Spectral Methods for Dimensionality Reduction

Prof. Lawrence Saul

Dept of Computer & Information Science
University of Pennsylvania

NIPS*05 Tutorial, December 5, 2005

Neural Information Processing Systems Conference

Dimensionality reduction

- **Question**

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

- **Applications**
  - **Digital image and speech processing**
  - **Analysis of neuronal populations**
  - **Gene expression microarray data**

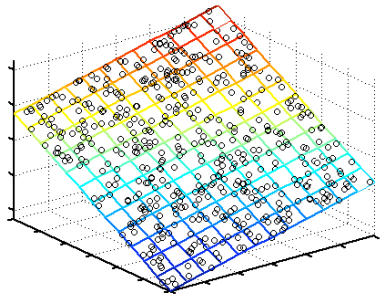
Framework

- **Data representation**

  Inputs are real-valued vectors in a high dimensional space.

- **Linear structure**

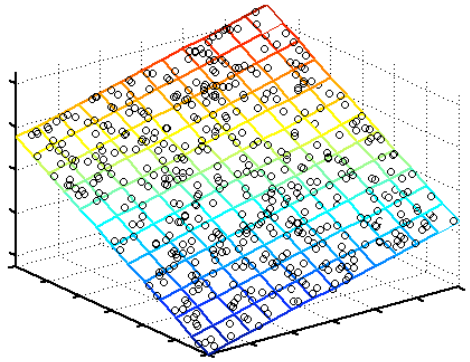
  Does the data live in a low dimensional subspace?



- **Nonlinear structure**

  Does the data live on a low dimensional submanifold?

Linear vs nonlinear



What computational price must we pay for nonlinear dimensionality reduction?

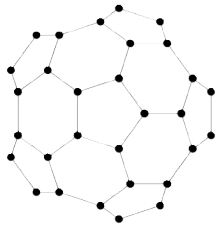
Spectral methods

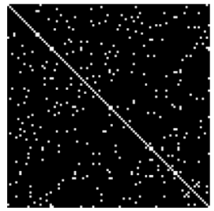
- **Matrix analysis**

  Low dimensional structure is revealed by eigenvalues and eigenvectors.

- **Links to spectral graph theory**

  Matrices are derived from sparse weighted graphs.





- **Usefulness**

  Tractable methods can reveal nonlinear structure.

Notation

- **Inputs (high dimensional)**

$$\vec{x}_i \in \Re^D \text{ with } i = 1, 2, ..., n$$

- **Outputs (low dimensional)**

$$\vec{y}_i \in \Re^d \text{ where } d \ll D$$

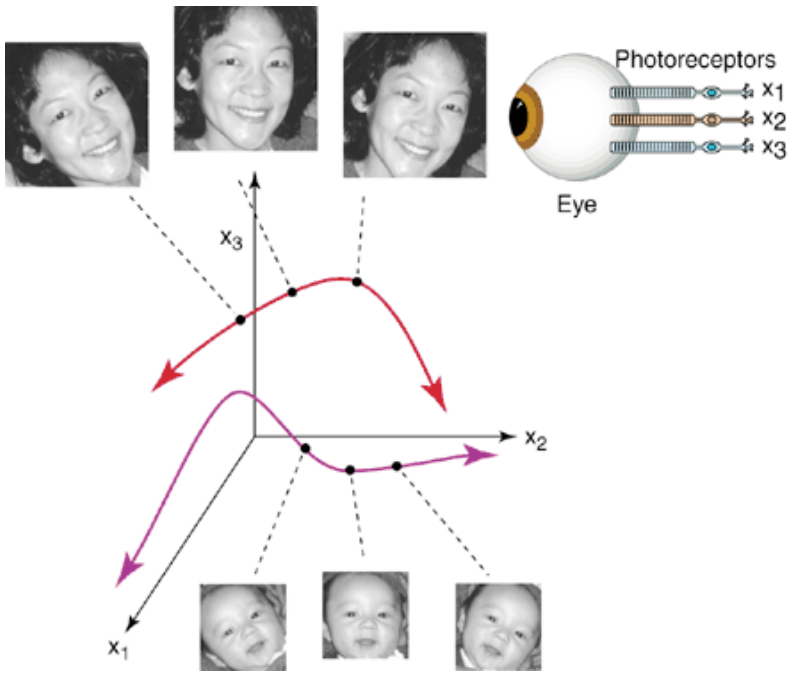
- **Goals**

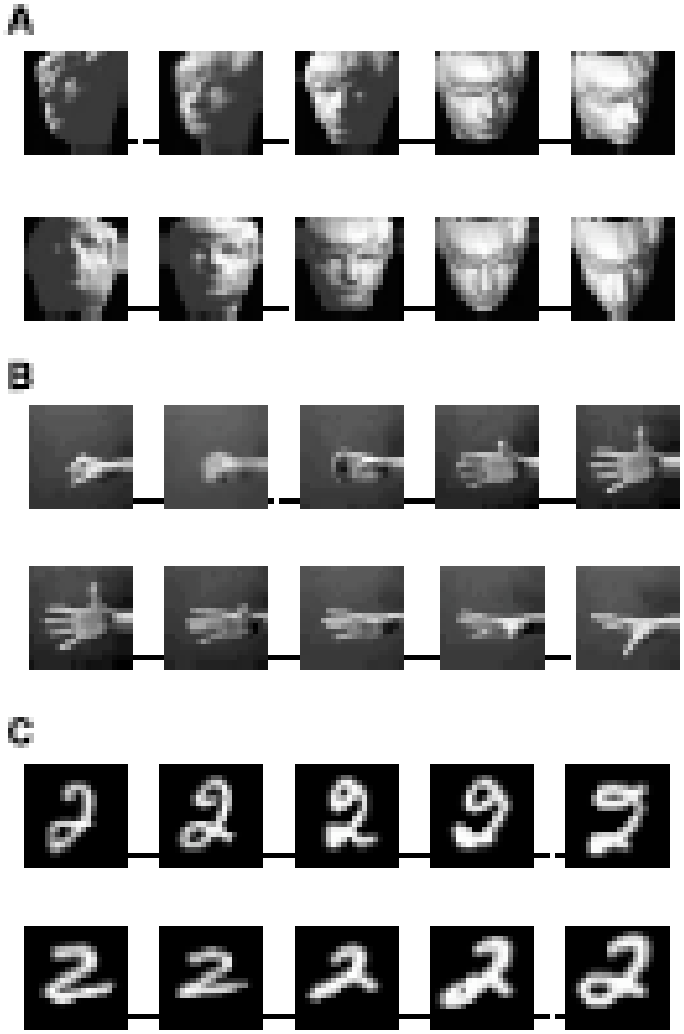
Nearby points remain nearby.
Distant points remain distant.
(Estimate *d*.)

Manifold learning

Given high dimensional data sampled from a low dimensional submanifold, how to compute a faithful embedding?

Image Manifolds





(Seung & Lee, 2000)
(Tenenbaum et al, 2000)

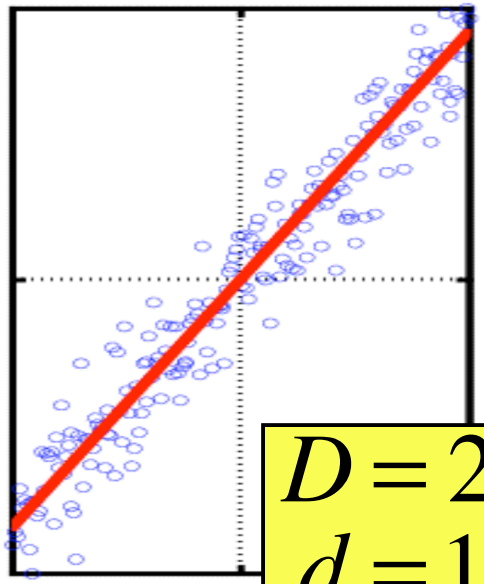
Outline

- **Part 1 - linear versus graph-based methods**
- **Part 2 - sparse matrix methods**
- **Part 3 - semidefinite programming**
- **Part 4 - kernel methods**
- **Part 5 - parting thoughts**

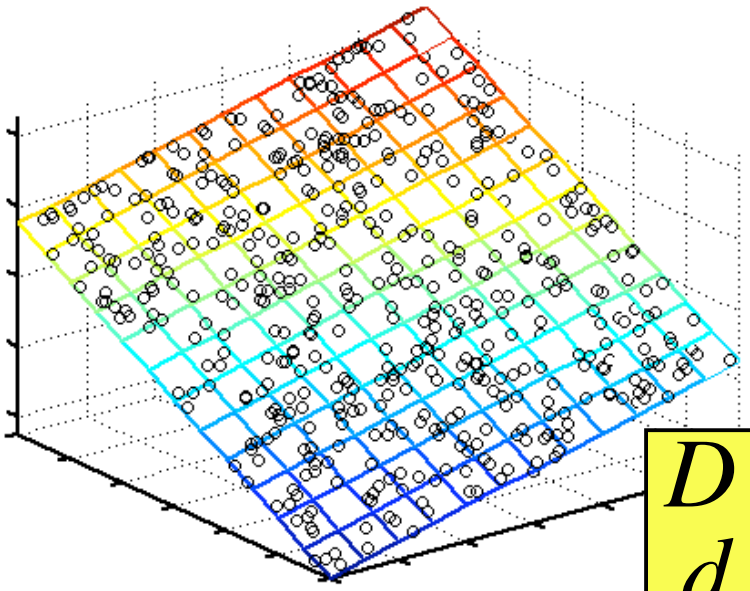
Linear method #1

Principal Components Analysis (PCA)

Principal components analysis



$D = 2$
$d = 1$



$D = 3$
$d = 2$

Does the data mostly lie in a subspace?
If so, what is its dimensionality?

Maximum variance subspace

- **Assume inputs are centered:**

$$\sum_i \vec{x}_i = \vec{0}$$

- **Project into subspace:**

$$\vec{y}_i = P\vec{x}_i \quad \text{with} \quad P^2 = P$$

- **Maximize projected variance:**

$$\text{var}(\vec{y}) = \frac{1}{n}\sum_i \| P\vec{x}_i \|^2$$

Matrix diagonalization

- **Covariance matrix**

$$\mathrm{var}(\vec{y}) = \mathrm{Tr}(PCP^{\mathrm{T}}) \quad \text{with} \quad C = n^{-1}\sum_i \vec{x}_i \vec{x}_i^{T}$$

- **Spectral decomposition**

$$C = \sum_{\alpha=1}^{D} \lambda_\alpha \vec{e}_\alpha \vec{e}_\alpha^{\mathrm{T}} \quad \text{with} \quad \lambda_1 \geq \cdots \geq \lambda_D \geq 0$$

- **Maximum variance projection**

$$P = \sum_{\alpha=1}^{d} \vec{e}_\alpha \vec{e}_\alpha^{\mathrm{T}}$$

Projects into subspace spanned by top d eigenvectors.

Interpreting PCA

- **Eigenvectors:**

  principal axes of maximum variance subspace.

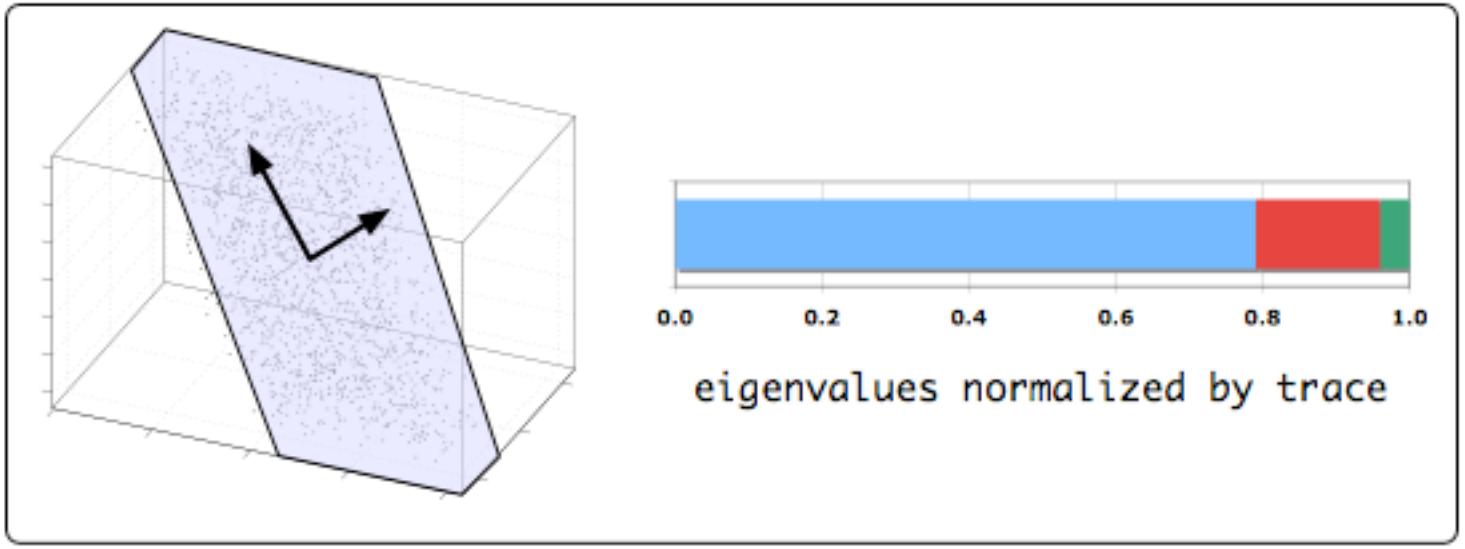
- **Eigenvalues:**

  projected variance of inputs along principle axes.

- **Estimated dimensionality:**

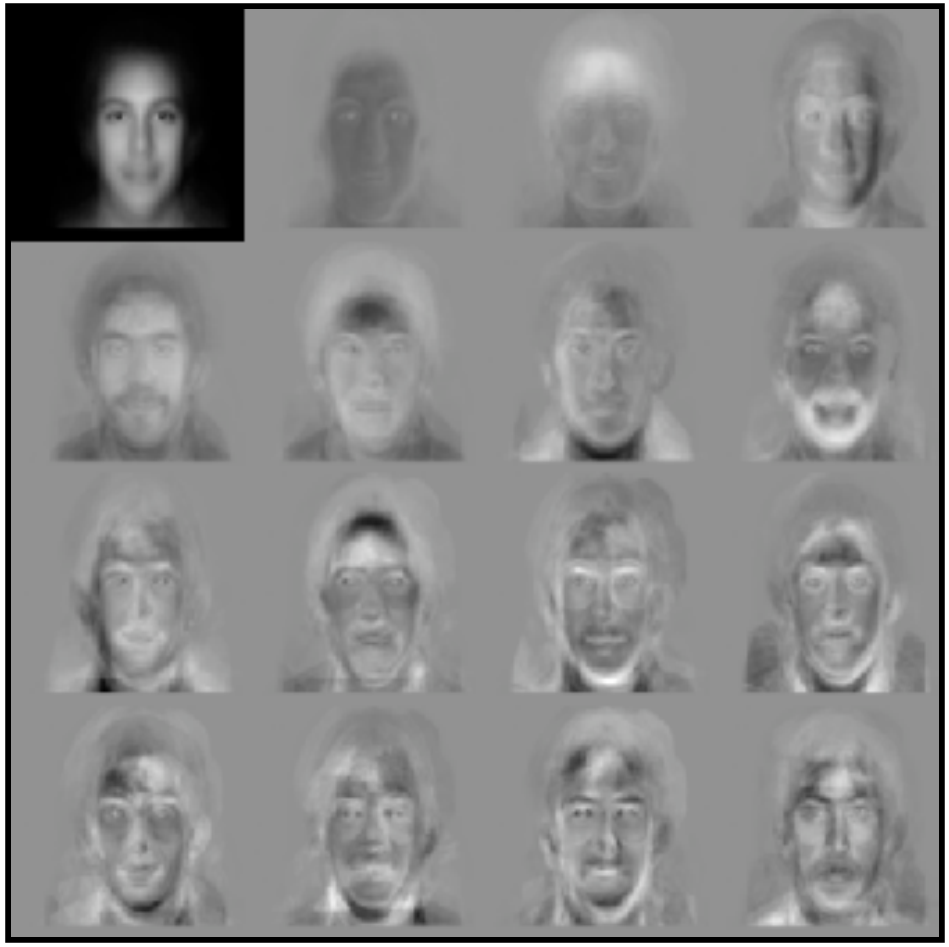
  number of significant (nonnegative) eigenvalues.

Example of PCA



Eigenvectors and eigenvalues of covariance matrix for $n=1600$ inputs in $d=3$ dimensions.

Example: faces



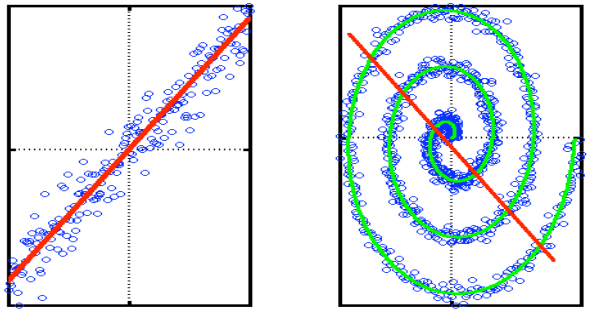
Eigenfaces from 7562 images:

top left image is linear combination of rest.

Sirovich & Kirby (1987)
Turk & Pentland (1991)

Properties of PCA

- **Strengths**
  - Eigenvector method
  - No tuning parameters
  - Non-iterative
  - No local optima
- **Weaknesses**
  - Limited to second order statistics
  - Limited to linear projections

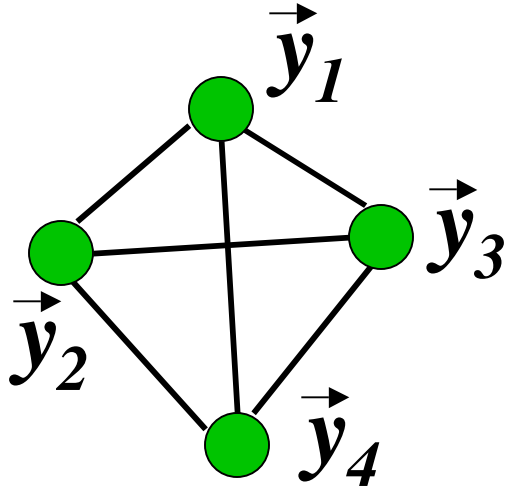


Linear method #2

Metric Multidimensional Scaling (MDS)

Multidimensional scaling

$$\begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{13} & \Delta_{23} & 0 & \Delta_{34} \\ \Delta_{14} & \Delta_{24} & \Delta_{34} & 0 \end{bmatrix}$$



Given $n(n-1)/2$ pairwise distances Δ_{ij}, find vectors $\vec{y}_i$ such that $\|\vec{y}_i - \vec{y}_j\| \approx \Delta_{ij}$.

