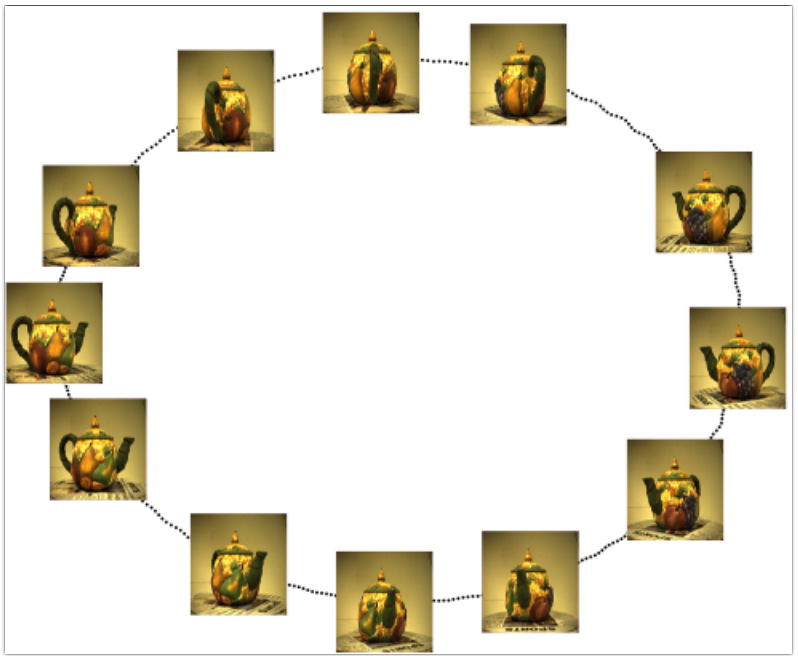
$$\boldsymbol{K} \succeq 0$$

Gram matrix needs to be positive semidefinite

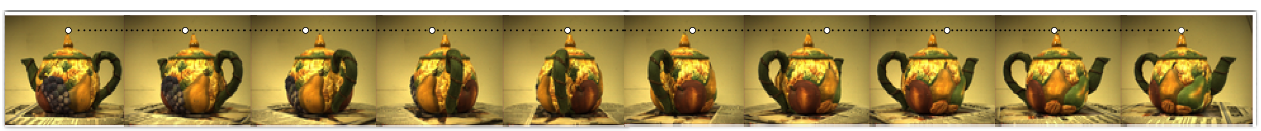
Example: images of rotating teapot

- Full rotation

N = 400
k = 4
D = 23028

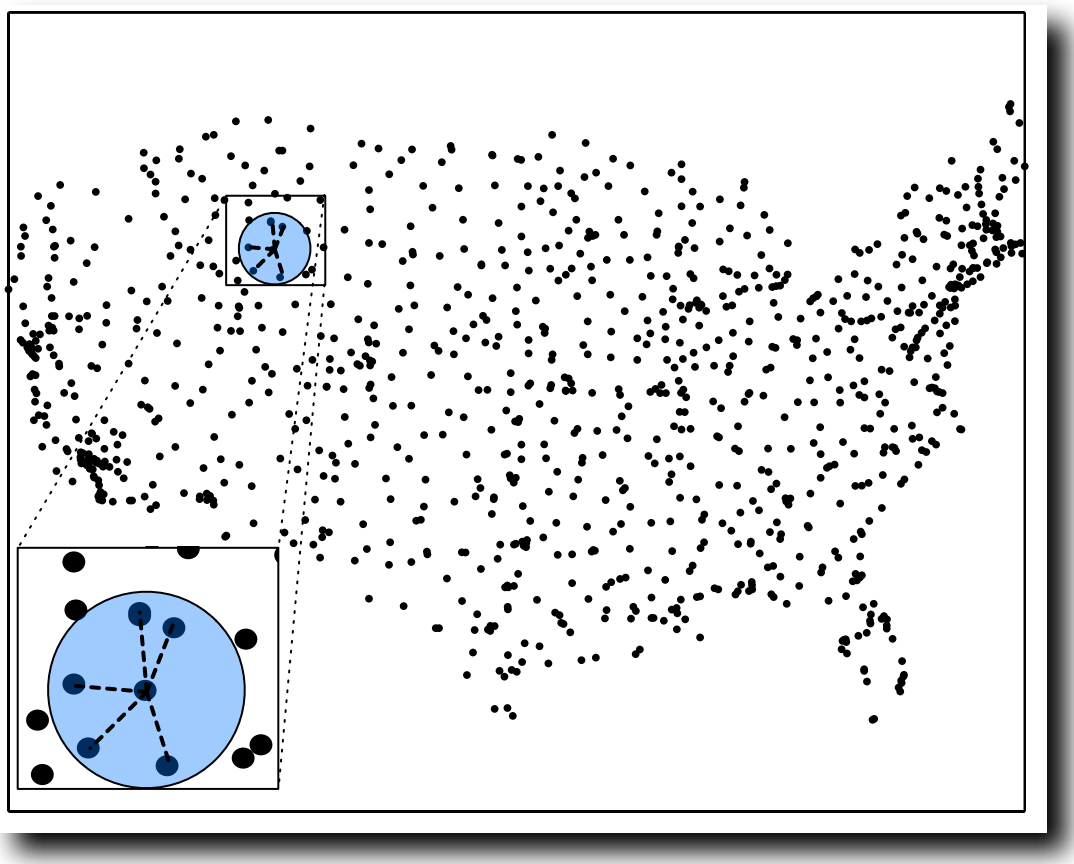


- Half rotation



Images are ordered by d=I embedding according to view angle

Example: sensor localization



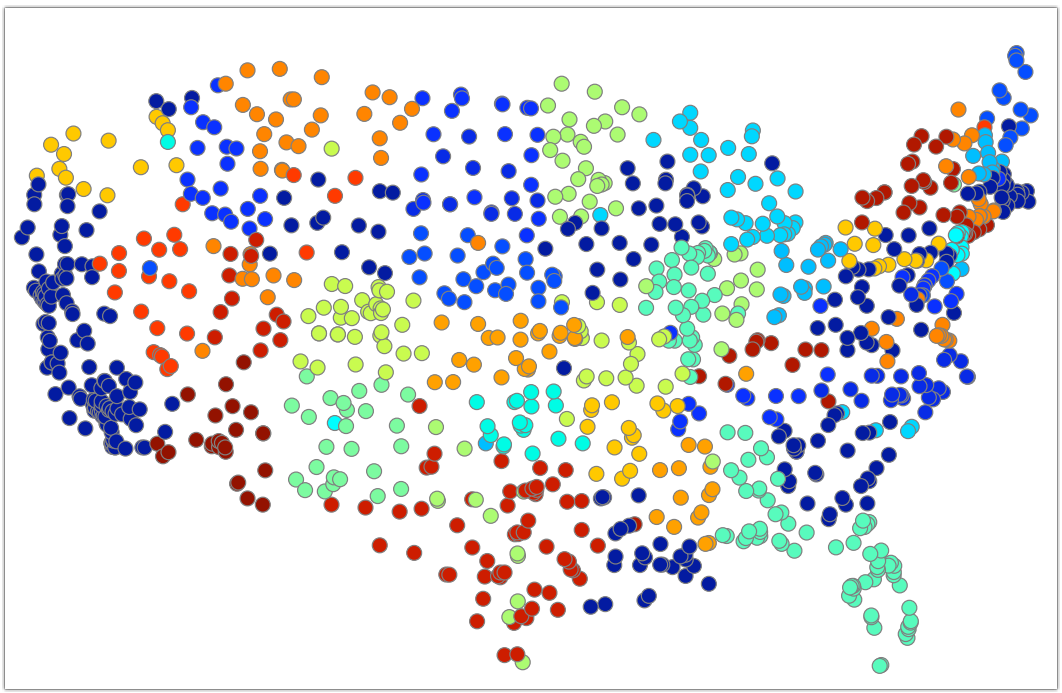
cities

cities

$$\begin{bmatrix} 0 & d_{12} & ? & d_{14} \\ d_{21} & 0 & d_{23} & ? \\ ? & d_{32} & 0 & d_{34} \\ d_{41} & ? & d_{43} & 0 \end{bmatrix}$$

sensors distributed in US cities.
Infer coordinates from limited measurement of distances
(Weinberger, Sha & Saul, NIPS 2006)

Where are the cities?



Discover US atlas almost perfectly!

Summary

- **3 methods**

  IsoMap

  Global geodesic

  LLE

  Local symmetry and proximity

  MVU

  Local isometry

- **Common properties**

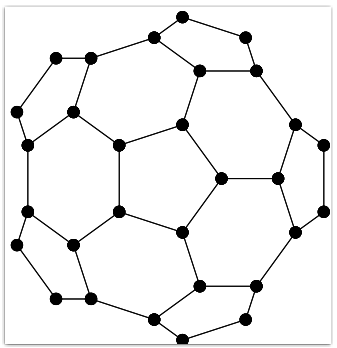
  graph methods, geometry, convex optimization