Metric Multidimensional Scaling

- **Lemma**

  If Δ_{ij} denote the Euclidean distances of zero mean vectors, then the inner products are:

$$G_{ij} = \frac{1}{2}\left[\frac{1}{n}\sum\nolimits_k \left(\Delta_{ik}^2 + \Delta_{kj}^2\right) - \Delta_{ij}^2 - \frac{1}{n^2}\sum\nolimits_{kl}\Delta_{kl}^2\right]$$

- **Optimization**

  Preserve dot products (proxy for distances). Choose vectors $\vec{y}_i$ to minimize:

$$\mathrm{err}(\vec{y}) = \sum\nolimits_{ij}\left(G_{ij} - \vec{y}_i \bullet \vec{y}_j\right)^2$$

Matrix diagonalization

- **Gram matrix "matching"**

$$\mathrm{err}(\vec{y}) = \sum_{ij} \left(G_{ij} - \vec{y}_i \bullet \vec{y}_j \right)^2$$

- **Spectral decomposition**

$$G = \sum_{\alpha=1}^{n} \lambda_\alpha \vec{v}_\alpha \vec{v}_\alpha^{\mathrm{T}} \quad \text{with} \quad \lambda_1 \geq \cdots \geq \lambda_n \geq 0$$

- **Optimal approximation**

$$y_{i\alpha} = \sqrt{\lambda_\alpha} v_{\alpha i} \quad \text{for} \quad \alpha = 1, 2, \ldots, d \quad \text{with } d \leq n$$

(scaled truncated eigenvectors)

Interpreting MDS

$$y_{i\alpha} = \sqrt{\lambda_\alpha} v_{\alpha i} \quad \text{for} \quad \alpha = 1, 2, \ldots, d \quad \text{with } d \ll n$$

- **Eigenvectors**

  Ordered, scaled, and truncated to yield low dimensional embedding.

- **Eigenvalues**

  Measure how each dimension contributes to dot products.

- **Estimated dimensionality**

  Number of significant (nonnegative) eigenvalues.

Relation to PCA

- **Dual matrices**

$$C_{\alpha\beta} = n^{-1} \sum_i x_{i\alpha} x_{i\beta} \quad \text{covariance matrix } (D \times D)$$
$$G_{ij} = \vec{x}_i \bullet \vec{x}_j \quad \text{Gram matrix} \quad (n \times n)$$

- **Same eigenvalues**

  Matrices share nonzero eigenvalues up to constant factor.

- **Same results, different computation**

  PCA scales as $O((n+d)D^2)$.
  MDS scales as $O((D+d)n^2)$.

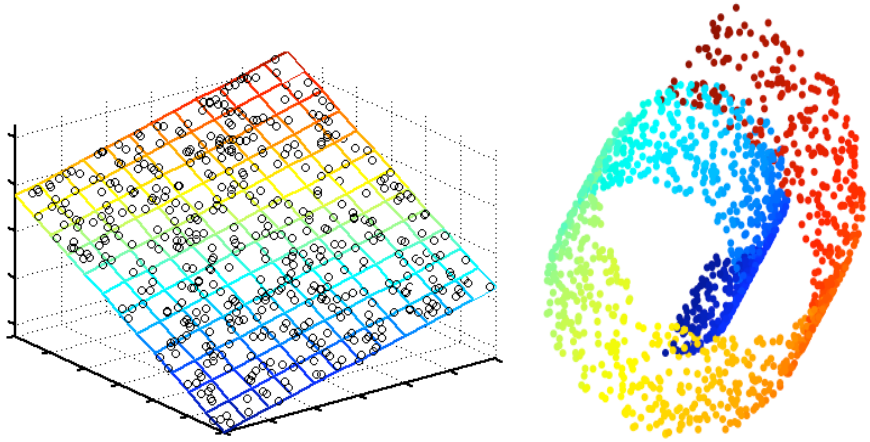
So far..

- **Q: How to detect linear structure?**

  A1: Principal components analysis

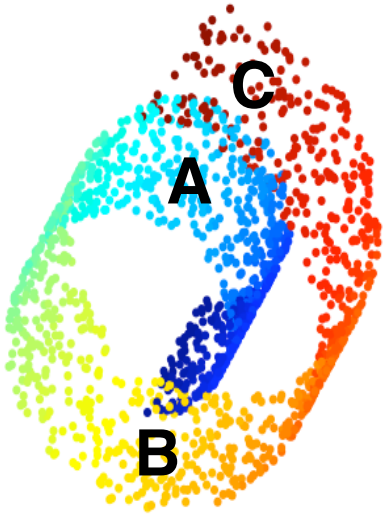
  A2: Metric multidimensional scaling

- **Q: How to generalize for manifolds?**

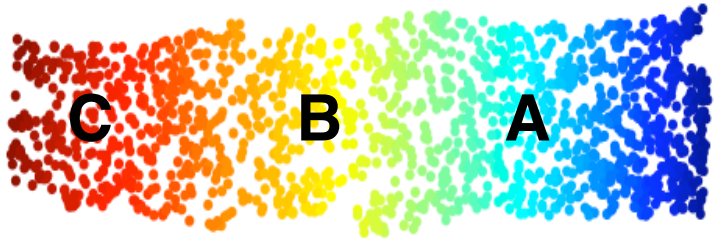


Non-monotonicity

Rank ordering of Euclidean distances is NOT preserved in "manifold learning".



$d(A,C) < d(A,B)$



$d(A,C) > d(A,B)$

Graph-based method #1

Isometric mapping of data manifolds (ISOMAP)

(Tenenbaum, de Silva, & Langford, 2000)

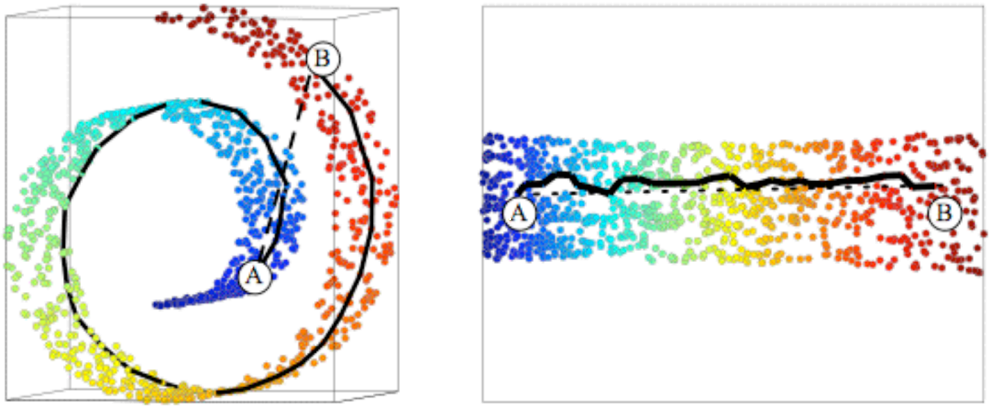
Isomap

- **Key idea:**

  Preserve geodesic distances as estimated along submanifold.

- **Algorithm in a nutshell:**

  Use geodesic instead of Euclidean distances in MDS.



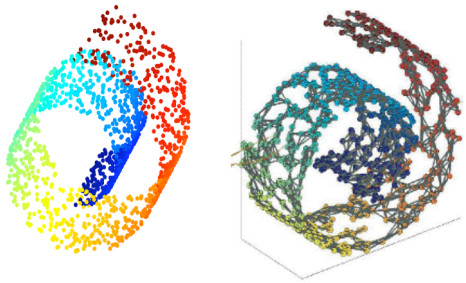
Step 1. Build adjacency graph.

- **Adjacency graph**

  Vertices represent inputs.
  Undirected edges connect neighbors.

- **Neighborhood selection**

  Many options: k-nearest neighbors, inputs within radius r, prior knowledge.



Graph is discretized approximation of submanifold.

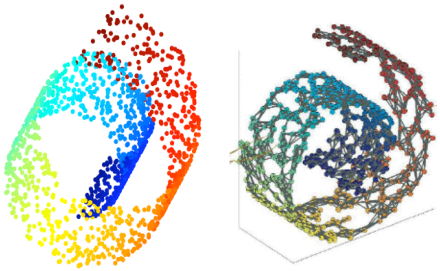
Building the graph

- **Computation**

  kNN scales naively as $O(n^2D)$.
  Faster methods exploit data structures.

- **Assumptions**

  1) Graph is connected.
  2) Neighborhoods on graph reflect neighborhoods on manifold.



No “shortcuts” connect different arms of swiss roll.

Step 2. Estimate geodesics.

- **Dynamic programming**

  Weight edges by local distances.
  Compute shortest paths through graph.

- **Geodesic distances**

  Estimate by lengths Δ_{ij} of shortest paths:
  denser sampling = better estimates.

- **Computation**

  Djikstra's algorithm for shortest paths
  scales as $O(n^2 \log n + n^2 k)$.

Step 3. Metric MDS

- **Embedding**

  Top d eigenvectors of Gram matrix yield embedding.

- **Dimensionality**

  Number of significant eigenvalues yield estimate of dimensionality.

- **Computation**

  Top d eigenvectors can be computed in $O(n^2 d)$.

Summary

- **Algorithm**

  1) k nearest neighbors

  2) shortest paths through graph

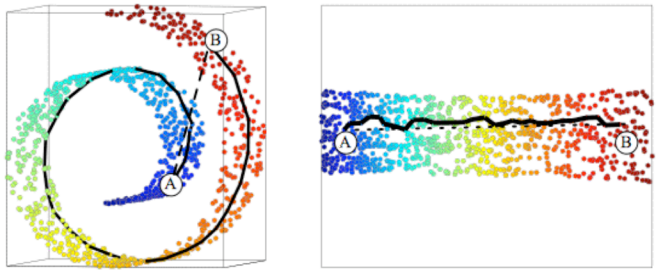
  3) MDS on geodesic distances

- **Impact**

  Much simpler than neural nets, Kohonen maps, etc. Does it work?

Examples

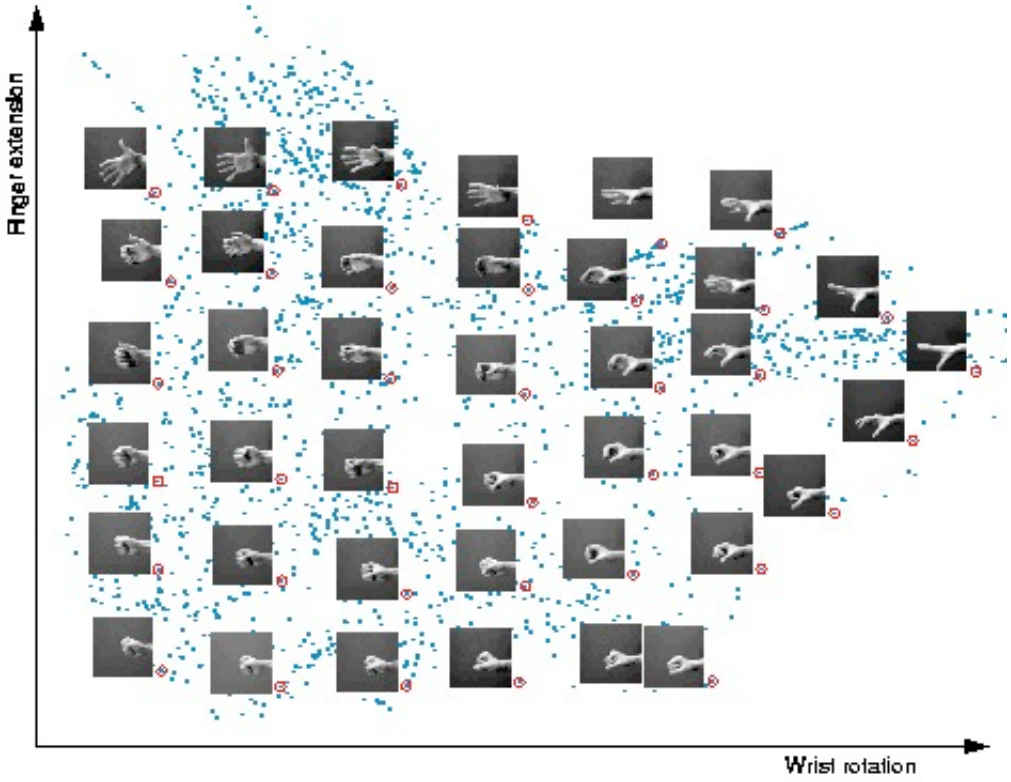
- **Swiss roll**



$n = 1024$
$k = 12$

- **Wrist images**

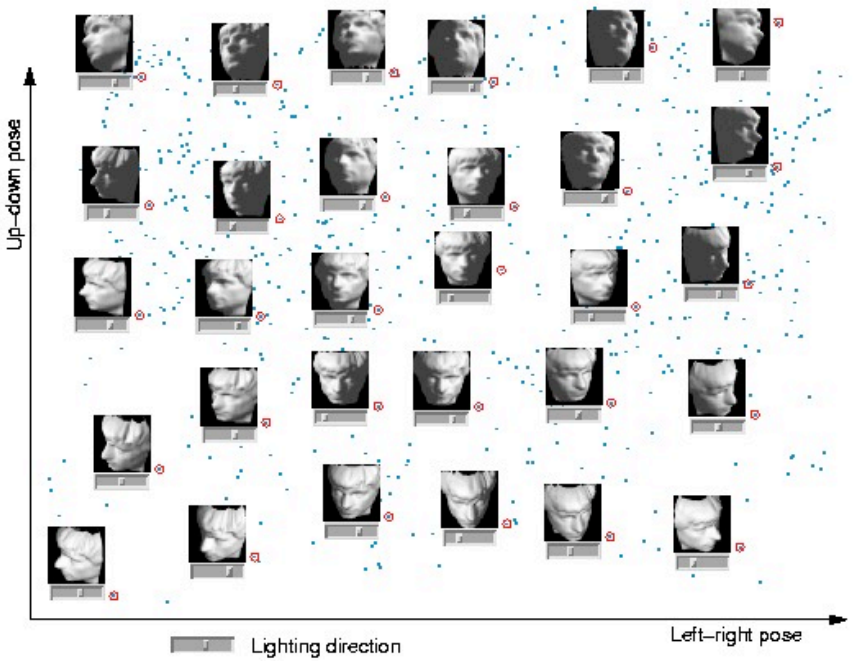
$n = 2000$
$k = 6$
$D = 64^2$



Examples

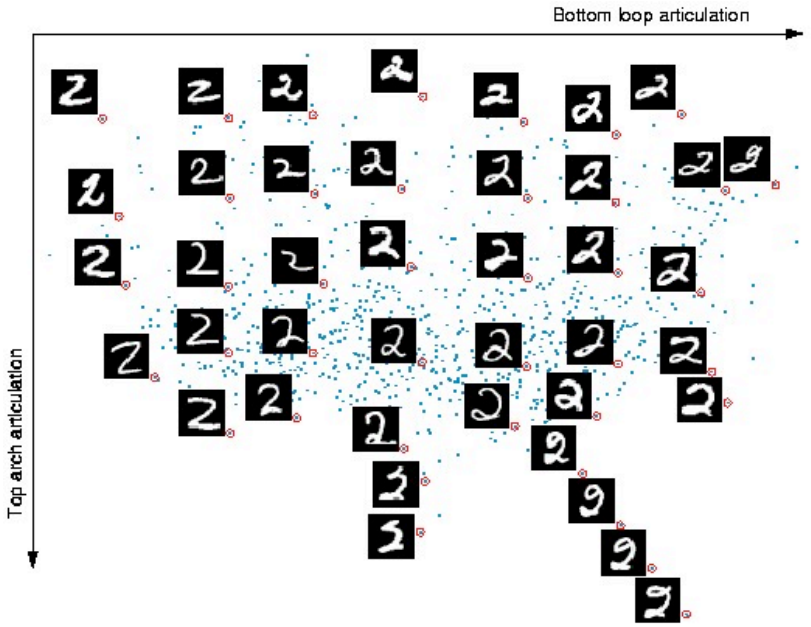
- **Face images**

$n = 698$
$k = 6$



- **Digit images**

$n = 1000$
$r = 4.2$
$D = 20^2$



Properties of Isomap

- **Strengths**
  - Polynomial-time optimizations
  - No local minima
  - Non-iterative (one pass thru data)
  - Non-parametric
  - Only heuristic is neighborhood size.
- **Weaknesses**
  - Sensitive to “shortcuts”
  - No immediate out-of-sample extension

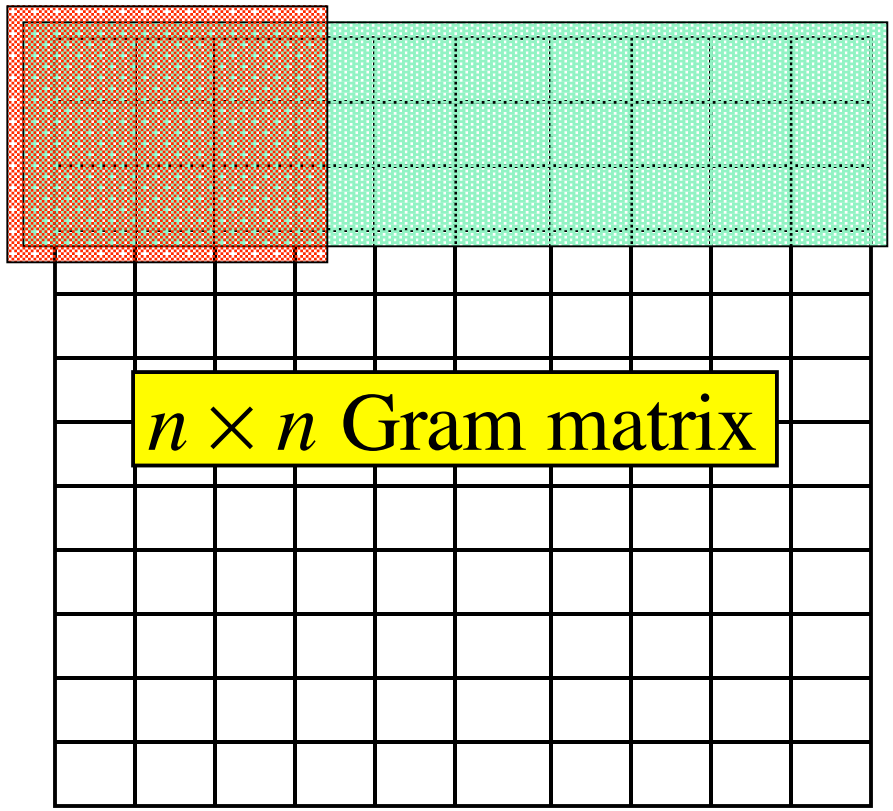
Large-scale applications

Problem:

Too expensive to compute all shortest paths and diagonalize full Gram matrix.

Solution:

Only compute shortest paths in green and diagonalize sub-matrix in red.