Summary: graph based spectral methods

- **Construct nearest neighbor graph**

  Vertices are data points

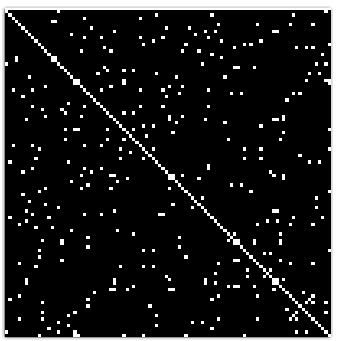
  Edges indicate nearest neighbors



- **Optimization**

  Spectral decomposition, or

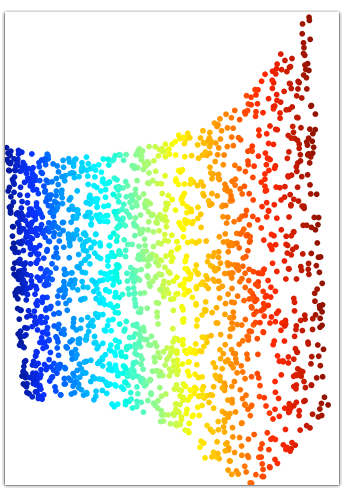
  Semidefinite programming



- **Derive embedding**

  Eigenvector as embedding

  Estimate dimensionality



Link to differential geometry

- **Thru Laplace-Betrami operator**

  Graph as discretized (continuous) manifold

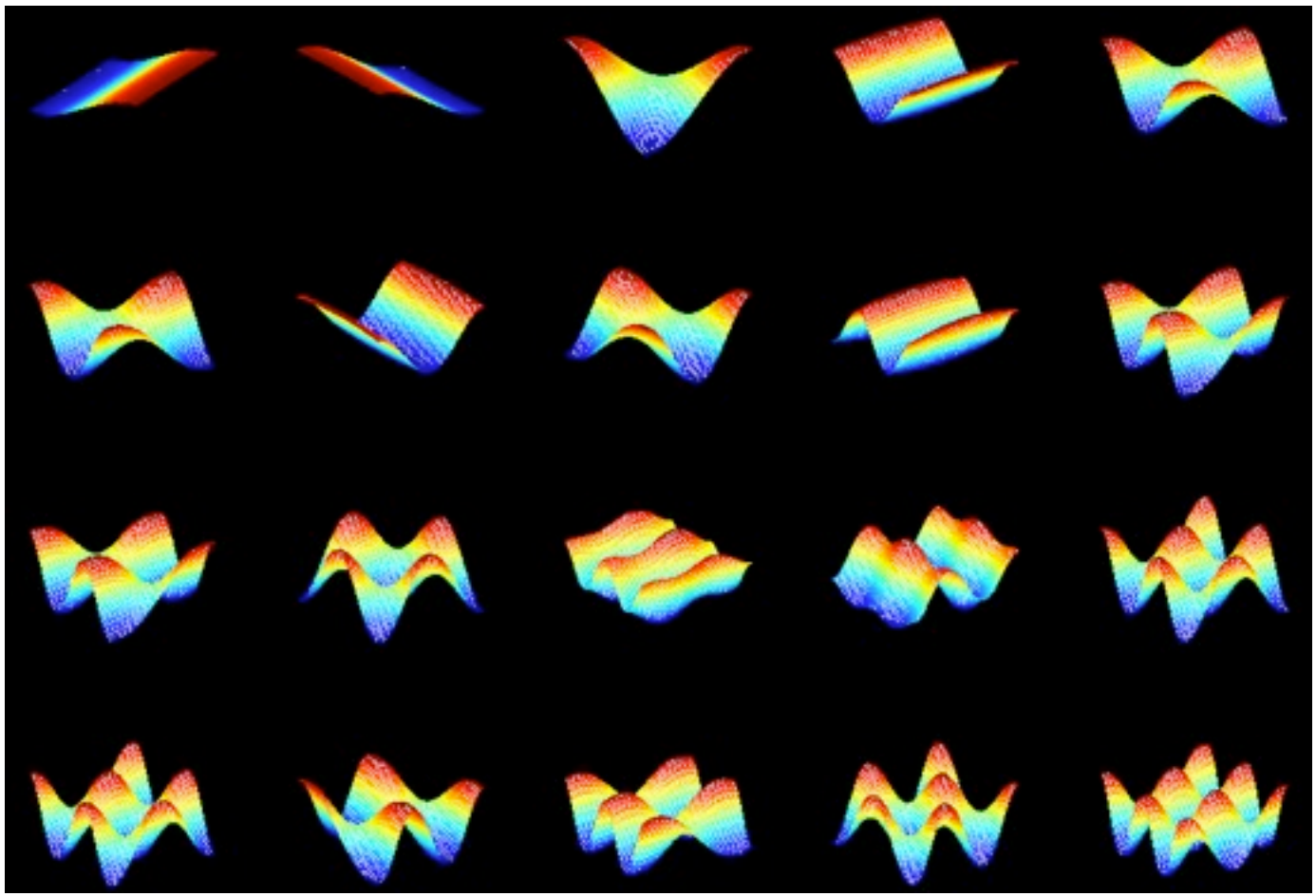
  Continuous operator --> discrete Laplacian

- **Spectral decomposition**

  Eigenvectors form functional basis.