Landmark Isomap

(de Silva & Tenenbaum, 2003)

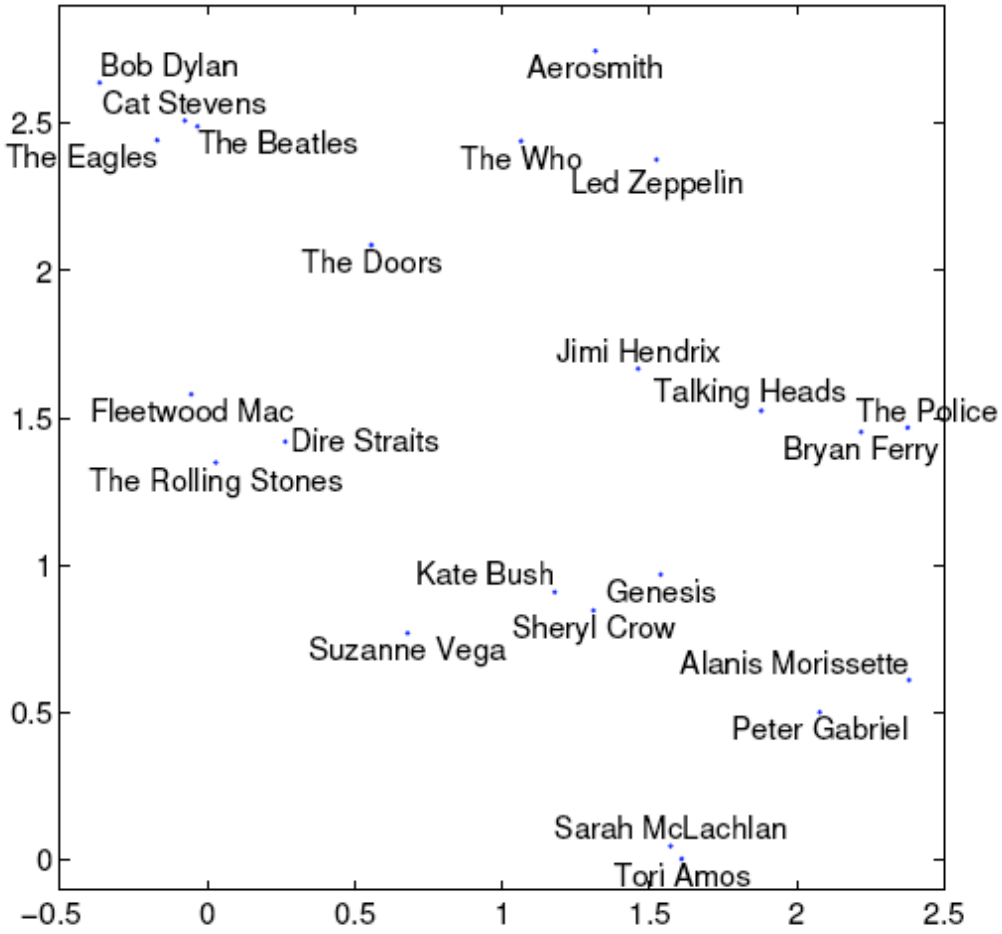
- **Approximation**
  - Identify subset of inputs as landmarks.
  - Estimate geodesics to/from landmarks.
  - Apply MDS to landmark distances.
  - Embed non-landmarks by triangulation.
  - Related to Nystrom approximation.
- **Computation**
  - Reduced by l/n for $l<n$ landmarks.
  - Reconstructs large Gram matrix from thin rectangular sub-matrix.

Example

Embedding of sparse music similarity graph

$n = 267\text{K}$
$e = 3.22\text{M}$
$\ell = 400$
$\tau = 6$ minutes

(Platt, 2004)



Theoretical guarantees

- **Asymptotic convergence**

  For data sampled from a submanifold that is isometric to a convex subset of Euclidean space, Isomap will recover the subset up to rotation & translation.

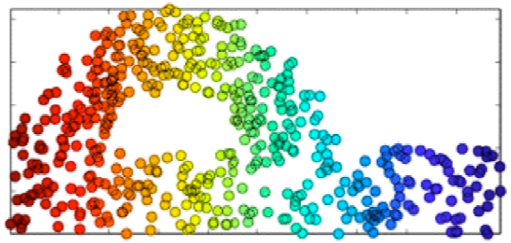
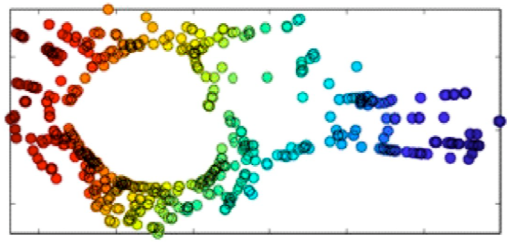
  (Tenenbaum et al; Donoho & Grimes)

- **Convexity assumption**

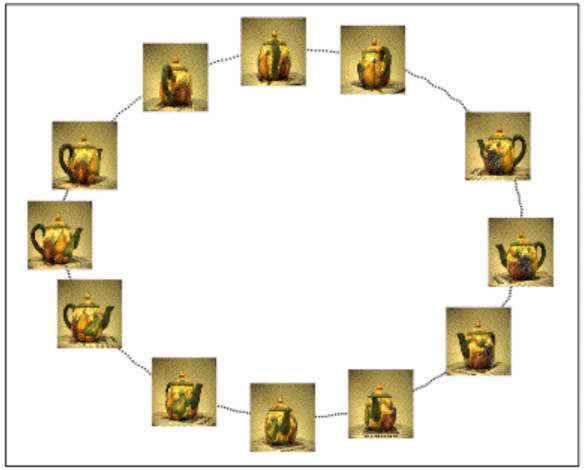
  Geodesic distances are not estimated correctly for manifolds with holes…

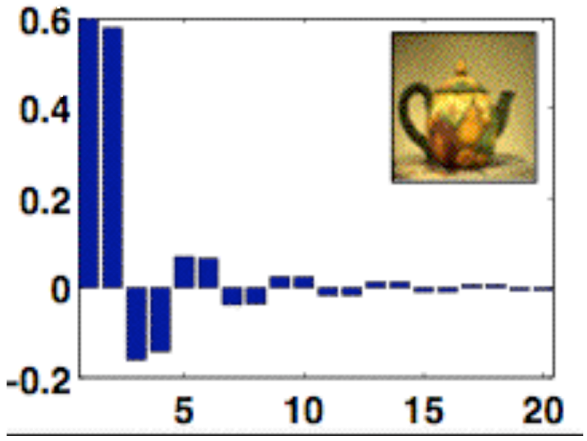
Connected but not convex

- 2d region with hole

input   Isomap

- Images of 360º rotated teapot



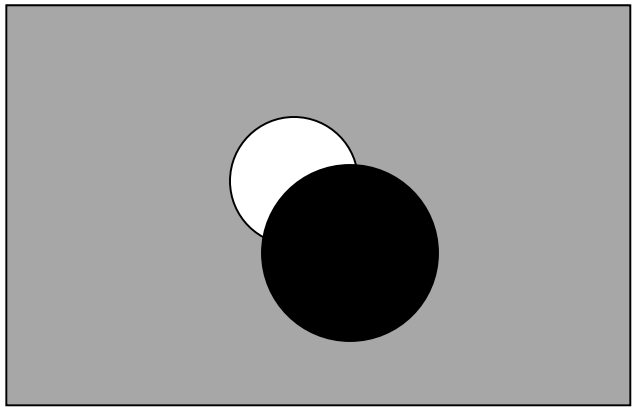


eigenvalues of Isomap

Connected but not convex

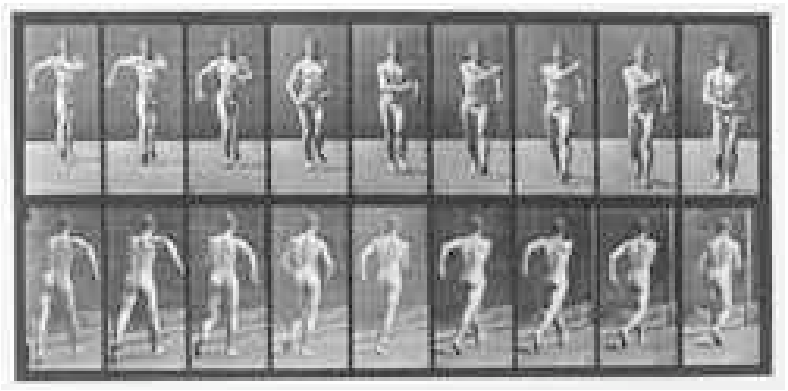
- **Occlusion**

  Images of two disks, one occluding the other.

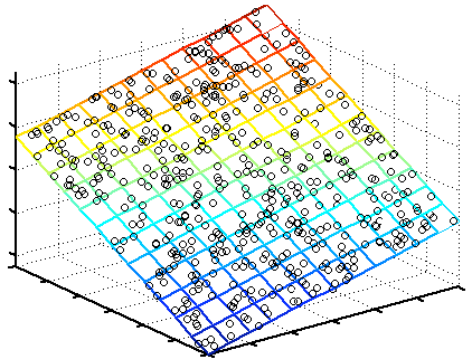


- **Locomotion**

  Images of periodic gait.



Linear vs nonlinear



What computational price must we pay for nonlinear dimensionality reduction?

Nonlinear dimensionality reduction since 2000…

These strengths and weaknesses are typical of graph-based spectral methods for dimensionality reduction.

Properties of Isomap

- **Strengths**
  - Polynomial-time optimizations
  - No local minima
  - Non-iterative (one pass thru data)
  - Non-parametric
  - Only heuristic is neighborhood size.
- **Weaknesses**
  - Sensitive to “shortcuts”
  - No out-of-sample extension

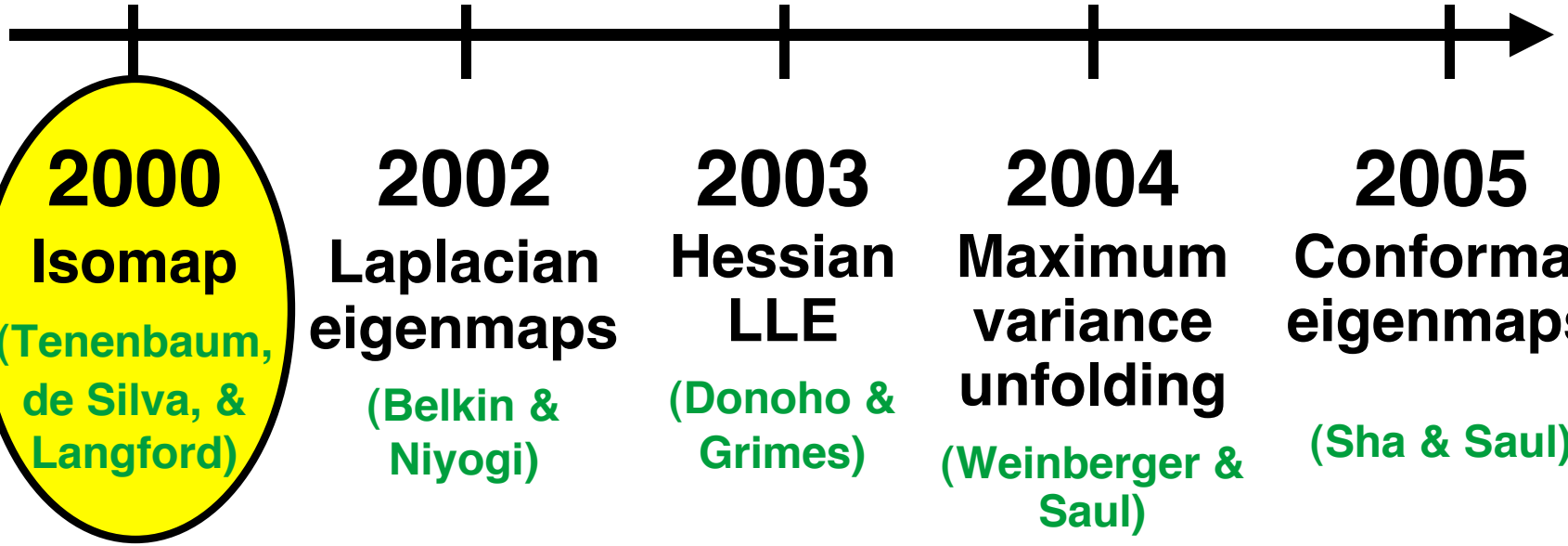
Spectral Methods

- **Common framework**
  1) Derive sparse graph from kNN.
  2) Derive matrix from graph weights.
  3) Derive embedding from eigenvectors.
- **Varied solutions**

  Algorithms differ in step 2.

  Types of optimization: shortest paths, least squares fits, semidefinite programming.

Algorithms


2000
Isomap
(Tenenbaum, de Silva, & Langford)
2002
Laplacian eigenmaps
(Belkin & Niyogi)
2003
Hessian LLE
(Donoho & Grimes)
2004
Maximum variance unfolding
(Weinberger & Saul)
(Sun, Boyd, Xiao, & Diaconis)
2005
Conformal eigenmaps
(Sha & Saul)
Locally Linear Embedding
(Roweis & Saul)

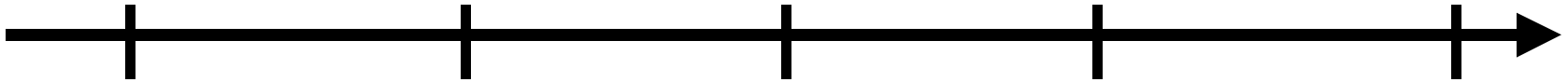
Outline

- **Part 1 - linear versus graph-based methods**
- **Part 2 - sparse matrix methods**
- **Part 3 - semidefinite programming**
- **Part 4 - kernel methods**
- **Part 5 - parting thoughts**

What's new in Part 2

- **MDS and Isomap**
  - preserve global pairwise distances
  - construct large, dense matrices
  - compute top eigenvectors
- **"Local" methods**
  - preserve local geometric relationships
  - construct large, sparse matrices
  - compute bottom eigenvectors

Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

2004
Maximum variance unfolding
(Weinberger & Saul)
(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal eigenmaps
(Sha & Saul)

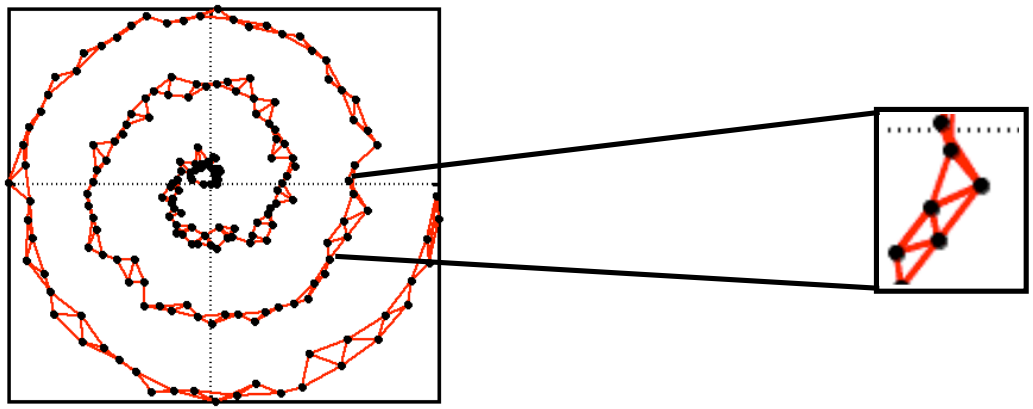
Locally Linear Embedding
(Roweis & Saul)

Locally linear embedding

- **Steps**
  1. Nearest neighbor search.
  2. Least squares fits.
  3. Sparse eigenvalue problem.
- **Properties**
  - Obtains highly nonlinear embeddings.
  - Not prone to local minima.
  - Sparse graphs yield sparse problems.

Step 2. Compute weights.

- Characterize local geometry of each neighborhood by weights W_{ij}.



- Compute weights by reconstructing each input (linearly) from neighbors.

Linear reconstructions

- **Local linearity**

  Assume neighbors lie on locally linear patches of a low dimensional manifold.

- **Reconstruction errors**

  Least squared errors should be small:

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

Least squares fits

- **Local reconstructions**

  Choose weights to minimize:

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

- **Constraints**

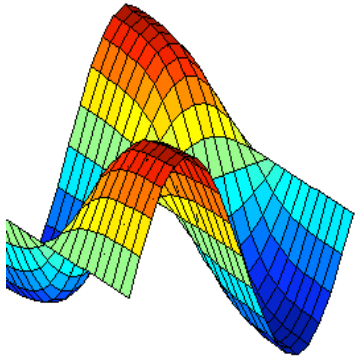
  Nonzero W_{ij} only for neighbors.
  Weights must sum to one:

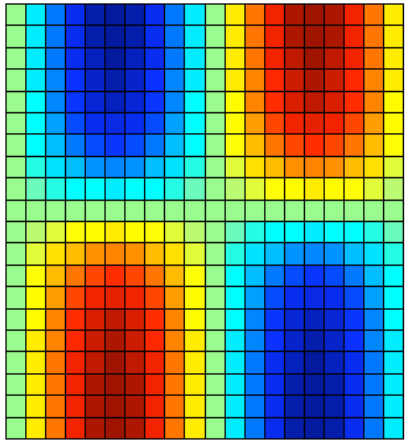
$$\sum_j W_{ij} = 1$$

- **Local invariance**

  Optimal weights W_{ij} are invariant to rotation, translation, and scaling.

Symmetries





- **Local linearity**

  If each neighborhood map looks like a translation, rotation, and rescaling...

- **Local geometry**

  …then these transformations do not affect the weights W_{ij}: they remain valid.

Thought experiment

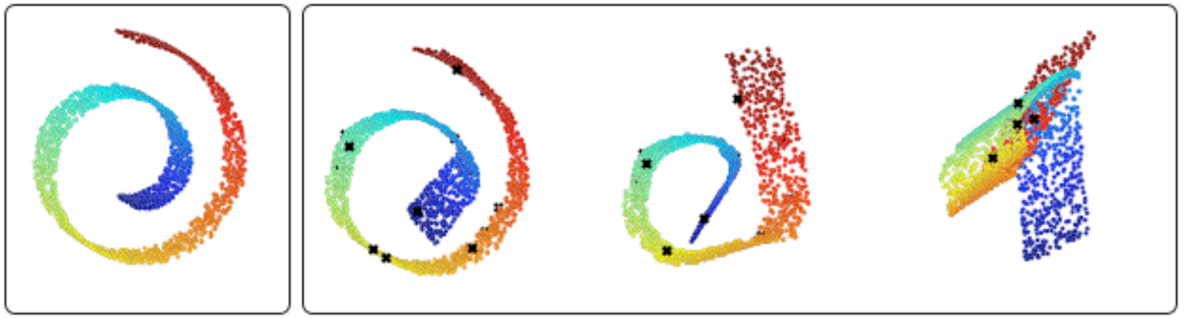
- **Reconstruction from landmarks**

  Clamp subset of inputs ("landmarks"), then reconstruct others by minimizing:

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

with respect to $\vec{x}_i$!

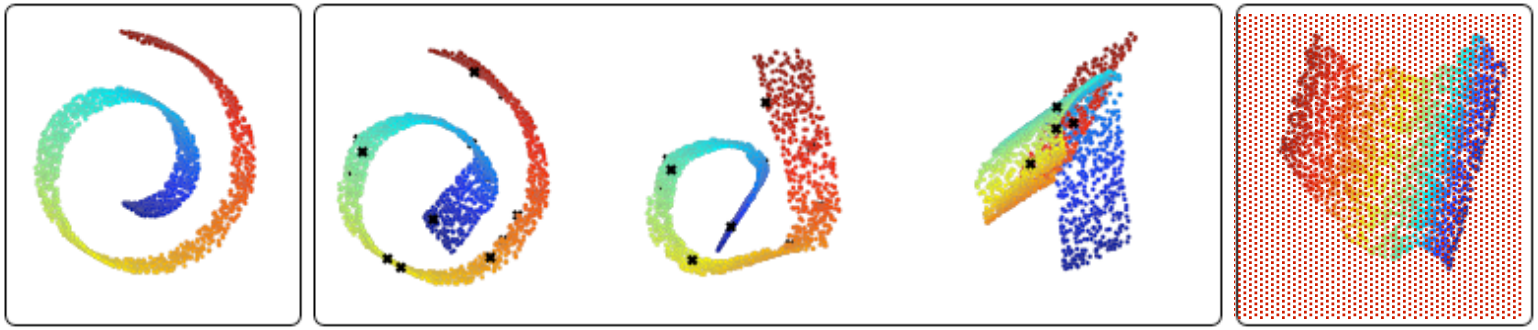
n=2000 inputs



Number of landmarks: L = 15, L = 10, L = 5

Thought experiment (con't)

- **Locally linear reconstruction**
  - Very accurate for sufficiently large number of landmarks.
  - Increasingly linearized with decreasing number of landmarks.



Number of landmarks: L = 15, L = 10, L = 5 , L = 0 ?

Step 3. "Linearization"

- **Low dimensional representation**

  Map inputs to outputs: $\vec{x}_i \in \Re^D$ to $\vec{y}_i \in \Re^d$

- **Minimize reconstruction errors.**

  Optimize outputs for fixed weights:

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

- **Constraints**

  Center outputs on origin: $\sum_i \vec{y}_i = \vec{0}$

  Impose unit covariance matrix: $\frac{1}{N} \sum_i \vec{y}_i \vec{y}_i^T = I_d.$

Sparse eigenvalue problem

- **Quadratic form**

$$\Psi(y) = \sum_{ij} \Psi_{ij} \left(\vec{y}_i \bullet \vec{y}_j \right) \text{ with } \Psi = (I - W)^T (I - W)$$

- **Rayleigh-Ritz quotient**

  Optimal embedding given by bottom d+1 eigenvectors.

- **Solution**

  Discard bottom eigenvector [1 1 … 1]. Other eigenvectors satisfy constraints.

Summary of LLE

- **Three steps**
  1. **Compute k-nearest neighbors.**
  2. **Compute weights W_{ij}.**
  3. **Compute outputs $\vec{y}_i$.**
- **Optimizations**

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

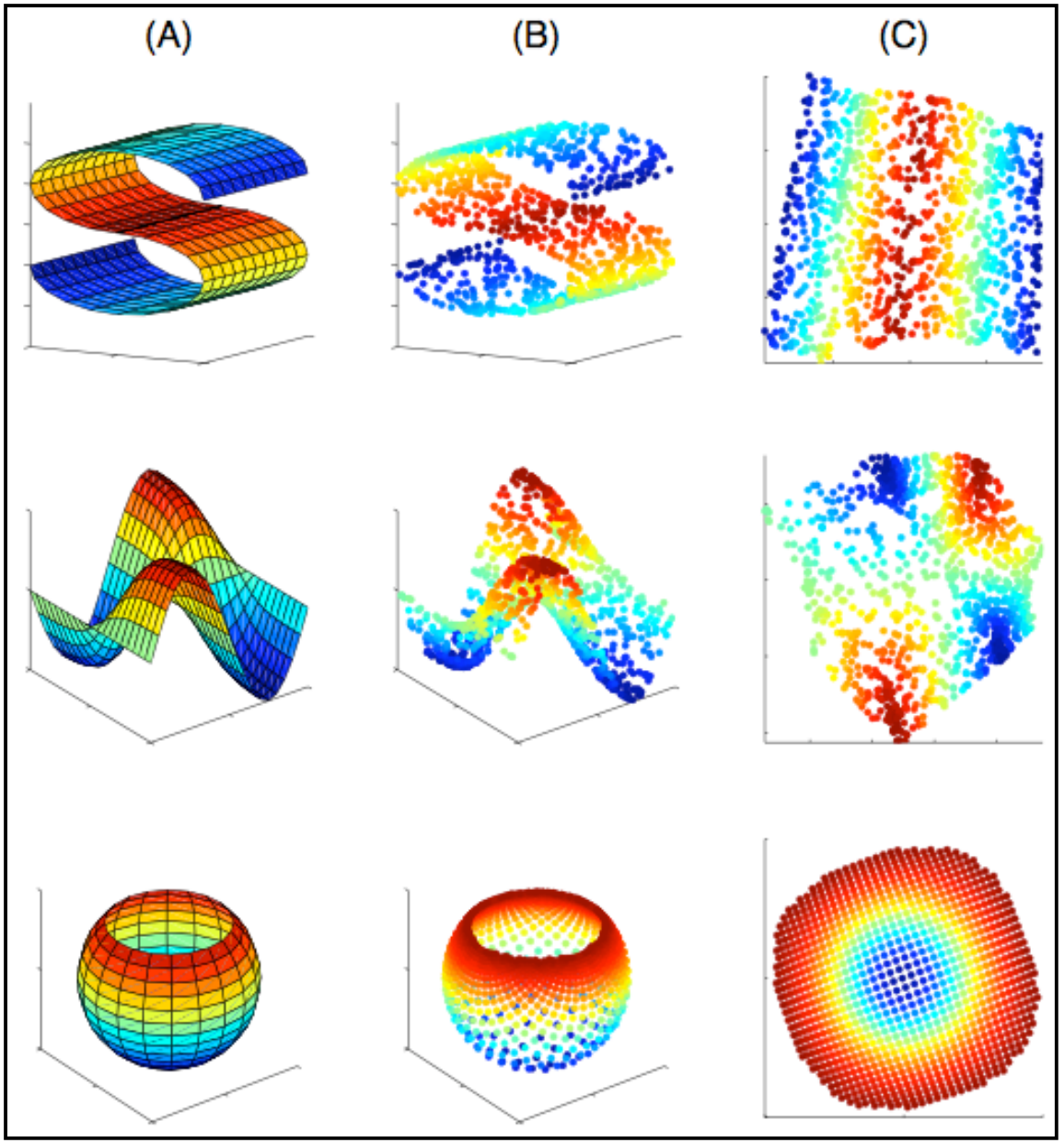
$$\Psi(y) = \sum_i \left| \vec{y}_i - \sum_j W_{ij} \vec{y}_j \right|^2$$

Surfaces

N=1000
inputs

k=8
nearest
neighbors

D=3
d=2
dimensions



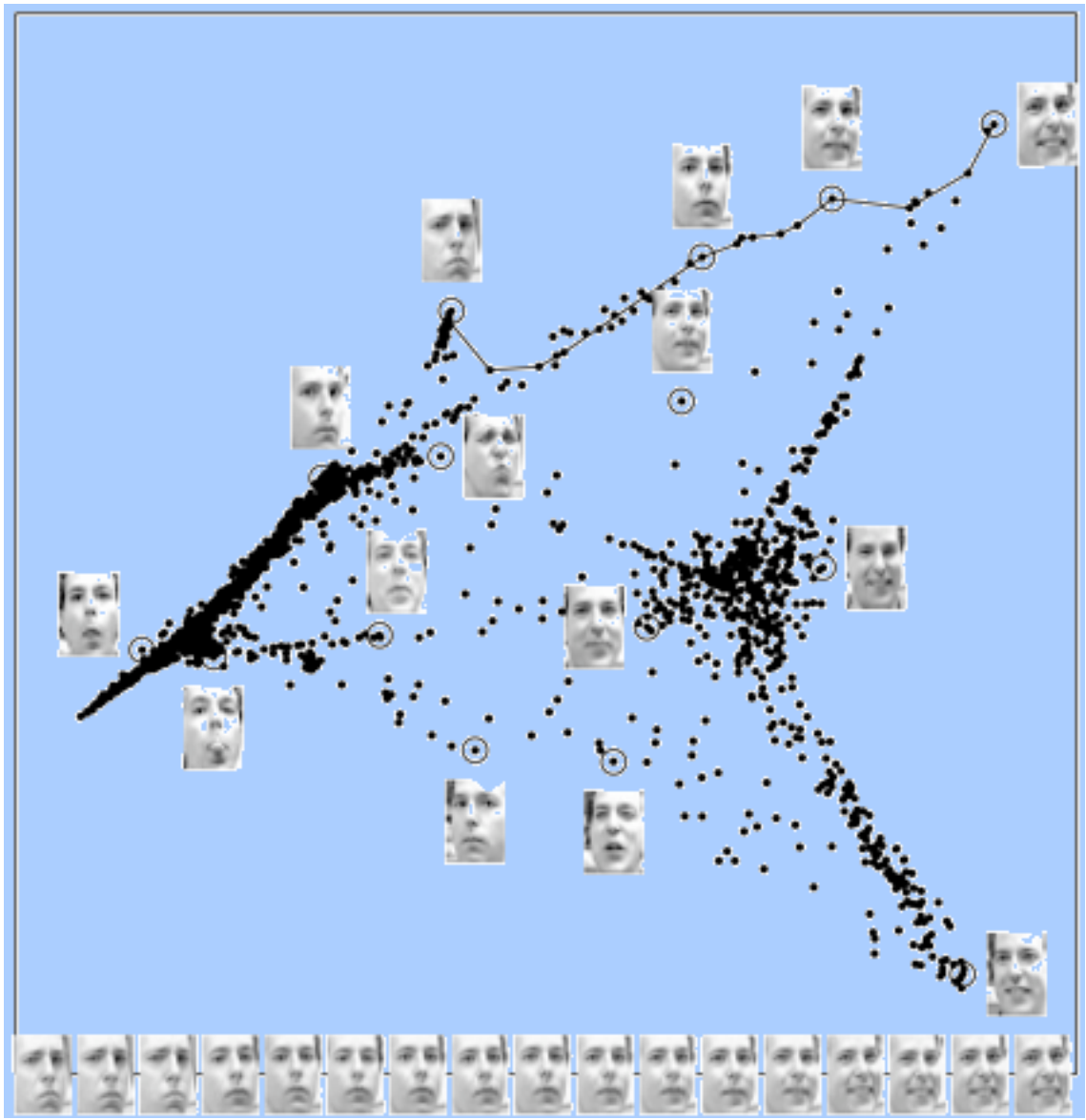
Pose and expression

N=1965
images

k=12
nearest neighbors

D=560
pixels

d=2
(shown)



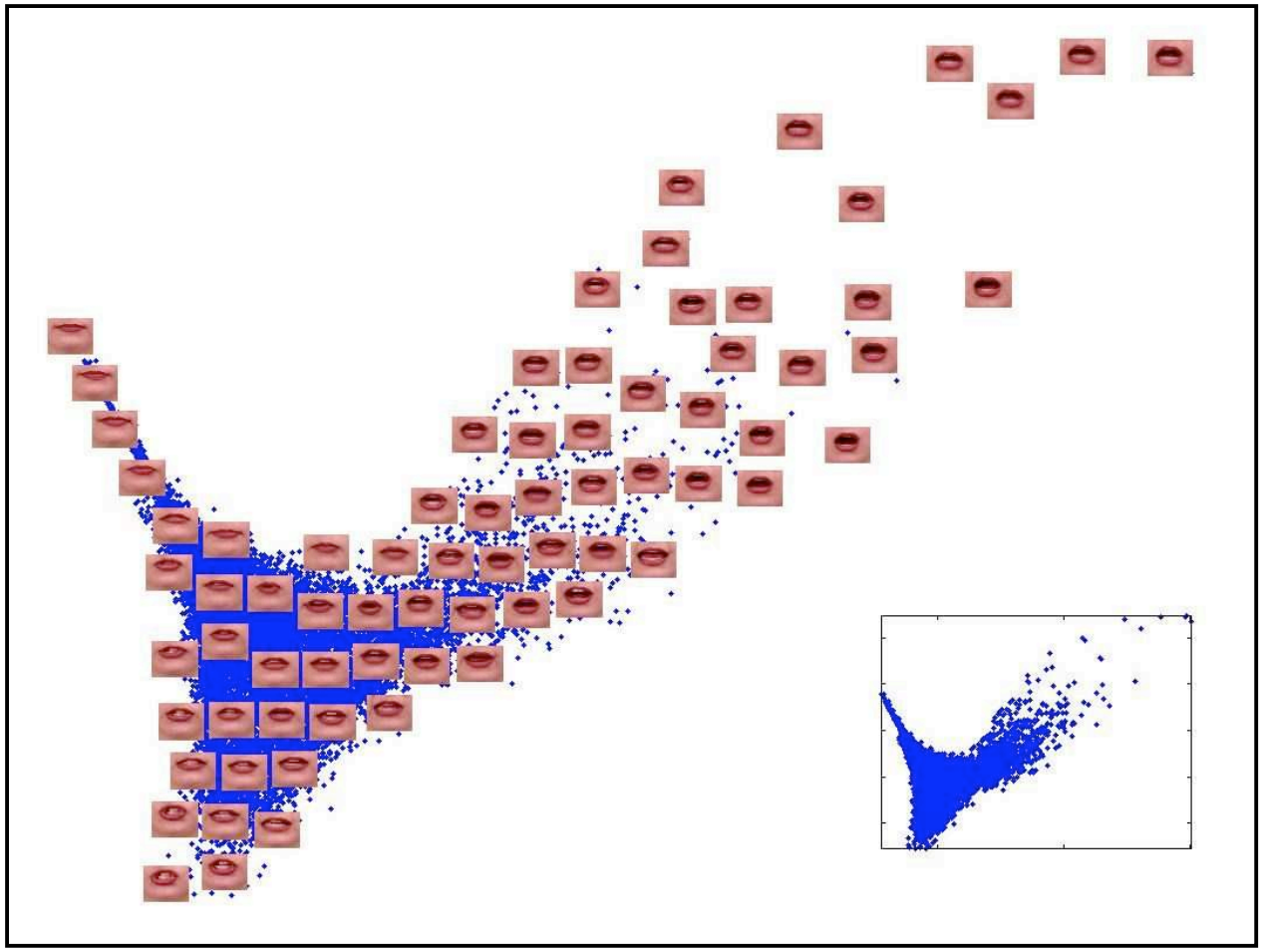
Lips

N=15960
images

K=24
neighbors

D=65664
pixels

d=2
(shown)

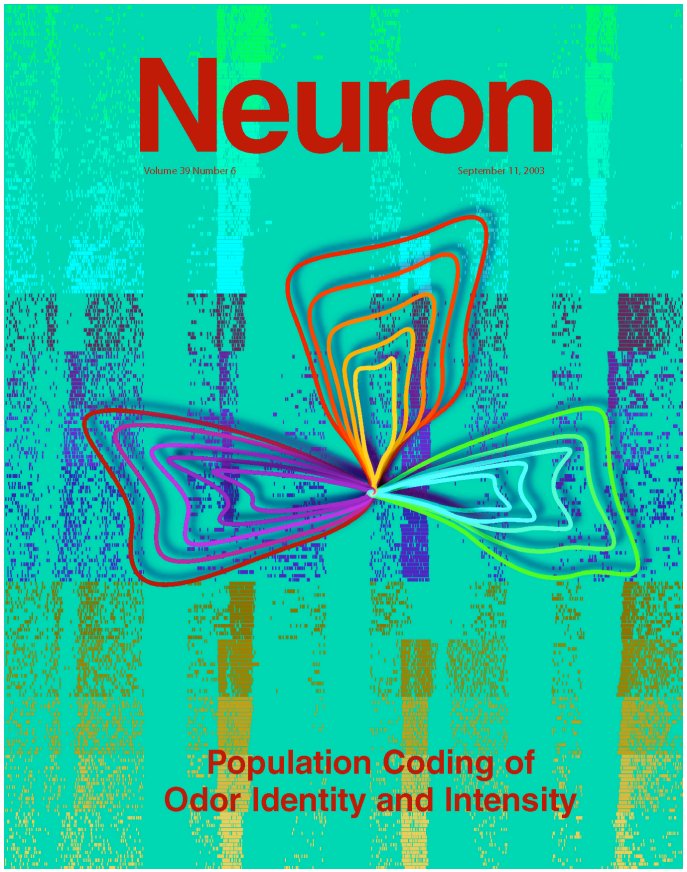


Exploratory data analysis

- **Spike patterns**

  In response to odor stimuli, neuronal spike patterns reveal intensity-specific trajectories on identity-specific surfaces (from LLE).

  (Stopfer et al, 2003)



Properties of LLE

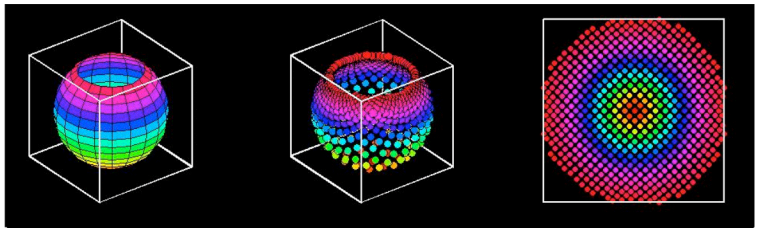
- **Strengths**
  - Polynomial-time optimizations
  - No local minima
  - Non-iterative (one pass thru data)
  - Non-parametric
  - Only heuristic is neighborhood size.
- **Weaknesses**
  - Sensitive to “shortcuts”
  - No out-of-sample extension
  - No estimate of dimensionality

LLE versus Isomap

- **Many similarities**
  - Graph-based, spectral method
  - No local minima
- **Essential differences**
  - Does not estimate dimensionality
  - No theoretical guarantees

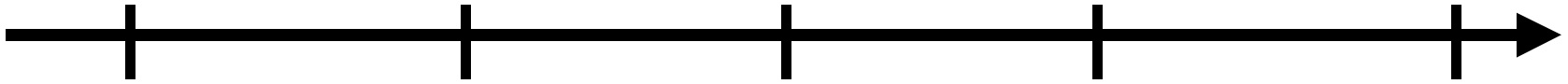
  \+ Constructs sparse vs dense matrix

  ? Preserves weights vs distances



Conformal mapping

Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

2004
Maximum variance unfolding
(Weinberger & Saul)
(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal eigenmaps
(Sha & Saul)

Locally Linear Embedding
(Roweis & Saul)

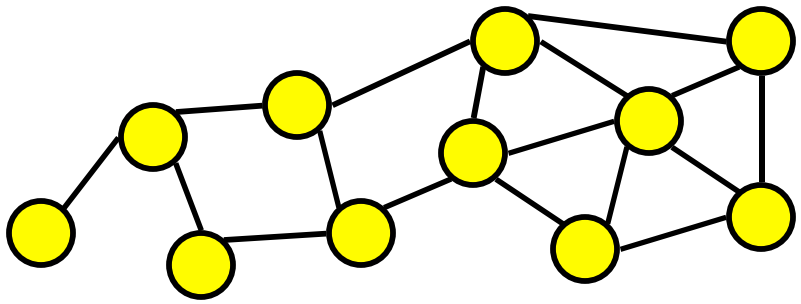
Laplacian eigenmaps

- **Key idea:**

  Map nearby inputs to nearby outputs, where nearness is encoded by graph.

- **Physical intuition:**

  Find lowest frequency vibrational modes of a mass-spring system.