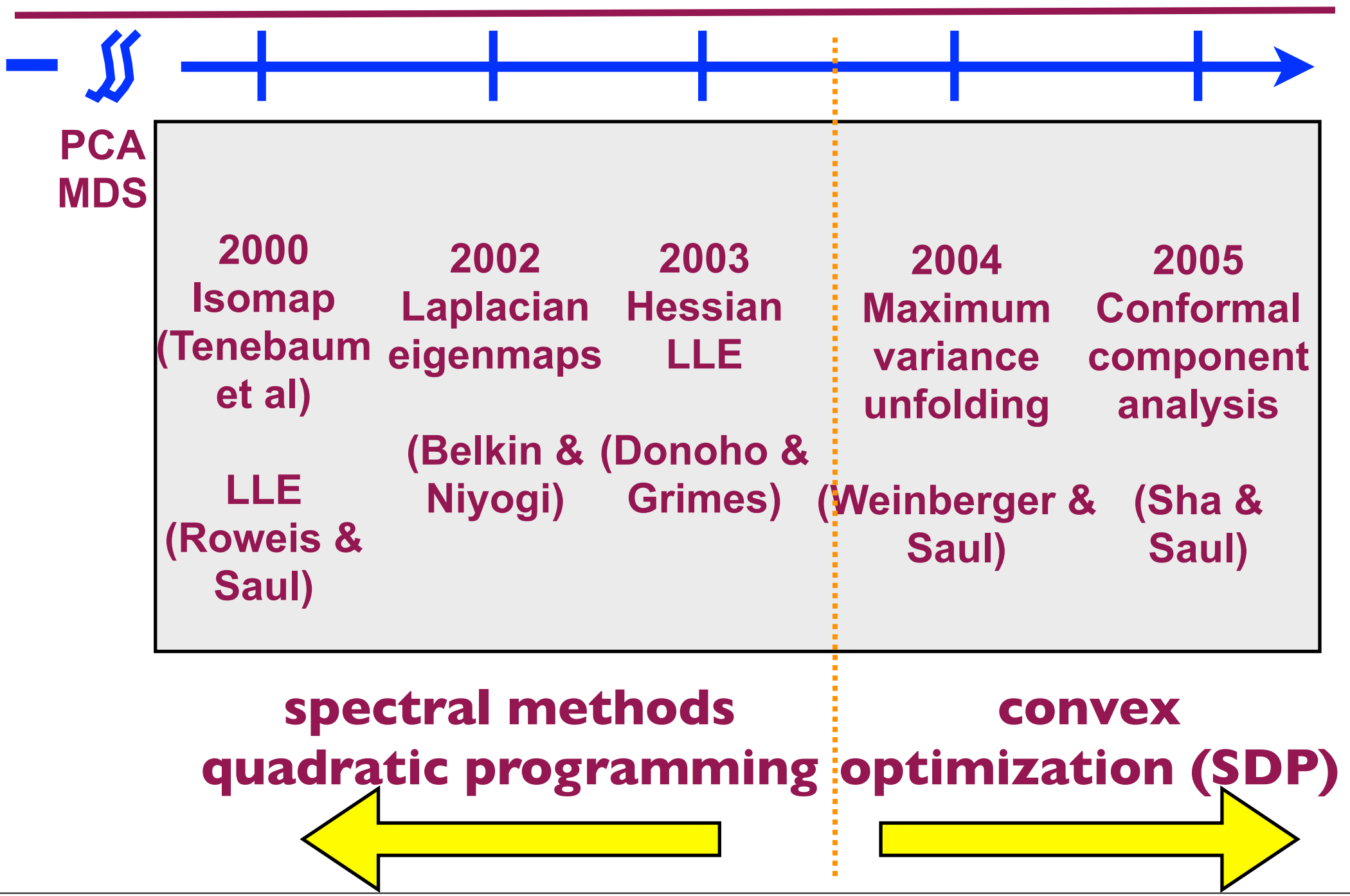
  Key difference: basis are data-dependent

Basis for functions defined on graphs



Conclude

Bird's eye view



Take-home messages

- **Dimension reduction**

  Many ideas and techniques

  Many applications (in vision, NLP, etc)

  Computationally tractable methods are often surprisingly powerful

- **Link to other areas**

  Learning sparse representation

  Learning global representation

Acknowledgements

- Many material taken from Prof. Lawrence Saul's IPAM 2005 Tutorial on Spectral Methods for Dimension Reduction