Summary of algorithm

- **Three steps**

1. **Identify k-nearest neighbors**
2. **Assign weights to neighbors:**

$$W_{ij} = 1 \;\text{ or }\; W_{ij} = \exp\left(-\beta \left\| \vec{x}_i - \vec{x}_j \right\|^2\right)$$

3. **Compute outputs by minimizing:**

$$\Psi(y) = \sum_{ij} \frac{W_{ij} \left\| \vec{y}_i - \vec{y}_j \right\|^2}{\sqrt{D_{ii} D_{jj}}} \;\text{ where }\; D_{ii} = \sum_j W_{ij}$$

(sparse eigenvalue problem as in LLE)

Laplacian vs LLE

- **More similar than different**
  - Graph-based, spectral method
  - Sparse eigenvalue problem
  - Similar results in practice
- **Essential differences**
  - Preserves locality vs local linearity
  - Uses graph Laplacian

$$L = D - W \quad \text{(unnormalized)}$$
$$\mathcal{L} = I - D^{-1/2} W D^{-1/2} \quad \text{(normalized)}$$

Analysis on Manifolds

- **Laplacian in $\mathcal{R}^d$**

  Function $f(x_1,x_2,\ldots,x_d)$ has Laplacian:

$$\Delta f = -\sum_i \frac{\partial^2 f}{\partial x_i^2}$$

- **Manifold Laplacian**

  Change is measured along tangent space of manifold.

- **Stokes theorem**

$$\int_M \|\nabla f\|^2 = \int_M f \Delta f$$

Spectral graph theory

- **Manifolds and graphs**

  Weighted graph is discretized representation of manifold.

- **Laplacian operators**

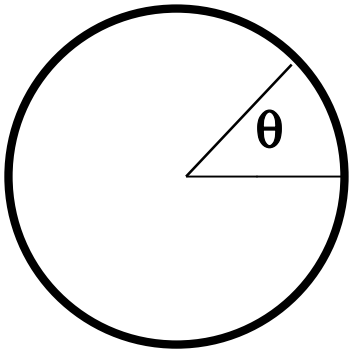
  Laplacian measures smoothness of functions over manifold (or graph).

$$\int_M \|\nabla f\|^2 = \int_M f \Delta f \quad \text{(manifold)}$$

$$\sum_{ij} W_{ij} \left(f_i - f_j \right)^2 = f^{\mathrm{T}} L f \quad \text{(graph)}$$

Example: S^1 (the circle)

- **Continuous**
  - **Eigenfunctions of Laplacian are basis for periodic functions on circle, ordered by smoothness.**
  - **Eigenvalues measure smoothness.**

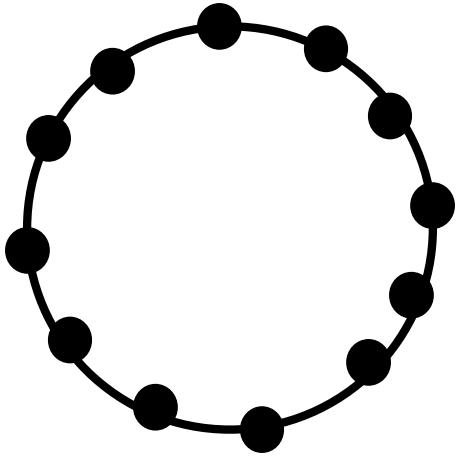


$$-\frac{\partial^2 f_m}{\partial^2 \theta} = \lambda_m f_m(\theta)$$

$$f_m(\theta) = \left\{ \begin{matrix} \sin(m\theta) \\ \cos(m\theta) \end{matrix} \right\} \text{ with } \lambda_m = m^2$$

Example: S^1 (the circle)

- **Discrete (n equally spaced points)**
  - **Eigenvectors of graph Laplacian are discrete sines and cosines.**
  - **Eigenvalues measure smoothness.**



Graph embedding from Laplacian eigenmaps:

$$\vec{y}_k = (\cos(2\pi k/n), \sin(2\pi k/n))$$

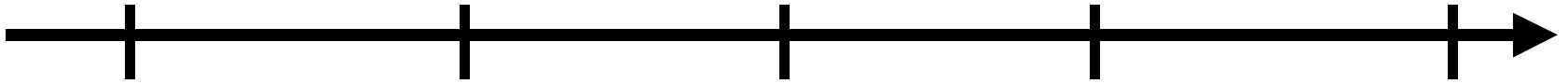
A critical view…

- **LLE and Laplacian eigenmaps**
  - Construct quadratic form over functions on graph.
  - Take d lowest cost (but non-constant) functions as manifold coordinates.
- **Theoretical guarantees?**
  - When do bottom eigenvectors give the "right answer"?
  - Depends on the definition of the "right answer"…

A critical view (con't)

- **Assumption**
  - Sample inputs from manifold that is isometrically embedded in $\mathcal{R}^D$.
  - Assume manifold is locally isometric to an open subset of $\mathcal{R}^d$, where $d < D$.
- **Hypothesis**
  - Isomap's top d eigenvectors recover parameterization for convex subsets.
  - Can bottom d (nonzero) eigenvectors of sparse matrix method do better?

Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

2004
Maximum variance unfolding
(Weinberger & Saul)
(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal components analysis
(Sha & Saul)

Locally Linear Embedding
(Roweis & Saul)

Hessian LLE

- **Assumption**

  Data manifold M is locally isometric to open, connected subset of $\mathcal{R}^d$.

- **Key ideas**
  - Define Hessian via orthogonal coordinates on tangent planes of M.
  - Quadratic form $\Psi(f)$ averages Frobenius norm of Hessian over M.

$$\Psi(f) = \int_M \| H_f(m) \|^2 \, dm$$

$$\int_M \| \nabla f \|^2 = \int_M f \Delta f$$

Hessian LLE

$$\Psi(f) = \int_M \| H_f(m) \|^2 \, dm$$

~~$$\int_M \| \nabla f \|^2 = \int_M f \Delta f$$~~

- **Key ideas (con't)**
  - **Every function with vanishing Hessian is linear. (Not so for Laplacian.)**
  - **Bottom eigenfunctions in null space of $H(f)$ yield isometric coordinates.**
  - **Graph-based discretization yields algorithm.**

Hessian LLE

- **Three steps**
  1. **Construct graph from kNN.**
  2. **Estimate Hessian operator at each data point.**
  3. **Compute bottom eigenvectors of sparse quadratic form.**

$$\Psi(f) = \int_M \| H_f(m) \|^2 \, dm$$

- **What's new?**

  (1) and (3) are same as before.
  (2) estimates Hessian. (Details omitted.)

Relation to previous work

- **Algorithm variant of LLE**

  Replaces least squares fits in LLE by estimation of Hessian.

- **Conceptual variant of Laplacian**

  Substitutes Frobenius norm of Hessian for norm of gradient vector.

- **Sparse matrix variant of Isomap**

  Also looks for isometric coordinates on data manifold.

Theoretical guarantees

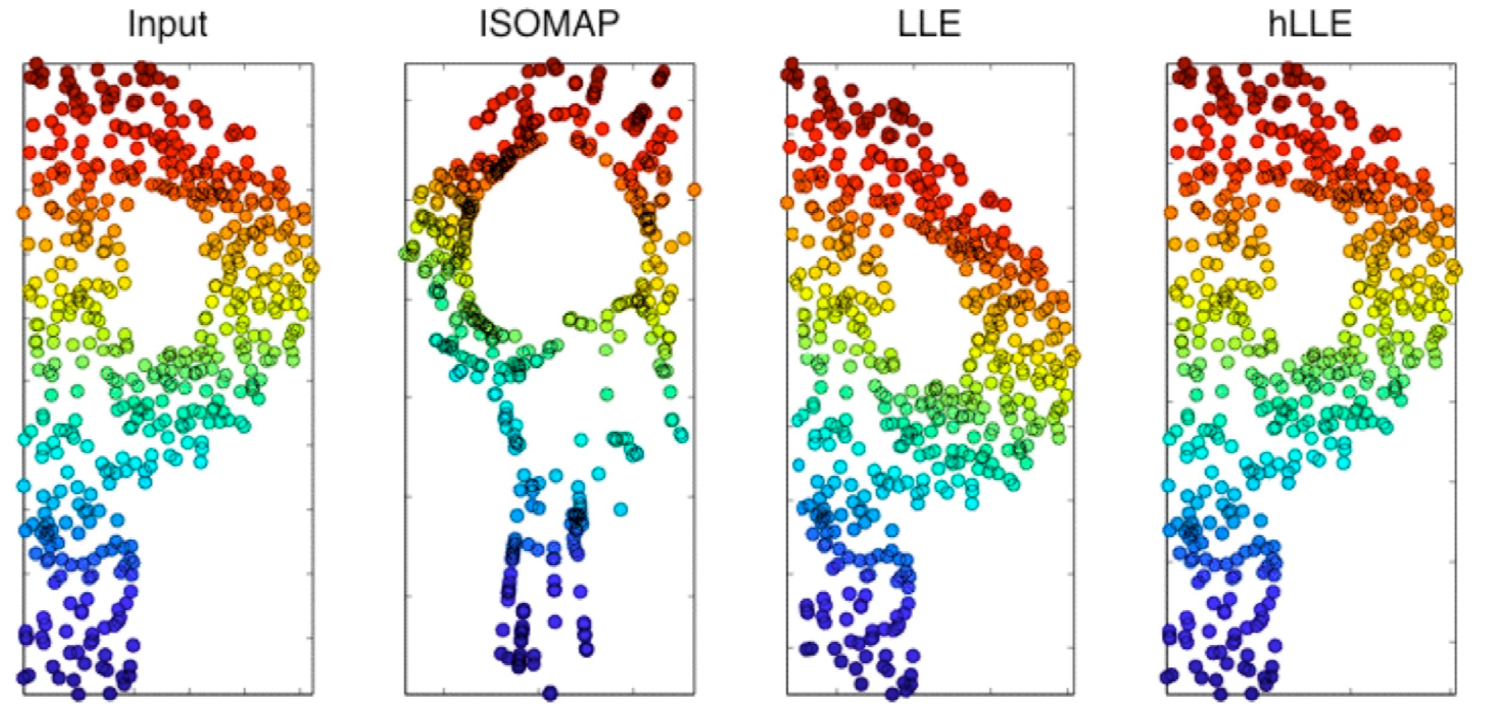
- **Asymptotic convergence**

  For data sampled from a submanifold that is isometric to an open, connected subset of Euclidean space, hLLE will recover the subset up to rigid motion.

- **No convexity assumption**

  Convergence is obtained for a larger class of manifolds than Isomap.

Connected but not convex

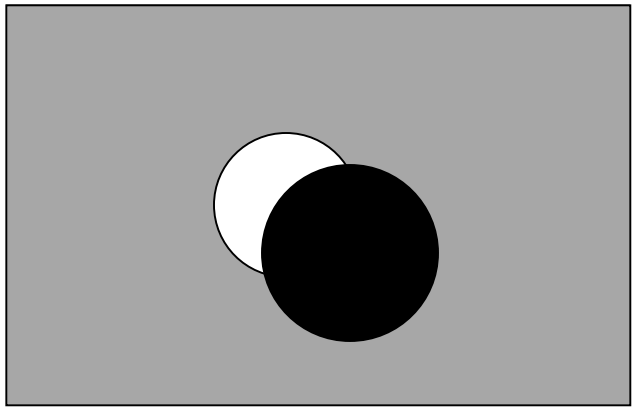


Hessian LLE yields an isometric embedding, but not Isomap or LLE.

Connected but not convex

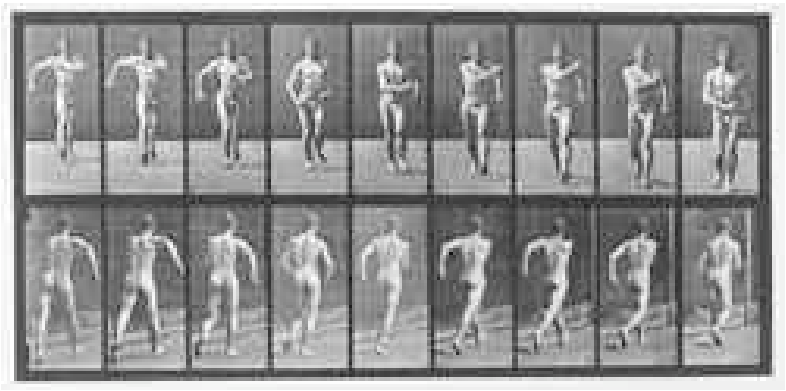
- **Occlusion**

  Images of two disks, one occluding the other.

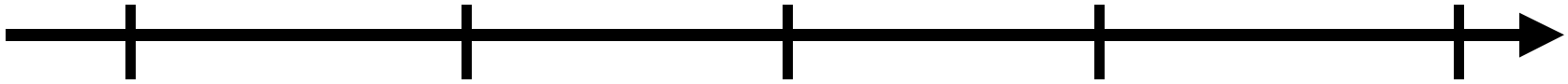


- **Locomotion**

  Images of periodic gait.



Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

Locally Linear Embedding
(Roweis & Saul)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

What is left to do?

Problem solved?

- For manifolds without "holes":
  - Isomap with asymptotic guarantees
  - landmark Isomap for large data sets
- More generally:
  - hLLE with asymptotic guarantees?
  - sparse matrix method should scale well to large data sets?

(If it seems too good to be true, it usually is…)

Flies in the ointment

- **How to estimate dimensionality?**

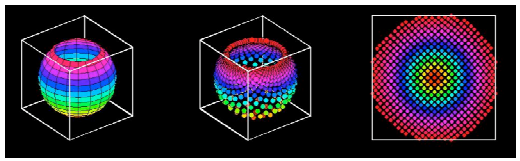
  Revealed by eigenvalue gap of Isomap, but specified in advance for (h)LLE.

- **How to compute eigenvectors?**

  Bottom eigenvalues are very closely spaced for large data sets.

- **Must we preserve distances?**

  Preserving distances may hamper dimensionality reduction.



Computing eigenvectors

- **Numerical difficulty**

  Inversely proportional to spacing between adjacent eigenvalues.

- **Scaling to large data sets**

  Bottom eigenvalue spacing shrinks with increased sampling of manifold.

- **Conundrum**

  Finer discretization of manifold trades off with ability to resolve eigenvectors.

Can we combine strengths of:

- **Isomap**

  Eigenvalues reveal dimensionality.
  Landmark version scales well.
  Numerically stable.

- **hLLE**

  Solves sparse eigenvalue problem.
  Handles manifolds with “holes”.

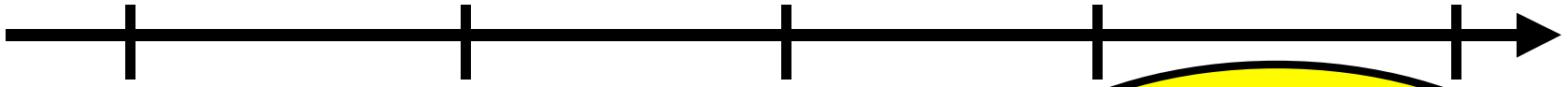
- **LLE and Laplacian eigenmaps**

  Aggressive dimensionality reduction.
  Locality vs distance-preserving maps.

Outline

- **Part 1 - linear versus graph-based methods**
- **Part 2 - sparse matrix methods**
- **Part 3 - semidefinite programming**
- **Part 4 - kernel methods**
- **Part 5 - parting thoughts**

Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

Locally Linear Embedding
(Roweis & Saul)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

2004
Maximum variance unfolding
(Weinberger & Saul)
(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal eigenmaps
(Sha & Saul)

Semidefinite Programming

Semidefinite program (SDP)

- **Definition**

  An SDP is a linear program with an extra constraint that a matrix whose elements are linear in the unknowns must be positive semidefinite (PSD).

- **Example**

  Minimize $\vec{a} \bullet \vec{x}$ subject to:

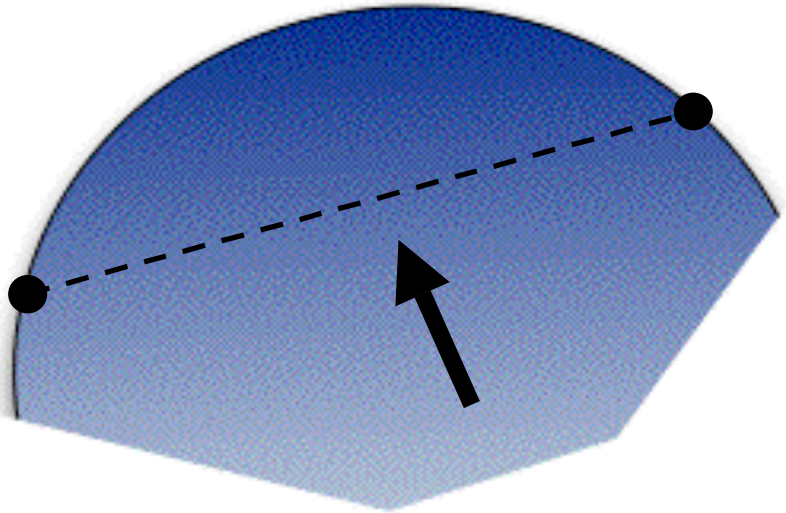
  (i) $\vec{w}_i \cdot \vec{x} > 0$ for $i = 1, 2, \ldots, c$

  (ii) $x_1 M_1 + x_2 M_2 + \ldots x_d M_d$ is PSD.

Convex optimization

- **Constraints**

  Linear and PSD constraints are convex.



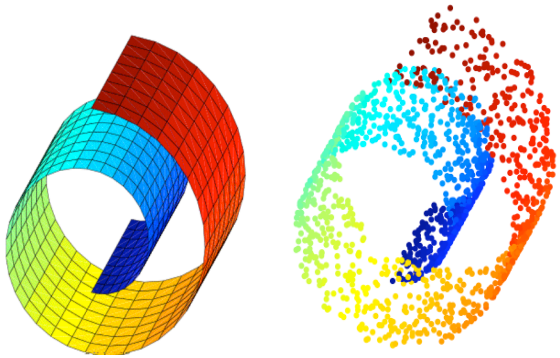
- **Cost function**

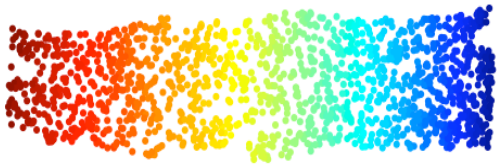
  Linear and bounded.

Efficient (poly-time) algorithms exist to compute global minimum.

What does

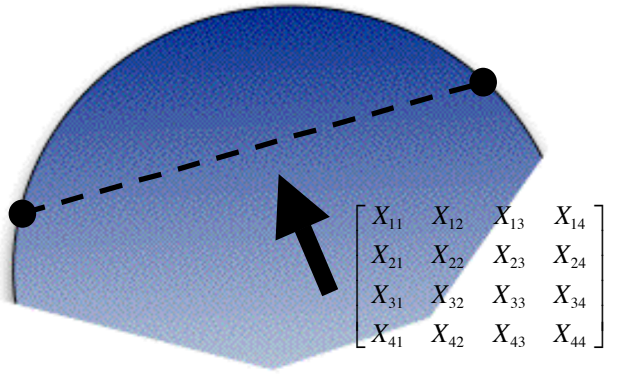
nonlinear dimensionality reduction





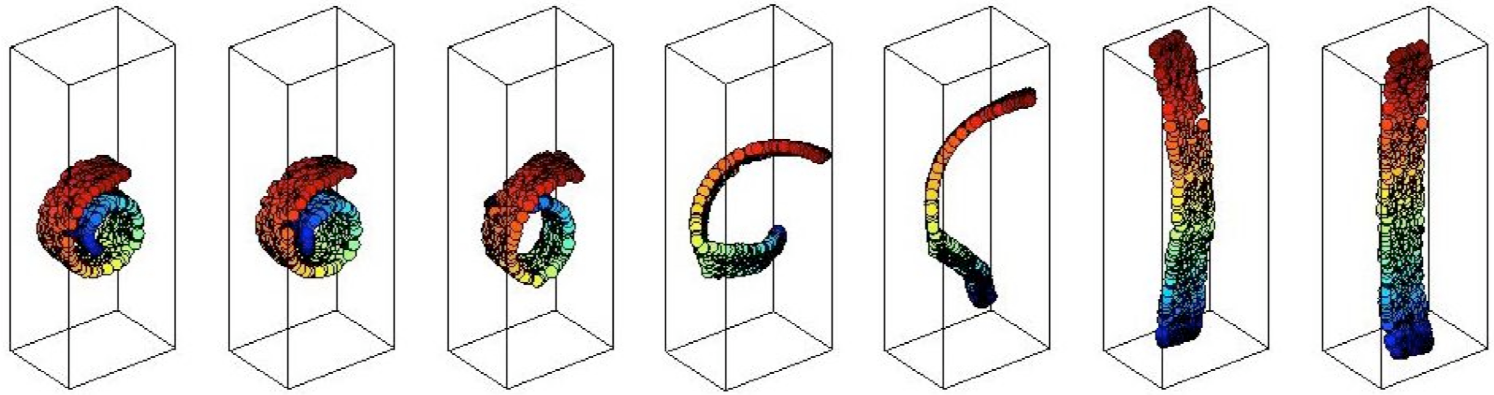
have to do with

semidefinite programming?



How to unfold a data set?

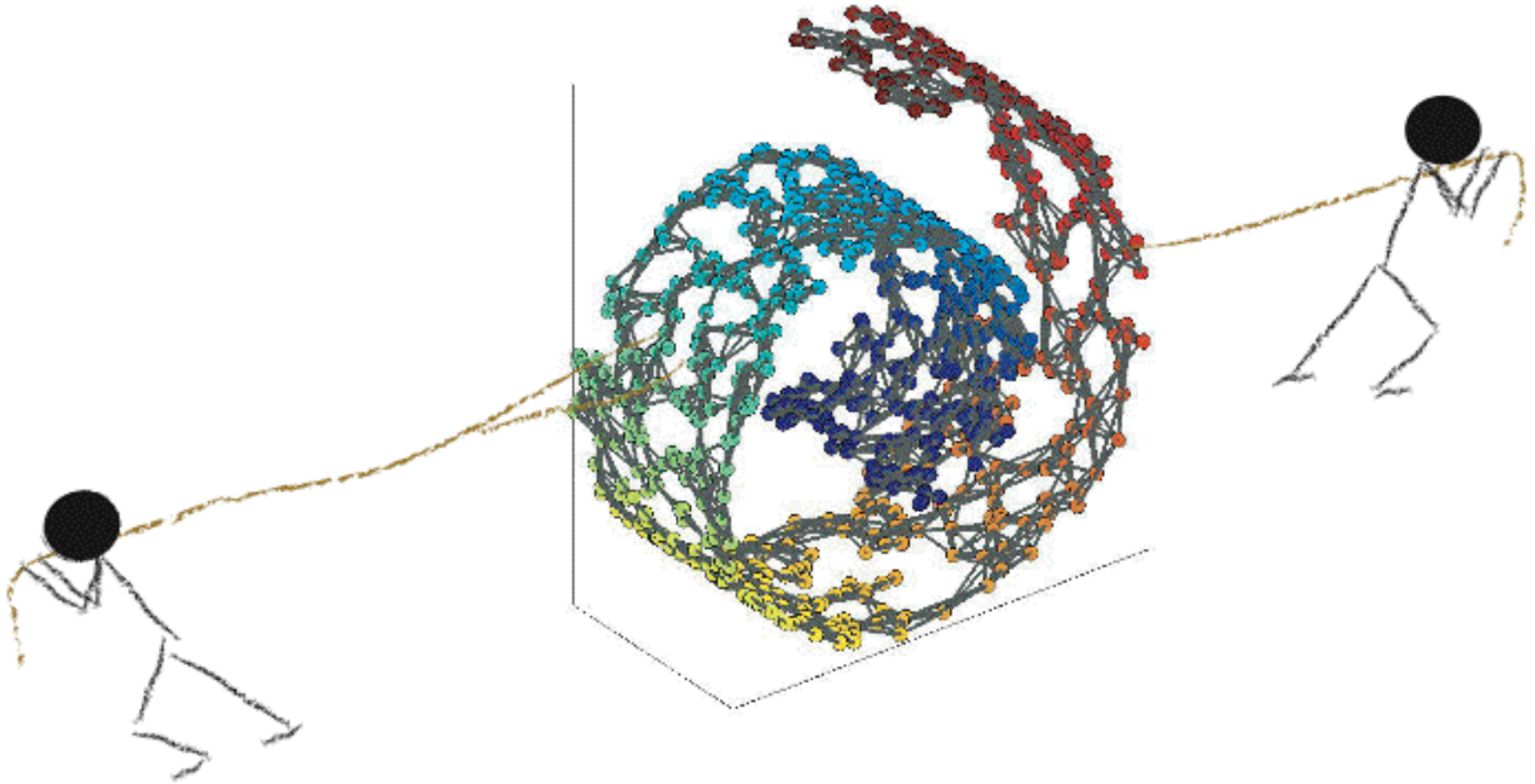
To unfurl a sheet, we pull on its four corners. What does this optimize?



inputs $\{\vec{x}_i\}$

outputs $\{\vec{y}_i\}$

“Maximum Variance Unfolding”



Generalizes PCA computation of “maximum variance subspace”.

Notation

- **Inputs (high dimensional)**

$$\vec{x}_i \in \Re^D \text{ with } i = 1, 2, ..., n$$

- **Outputs (low dimensional)**

$$\vec{y}_i \in \Re^d \text{ where } d < D$$

- **Goals**

  Nearby points remain nearby.
  Distant points remain distant.
  (Estimate d.)

Optimization

- **Quadratic programming**

$$\text{Maximize} \sum_i \|\vec{y}_i\|^2 \text{ subject to :}$$

$$\text{(i)} \ \|\vec{y}_i - \vec{y}_j\|^2 = \|\vec{x}_i - \vec{x}_j\|^2 \text{ if } (\vec{x}_i, \vec{x}_j) \text{ are kNN}$$

$$\text{(ii)} \ \sum_i \vec{y}_i = \vec{0}$$

Allowing slack in this constraint turns rods into strings.

- **Intuition:**

Nearby inputs are connected by rigid rods.
Pull inputs apart without breaking rods.

Convex optimization

- **Change of variables**

  Gram matrix $K_{ij} = \vec{y}_i \bullet \vec{y}_j$ determines outputs up to rotation: $Y = K^{1/2}$

- **Semidefinite program**

$$\text{Maximize} \sum_i K_{ii} \text{ subject to :}$$

$$\text{(i) } K_{ii} + K_{jj} - 2K_{ij} = \left\| \vec{x}_i - \vec{x}_j \right\|^2 \text{ if } (\vec{x}_i, \vec{x}_j) \text{ are kNN}$$

$$\text{(ii) } \sum_i K_{ij} = 0$$

$$\text{(iii) } K \succ 0 \text{ is PSD}$$

Summary of algorithm

1) Nearest neighbors

Compute k-nearest neighbors and local distances.

2) Semidefinite programming

Compute maximum variance unfolding that preserves local distances.

3) Diagonalize Gram matrix

Matrix square root yields outputs.
Estimate dimensionality from rank.

Surrogate optimization

- **Heuristic**

  We have substituted an "easy" problem (maximizing variance) for a "hard" problem (minimizing rank).

- **Convex vs complex**

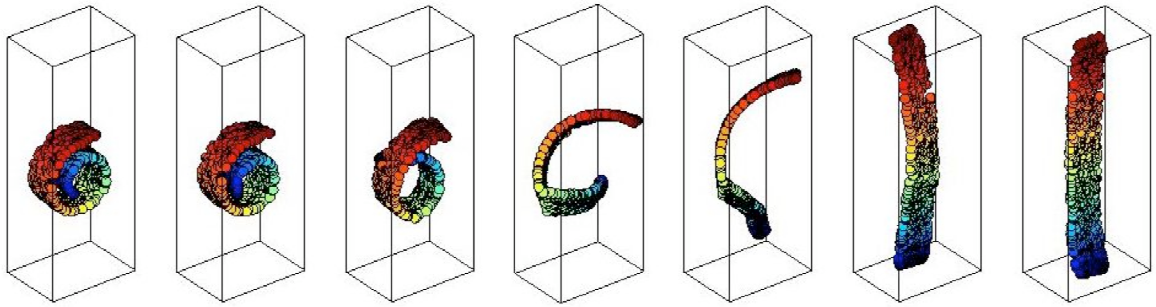
  The former is a tractable optimization.
  The latter is an NP-hard optimization.

Does it work?

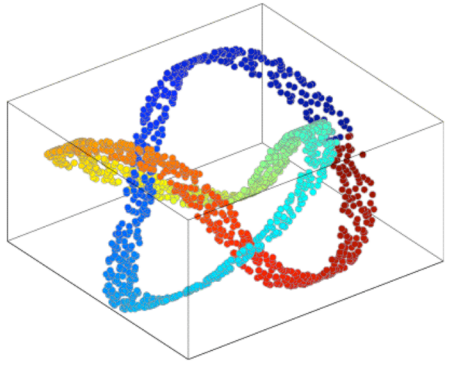
Surfaces

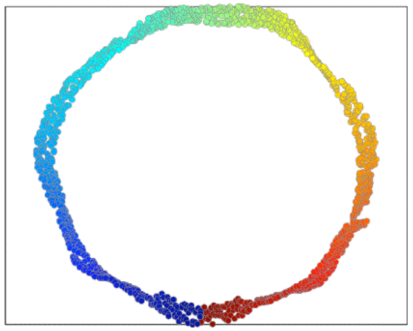
- **Swiss roll**

$n = 800$
$k = 6$



- **Trefoil knot**



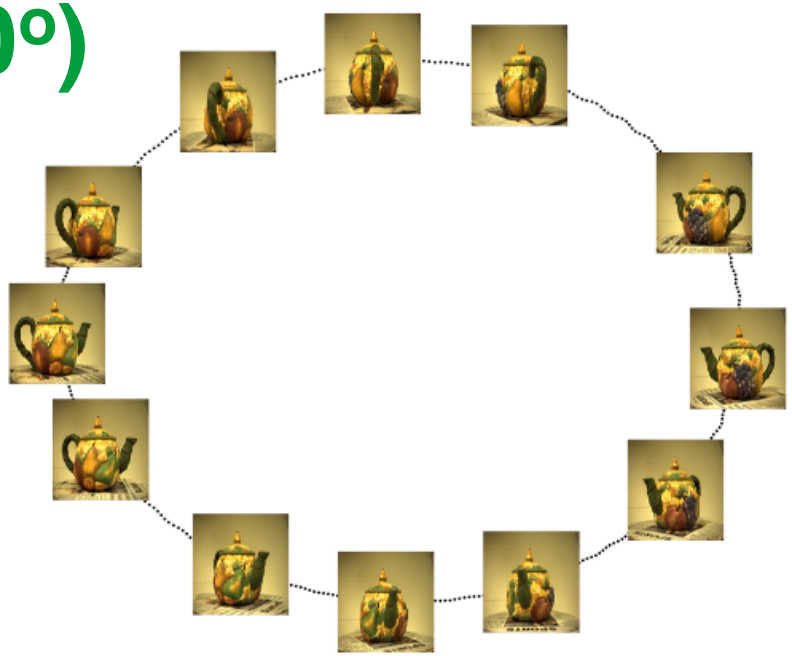


$n = 1617$
$k = 5$

Images of teapots

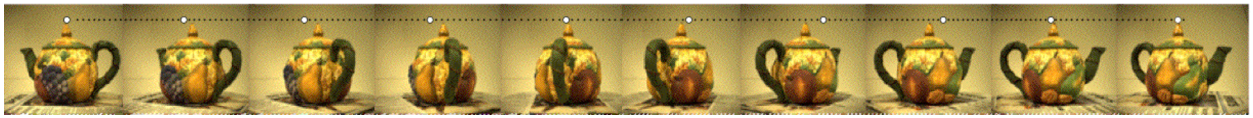
- **full rotation (360º)**

$n = 400$
$k = 4$
$D = 23028$



- **half rotation (180º)**

$n = 200$



Images are ordered by d=1 embedding.

Handwritten digits

$n = 638$

$k = 4$

$D = 256$

Images of faces

$n = 1000$

$k = 4$

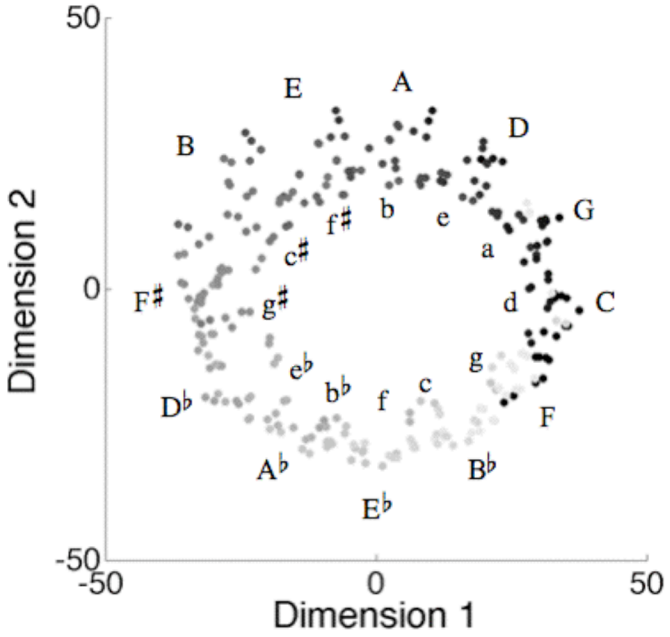
$D = 560$

Visualization

- **Tonal pitch space**

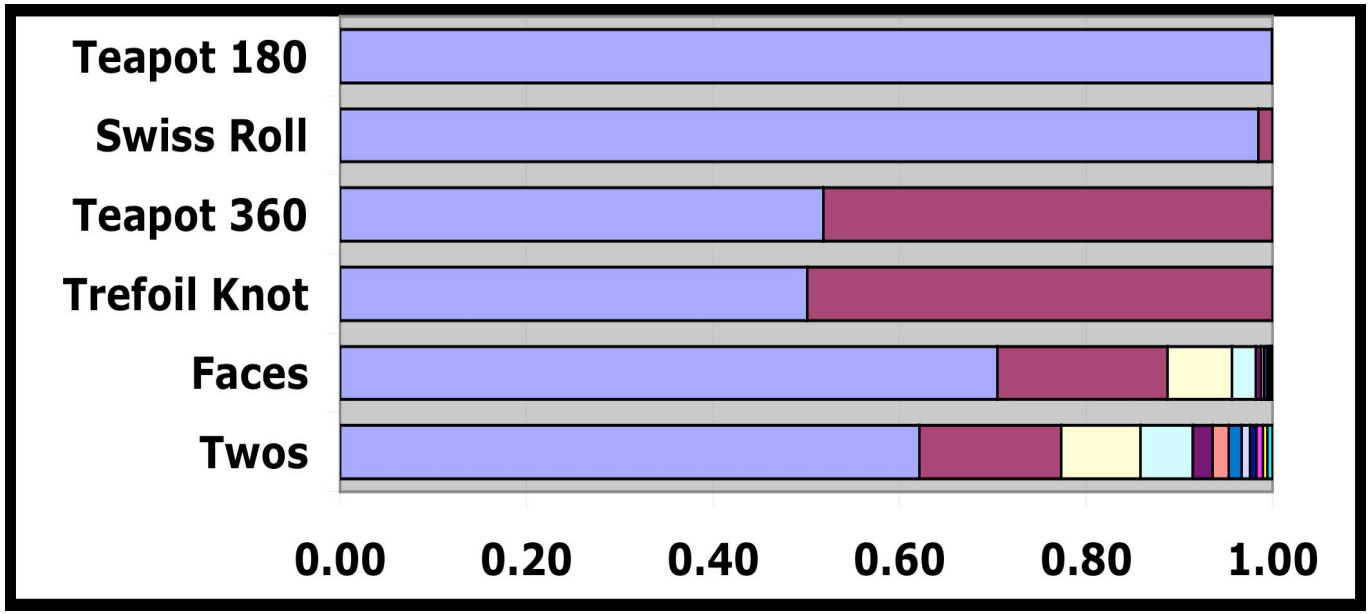
  Music theorists have defined distance functions between harmonies, such as C/C, C/g, C/C#, etc.

  (Burgoyne & Saul, 2005)



Circle of fifths (from MVU)

Eigenvalues from SDP



(normalized by trace)

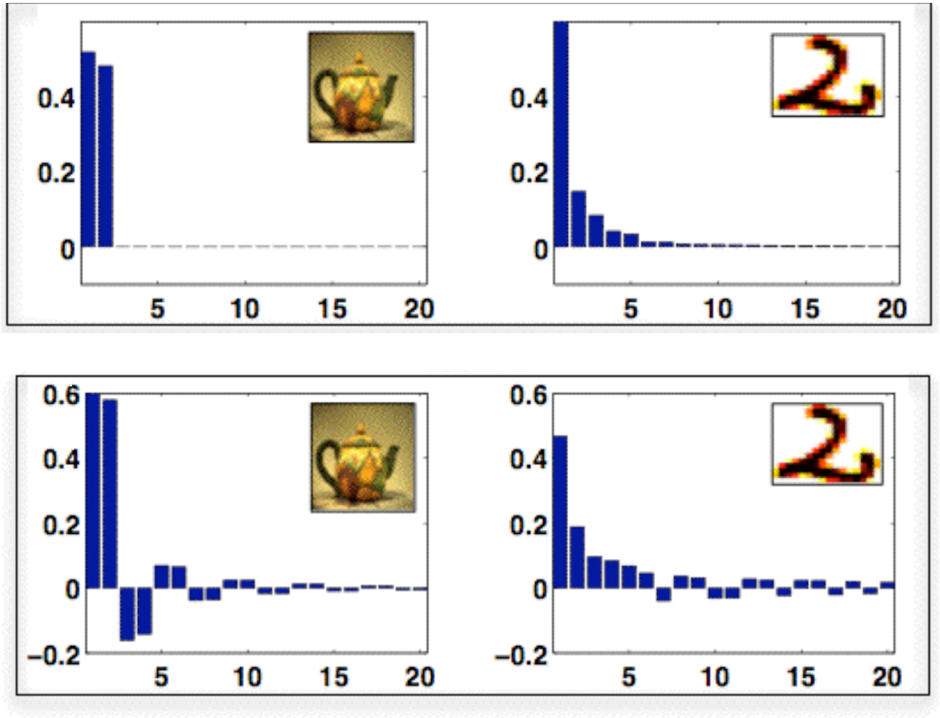
 # large eigenvalues ≈ dimensionality

MVU versus Isomap

- **Similarities**
  - Motivated by isometry
  - Based on constructing Gram matrix
  - Eigenvalues reveal dimensionality
- **Differences**
  - Semidefinite vs dynamic programming
  - Finite vs asymptotic guarantees
  - Handling of manifolds with “holes”

MVU versus Isomap

Eigenvalues of Gram matrices



Maximum variance unfolding

Isomap (foiled by "holes")

Open questions

- **Variance vs rank?**

  Why and when does maximizing variance lead to low dimensional solutions?

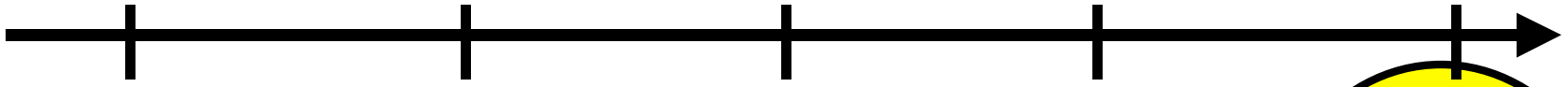
- **Asymptotic convergence?**

  Under what conditions does maximum variance unfolding converge to the "right answer"?

Properties of MVU

- **Strengths**
  - **Eigenvalues reveal dimensionality.**
  - **Constraints ensure local isometry.**
- **Weaknesses**
  - **Computation intensive**
  - **Limited to $n \leq 2000$, $k \leq 6$.**
  - **Limited to isometric embeddings.**

Algorithms



2000
Isomap
(Tenenbaum, de Silva, & Langford)

Locally Linear Embedding
(Roweis & Saul)

2002
Laplacian eigenmaps
(Belkin & Niyogi)

2003
Hessian LLE
(Donoho & Grimes)

2004
Maximum variance unfolding
(Weinberger & Saul)

(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal eigenmaps
(Sha & Saul)

Extensions

- **Conformal versus isometric maps**

  Unfold data but only preserve local angles (not distances).

- **Soft constraints**

  Allow slack in distance constraints, with linear or quadratic penalty.

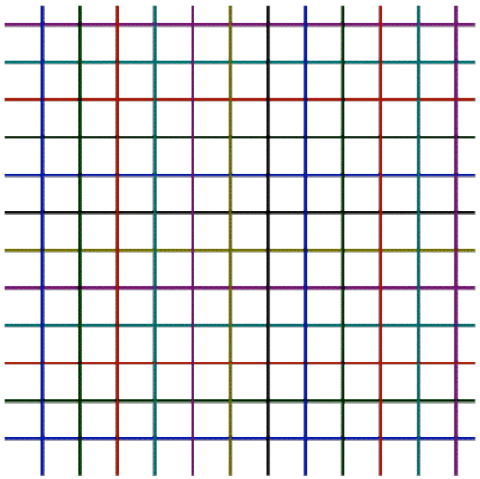
- **Graph regularization**

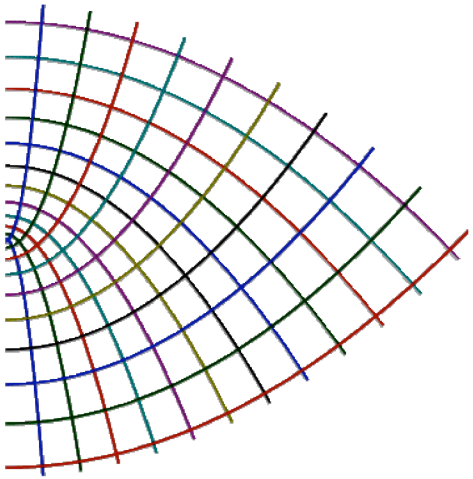
  Express solution in terms of bottom eigenvectors of graph Laplacian.

Motivation

- **Conformal map**

  Continuous and angle-preserving;
  locally preserves shapes, not distances;
  looks like rotation, translation, scaling.





Objective function

- **Measure local similarity**

  Do outputs preserve distances between kNNs up to local scaling?

$$\mathcal{D}(y,s) = \sum_{i \sim j,k} \left[\| \vec{y}_j - \vec{y}_k \|^2 - s_i \| \vec{x}_j - \vec{x}_k \|^2 \right]^2$$

vertices of kNN triangles

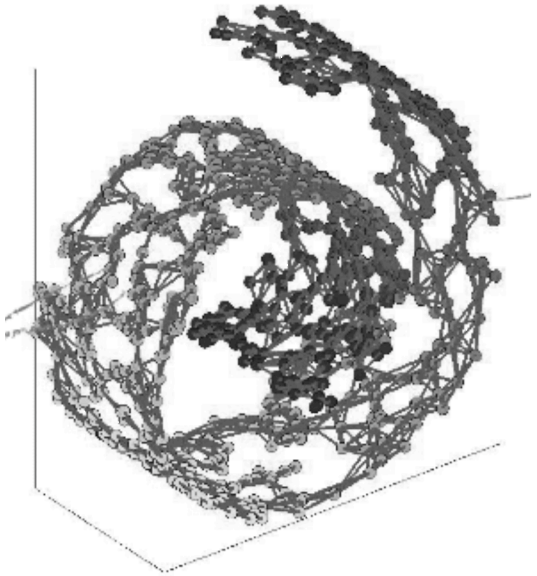
scaling factor at $\vec{x}_i$

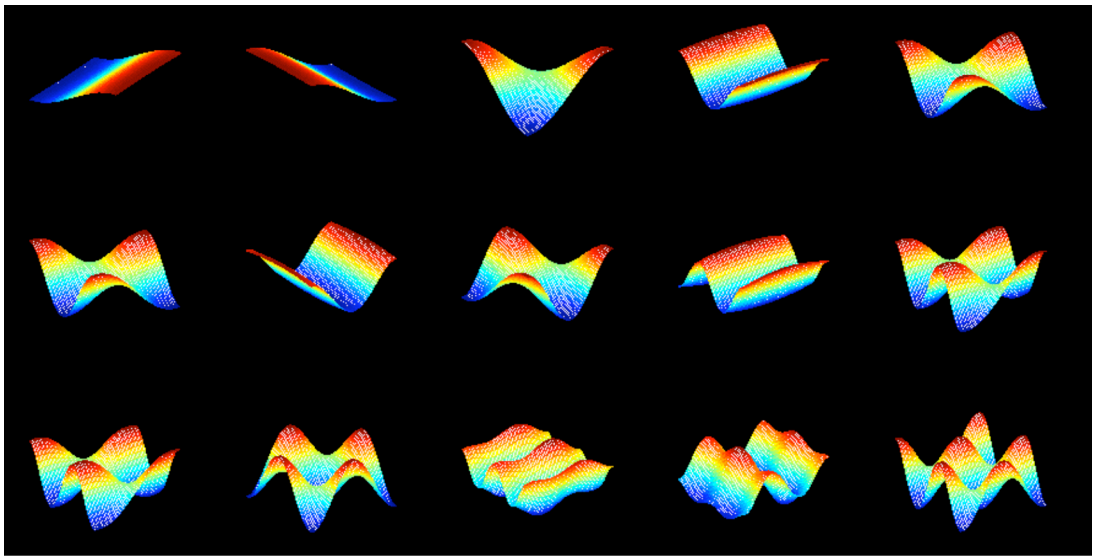
Graph Regularization

- **Spectral graph theory**

  Eigenvectors of graph Laplacian yield ordered basis for functions over graph.

- **Ex: from kNN graph on Swiss roll**





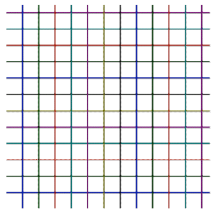
Graph regularization

- **Enforce smoothness:**

  Express outputs in terms of m bottom eigenvectors of graph Laplacian.

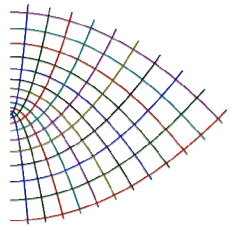
$$\vec{y}_i = \mathbf{L}\vec{e}_i \text{ with } \mathbf{L} \in \mathcal{R}^{m \times m}$$

- **Simplify optimization:**
  - **old SDP over n x n matrix $K_{ij} = y_i \bullet y_j$**
  - **new SDP over m x m matrix $P = L^T L$**
  - **huge savings** $m \ll n$



Conformal eigenmaps

(Sha & Saul, 2005)

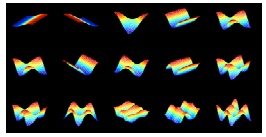


- **Cost function** $\mathcal{D}(y,s) = \sum_{ijk} \eta_{ij}\eta_{ik}\left[\left\|\vec{y}_j - \vec{y}_k\right\|^2 - s_i\left\|\vec{x}_j - \vec{x}_k\right\|^2\right]^2$

  Angles between nearby outputs should equal angles between nearby inputs.

- **Graph regularization** $\vec{y}_i = \mathbf{L}\vec{e}_i$ with $\mathbf{L} \in \mathcal{R}^{m\times m}$

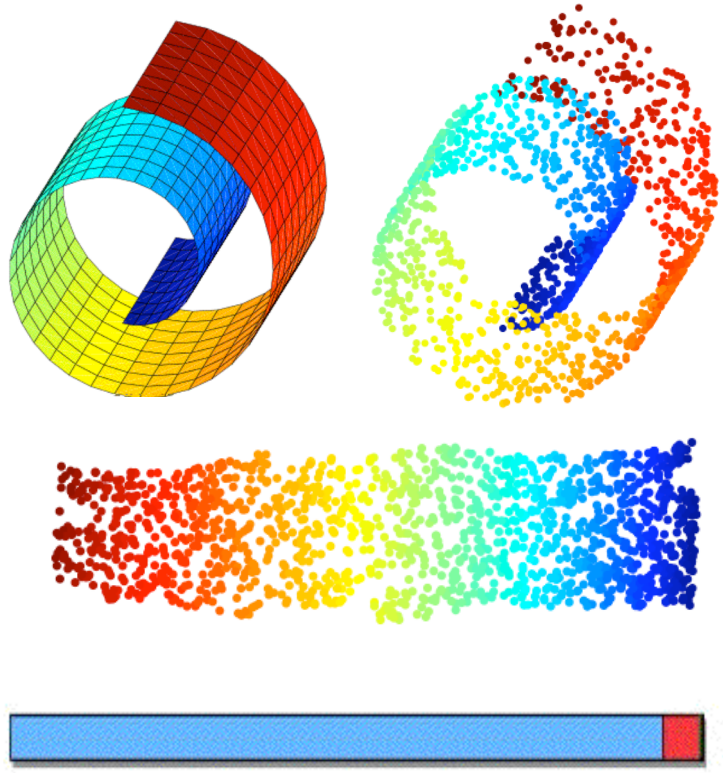
  Expand solution in terms of bottom eigenvectors of graph Laplacian.



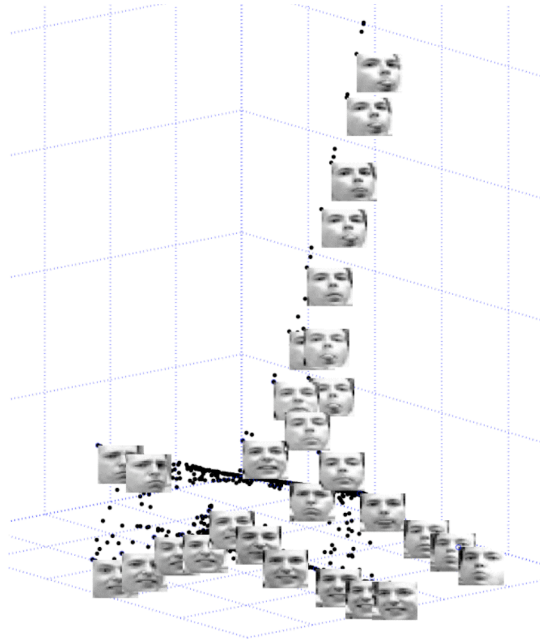
- **Optimization**

  Solve small SDP over m x m matrices.

Conformal embeddings

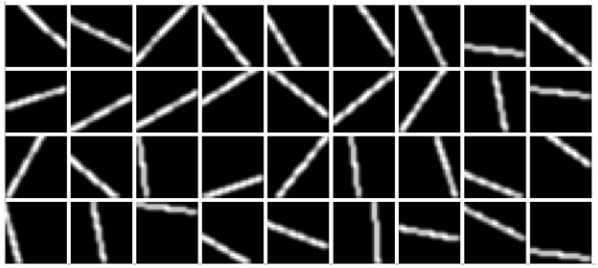


Swiss roll

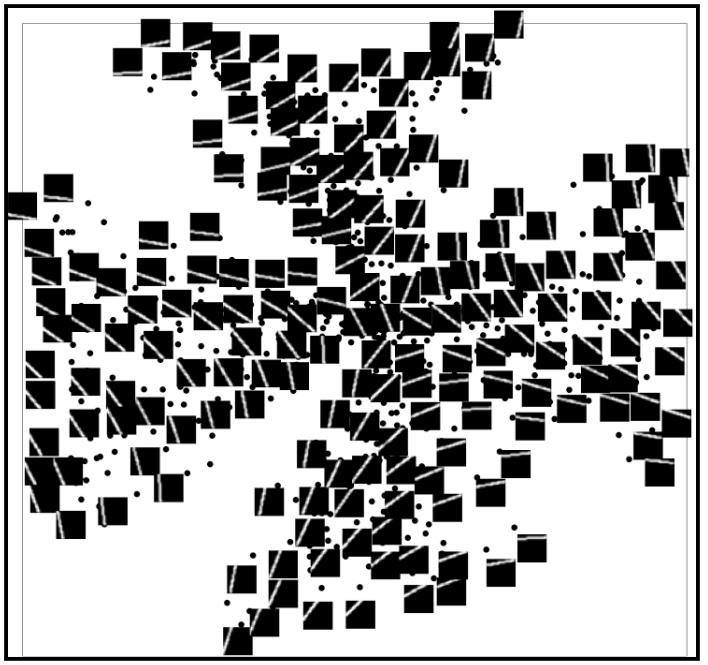


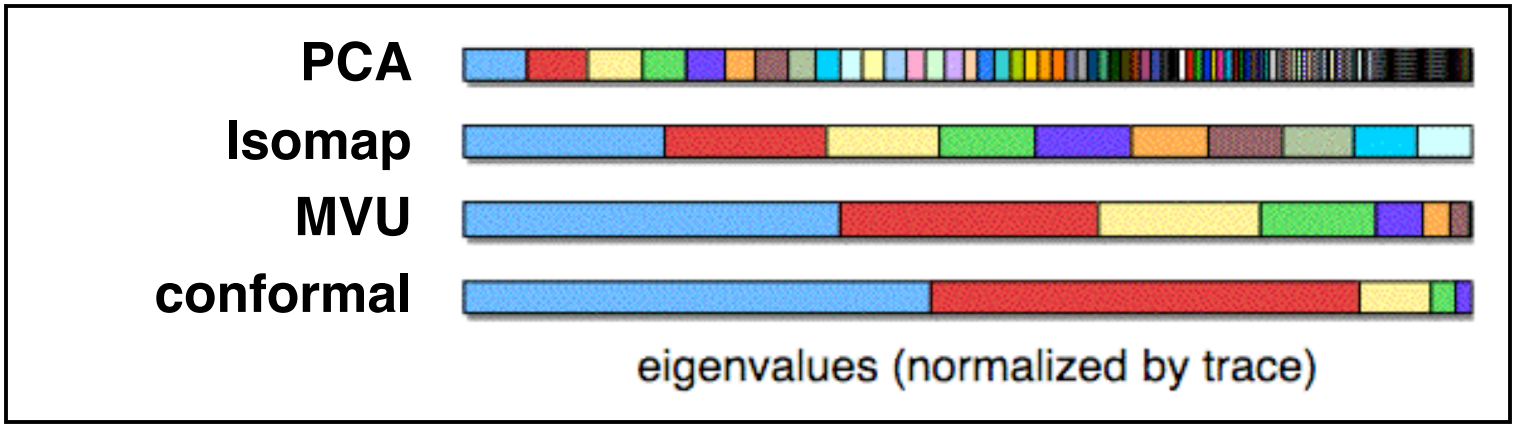
Faces

Images of Oriented Edges



$N=2016$
$D=576$





SDPs and manifold learning

- **Constrained optimizations**

  SDPs give finite-sample (vs asymptotic) guarantees for preserving distances.

- **Dimensionality estimation**

  SDP eigenvalues reveal dimensionality more robustly than Isomap.

- **Conformal transformations**

  SDPs can enforce angle-preserving maps (that originally motivated LLE).

Outline

- **Part 1 - linear versus graph-based methods**
- **Part 2 - sparse matrix methods**
- **Part 3 - semidefinite programming**
- **Part 4 - kernel methods**
- **Part 5 - parting thoughts**

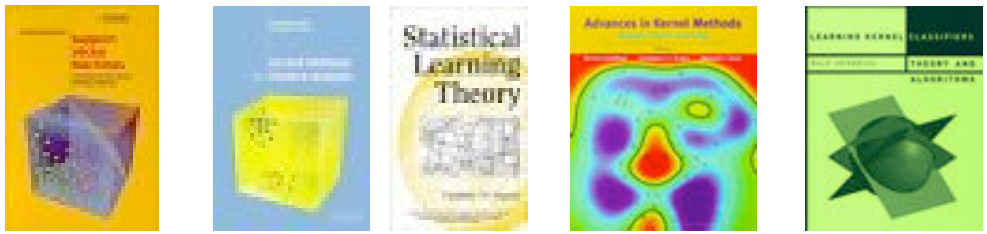
Kernel methods

- **Kernel trick**

  Substitute generalized (nonlinear) inner product for Euclidean dot product.

- **Applications**

  Kernel classifiers
  Kernel PCA
  Kernel [insert favorite linear model here]



Kernel trick

- **Kernel function**

  Measure similarity between inputs by real-valued function: $K(\vec{x}, \vec{x}')$

- **Implicit mapping**

  Appropriately chosen, the kernel function defines an inner product in "feature space":

$$K(\vec{x}, \vec{x}') = \vec{\Phi}(\vec{x}) \cdot \vec{\Phi}(\vec{x}')$$

Example

- **Gaussian kernel**

  Measure similarity between inputs by the real-valued function:

$$K(\vec{x},\vec{x}') = \exp\left(-\beta\|\vec{x}-\vec{x}'\|^2\right)$$

- **Implicit mapping**

  Inputs are mapped to surface of (infinite-dimensional) sphere:

$$K(\vec{x},\vec{x}) = \left\|\vec{\Phi}(\vec{x})\right\|^2 = 1$$

Kernel methods

- **Supervised learning**
  - Large margin classifiers
  - Kernel Fisher discriminants
  - Kernel k-nearest neighbors
  - Kernel logistic and linear regression
- **Unsupervised learning**
  - Kernel k-means
  - Kernel PCA (for manifold learning?)

Kernel PCA

- **Linear methods**

  PCA maximizes variance.

  MDS preserves inner products.

  Dual matrices yield same projections.

- **Kernel trick**

  Diagonalize kernel matrix instead of Gram matrix.

$$K_{ij} = K(\vec{x}_i, \vec{x}_j)$$
$$G_{ij} = \vec{x}_i \cdot \vec{x}_j$$

- **Interpreting kPCA**

  Map inputs to nonlinear feature space, then extract principal components.

kPCA with Gaussian kernel

- **Implicit mapping**

  Nearby inputs map to nearby features.
  Gaussian kernel map is local isometry!

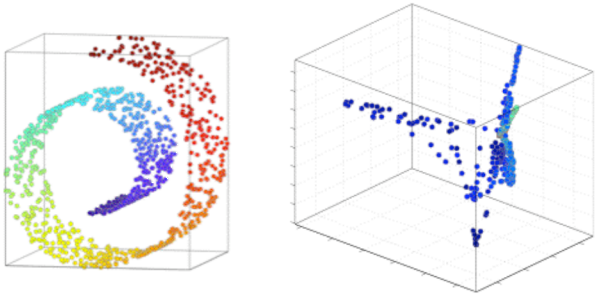
$$\begin{aligned}\left\|\vec{\Phi}_i - \vec{\Phi}_j\right\|^2 &= \left\|\vec{\Phi}_i\right\|^2 + \left\|\vec{\Phi}_j\right\|^2 - 2\vec{\Phi}_i \cdot \vec{\Phi}_j \\ &= K_{ii} + K_{jj} - 2K_{ij} \\ &\approx 2\beta\left\|\vec{x}_i - \vec{x}_j\right\|^2 \text{ for nearby inputs}\end{aligned}$$

- **Manifold learning**

  Does kernel PCA with Gaussian kernel unfold a data set? **No!**

kPCA with Gaussian kernel

- **Swiss roll**



top three kernel principal components

$$K(\vec{x},\vec{x}') = \exp\left(-\beta\|\vec{x}-\vec{x}'\|^2\right)$$

kPCA eigenvalues normalized by trace

- **Explanation**
  - **Distant patches of manifold map to orthogonal parts of feature space.**
  - **kPCA enumerates patches of size $\beta^{-1/2}$, fails terribly for manifold learning.**

kPCA and manifold learning

- **Generic kernels do not work**

  Gaussian $K(\vec{x},\vec{x}') = \exp\left(-\beta \| \vec{x} - \vec{x}' \|^2\right)$

  Polynomial $K(\vec{x},\vec{x}') = (1 + \vec{x} \cdot \vec{x}')^p$

  Hyperbolic tangent $K(\vec{x},\vec{x}') = \tanh(\vec{x} \cdot \vec{x}' + \delta)$

- **Data-driven kernel matrices**

  Spectral methods can be seen as constructing kernel matrices for kPCA.

  (Ham et al, 2004)

Spectral methods as kPCA

- **Maximum variance unfolding**

  Learns a kernel matrix by SDP.
  Guaranteed to be positive semidefinite.

- **Isomap**

  Derives kernel matrix consistent with estimated geodesics. Not always PSD.

- **Graph Laplacian**

  Pseudo-inverse yields Gram matrix for "diffusion geometry".

Diffusion geometry

- **Diffusion on graph**

  Laplacian defines continuous-time Markov chain:

$$\frac{\partial \psi}{\partial t} = -L\psi$$

- **Metric space**

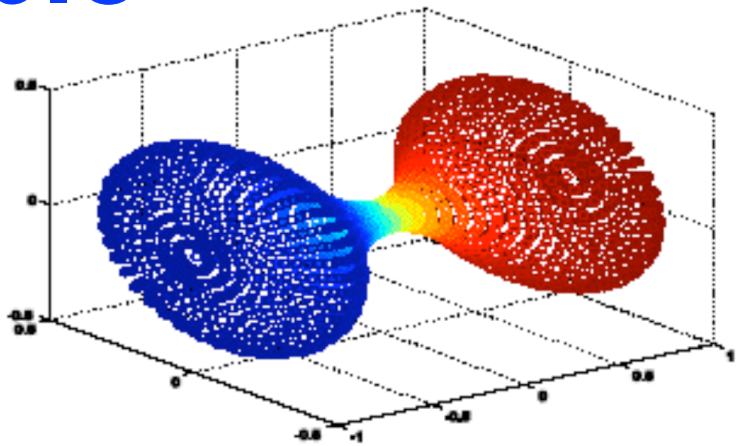
  Distances from pseudo-inverse are expected round-trip commute times:

$$\tau_{ij} = n\left(L^{\dagger}_{ii} + L^{\dagger}_{jj} - L^{\dagger}_{ij} - L^{\dagger}_{ji} \right)$$

Example

- **Barbell data set**

  Lobes are connected by bottleneck.

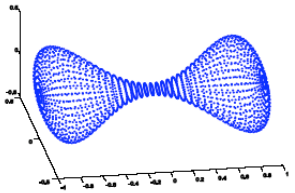


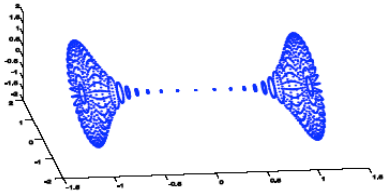
- **Comparison of induced geometries**

  + MVU will not alter barbell.

  + Laplacian will warp due to bottleneck.

  – Isomap will warp due to non-convexity.





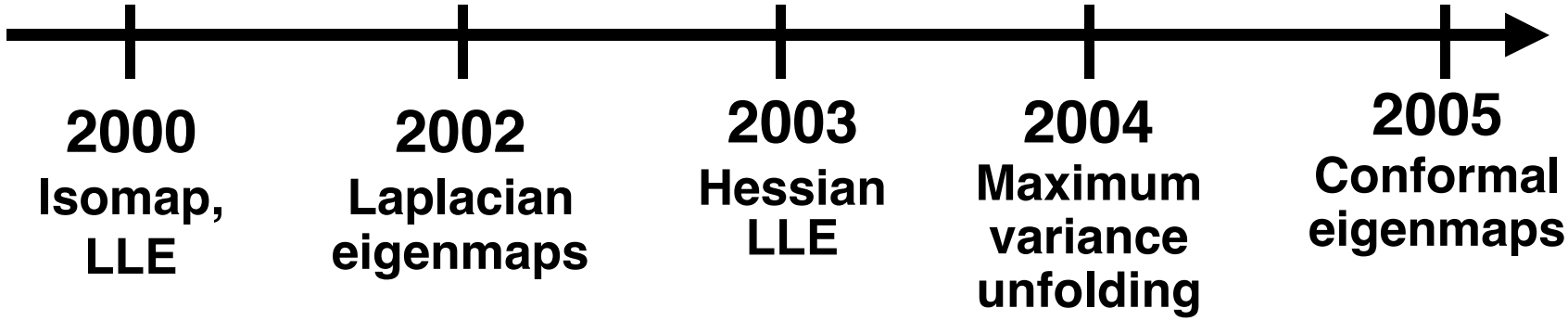
(Coifman & Lafon, 2004)

Outline

- **Part 1 - linear versus graph-based methods**
- **Part 2 - sparse matrix methods**
- **Part 3 - semidefinite programming**
- **Part 4 - kernel methods**
- **Part 5 - parting thoughts**

Quick review

- **Linear methods**
  - Principal components analysis (PCA) finds maximum variance subspace.
  - Metric multidimensional scaling (MDS) finds distance-preserving subspace.
- **Graph-based methods**



Graph-Based Methods

- **Common framework**

  1) Derive sparse graph (e.g., from kNN).

  2) Derive matrix from graph weights.

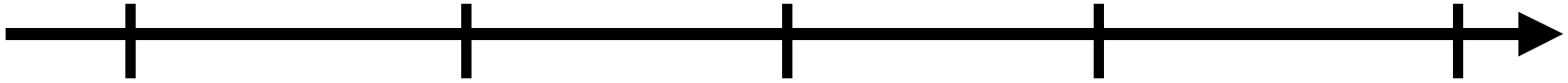
  3) Derive embedding from eigenvectors.

- **Varied solutions**

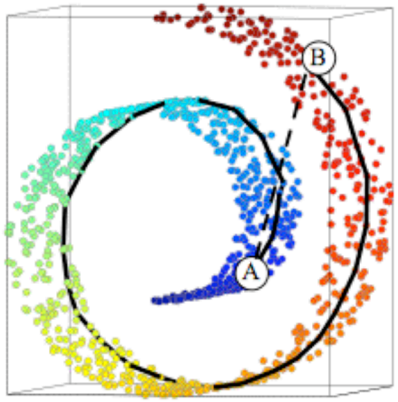
  Algorithms differ in step 2.

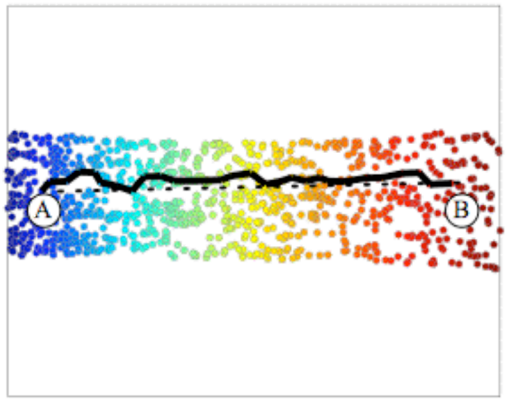
  Types of optimization: shortest paths, least squares fits, semidefinite programming.

In sixty seconds or less…



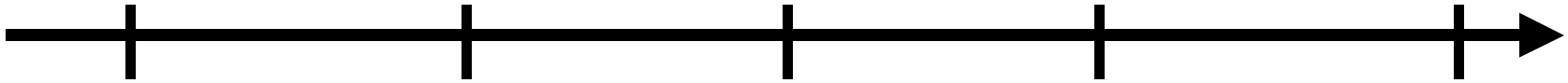
2000	2002	2003	2004	2005
Isomap, LLE	Laplacian eigenmaps	Hessian LLE	Maximum variance unfolding	Conformal eigenmaps



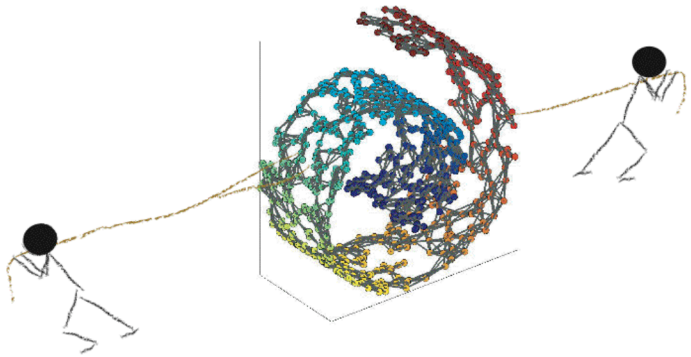


Compute shortest paths through graph.
Apply MDS to lengths of geodesic paths.

In sixty seconds or less…

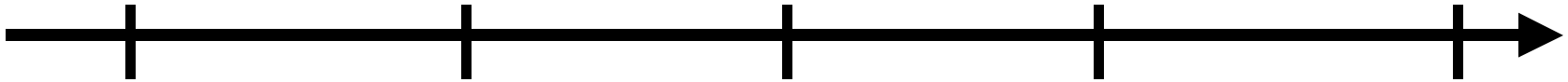


2000	2002	2003	2004	2005
Isomap, LLE	Laplacian eigenmaps	Hessian LLE	**Maximum variance unfolding**	Conformal eigenmaps

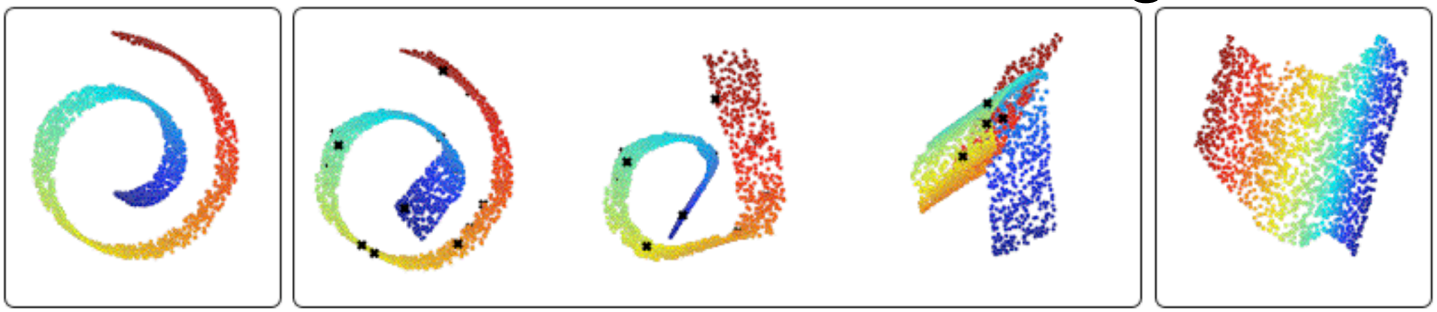


Maximize variance while respecting local distances, then apply MDS.

In sixty seconds or less…



2000	2002	2003	2004	2005
Isomap, LLE	Laplacian eigenmaps	Hessian LLE	Maximum variance unfolding	Conformal eigenmaps



Integrate local constraints from overlapping neighborhoods. Compute bottom eigenvectors of sparse matrix.

In sixty seconds or less…

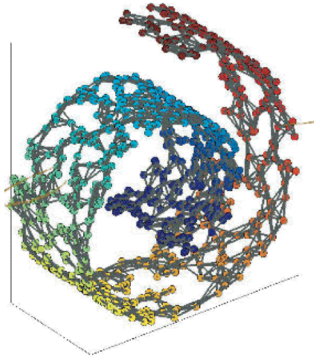
2000
Isomap, LLE

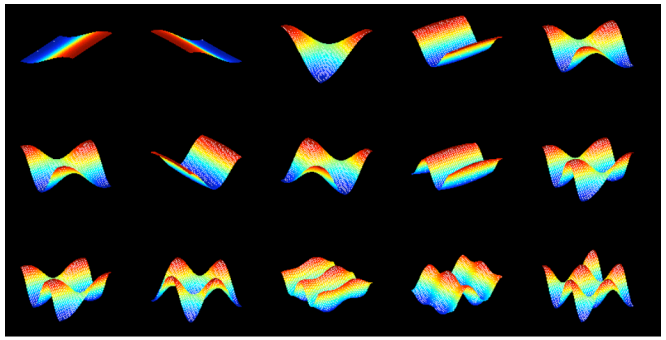
2002
Laplacian eigenmaps

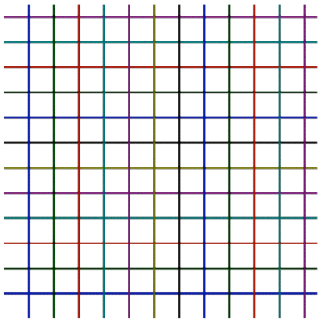
2003
Hessian LLE

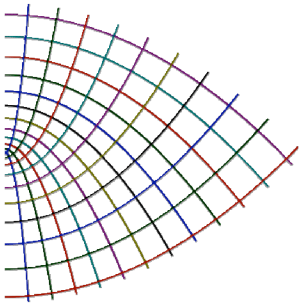
2004
Maximum variance unfolding

2005
Conformal eigenmaps









Compute best angle-preserving map using partial basis from graph Laplacian.

Other spectral methods

- **c-Isomap**

  Extends Isomap to conformal mappings (de Silva & Tenenbaum, 2003).

- **Charting**

  Parameterizes solution by radial basis functions (Brand, 2003).

- **Local tangent space alignment**

  Computes solution from analysis of over-lapping tangent spaces (Zhang & Zha, 2004).

- **Geodesic nullspace analysis**

  Recovers exact parameterizations of a certain class of manifolds (Brand, 2004).

Resources on the web

- **Software**

  http://isomap.stanford.edu

  http://www.cs.toronto.edu/~roweis/lle

  http://basis.stanford.edu/WWW/HLLE

  http://www.seas.upenn.edu/~kilianw/sde/download.htm

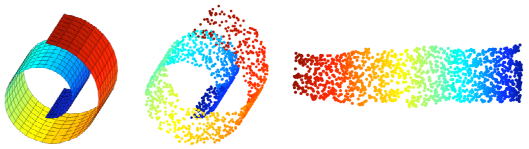
- **Links, papers, etc.**

  http://www.cs.ubc.ca/~mwill/dimreduct.htm

  http://www.cse.msu.edu/~lawhiu/manifold

  http://www.cis.upenn.edu/~lsaul

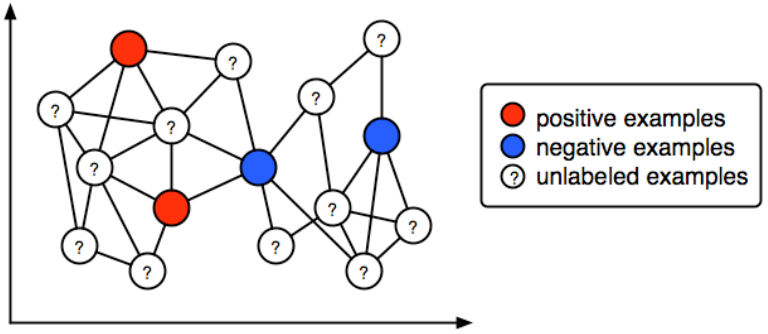
Uses of manifold learning

- **Dimensionality reduction** 

  Search for low dimensional manifolds in high dimensional data.

- **Semi-supervised learning** [Belkin & Niyogi, 2004; Zien et al, Eds., 2005]

  Use graph-based discretization of manifold to infer missing labels.



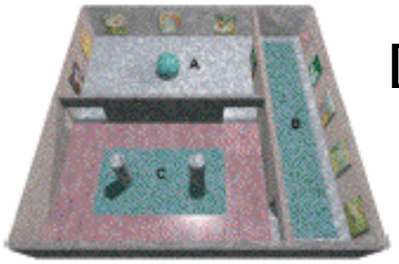
Build classifiers from bottom eigenvectors of graph Laplacian.

More uses of manifold learning

- **Reinforcement learning** [Mahadevan & Maggioini, 2005]

  Infer graph from topology of state space in a Markov decision process. Approximate value functions using graph Laplacian eigenfunctions.

- **Mapping and robot localization**
  - Action-respecting embeddings [Bowling et al, 2005]
  - Learning robot pose from panoramic images [Ham et al, 2005]



More uses of manifold learning

- **Learning correspondences**

  How to learn manifold structure that is shared across multiple data sets?

[Ham et al. 2003, 2005]

http://www.bushorchimp.com

Conclusion

- **Big ideas**
  - Manifolds are everywhere.
  - Graph-based methods can learn them.
  - Seemingly nonlinear; nicely tractable.
- **Ongoing work**
  - Theoretical guarantees & extrapolation
  - Spherical & toroidal geometries
  - Applications (vision, graphics, speech)