Spectral Methods for Dimensionality Reduction

Prof. Lawrence Saul

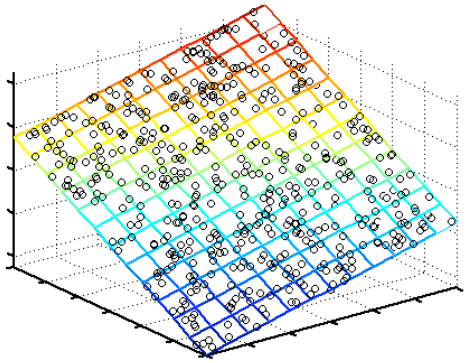
Dept of Computer & Information Science
University of Pennsylvania

UCLA IPAM Tutorial, July 11-14, 2005

Manifold learning

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

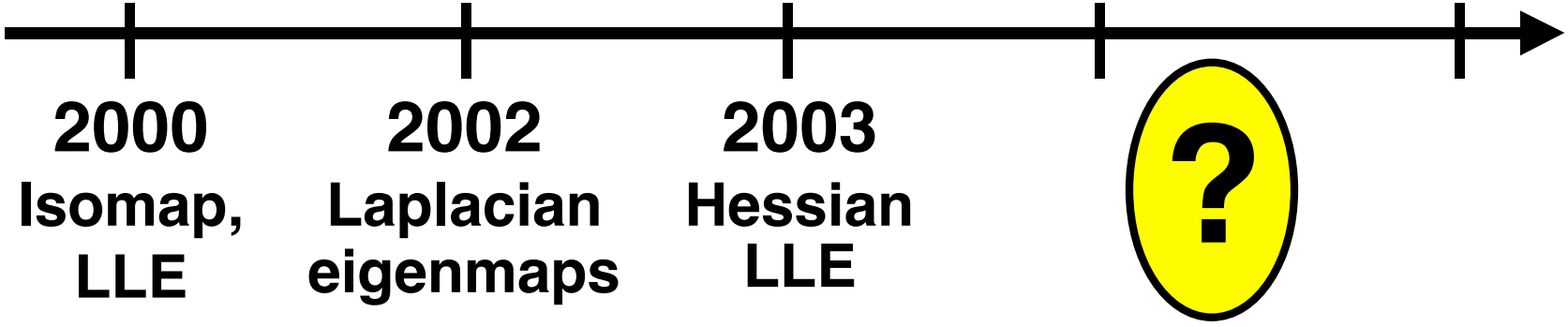
Linear vs nonlinear



What computational price must we pay for nonlinear dimensionality reduction?

Quick review

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



Nonlinear methods

1. Find k-nearest neighbors.
2. Estimate:
   - (a) geodesic distances (Isomap)
   - (b) local linear reconstructions (LLE)
   - (c) discrete Laplacian
   - (d) discrete Hessian (hLLE)
3. Compute:
   - (a) top eigenvectors of Gram matrix
   - (b-d) bottom eigenvectors of sparse matrix

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?

Unresolved issues

- **How to estimate dimensionality?**

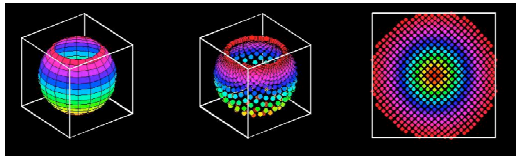
  Revealed by eigenvalues of Isomap, but specified in advance for (h)LLE.

- **How to compute eigenvectors?**

  Bottom eigenvalues of local methods are tightly spaced for large data sets.

- **Must we preserve distances?**

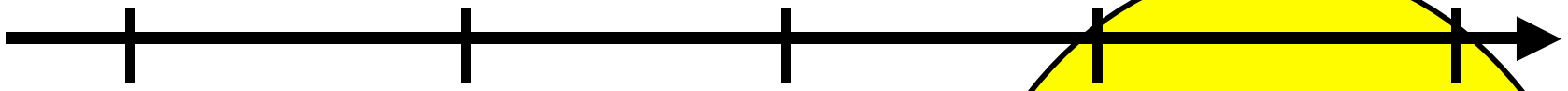
  Preserving distances may hamper dimensionality reduction.



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
  - Non-distance-preserving maps?

Today



2000
Isomap, LLE

2002
Laplacian eigenmaps

2003
Hessian LLE

2004
Maximum variance unfolding

(Weinberger & Saul)

(Sun, Boyd, Xiao, & Diaconis)

2005
Conformal eigenmaps

(Sha & Saul)

Nonlinear dimensionality reduction by 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 $\vec{a} \bullet \vec{u}$ subject to:

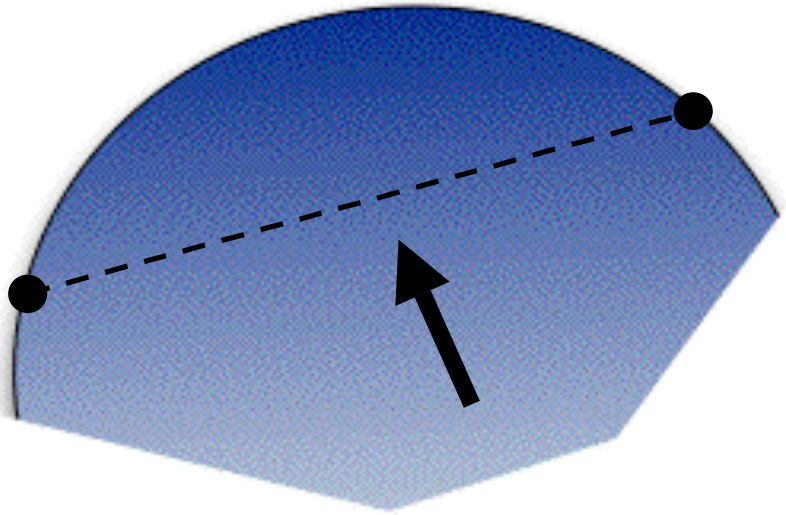
  (i) $\vec{b}_i \cdot \vec{u} > 0$ for $i = 1, 2, \ldots, c$

  (ii) $u_1 M_1 + u_2 M_2 + \ldots u_d M_d$ is PSD.

Convex optimization

- **Constraints**

  Linear and PSD constraints are convex.



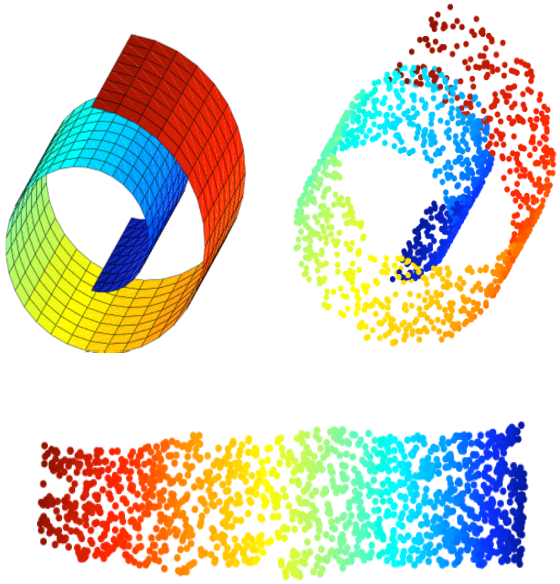
- **Cost function**

  Linear and bounded.

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

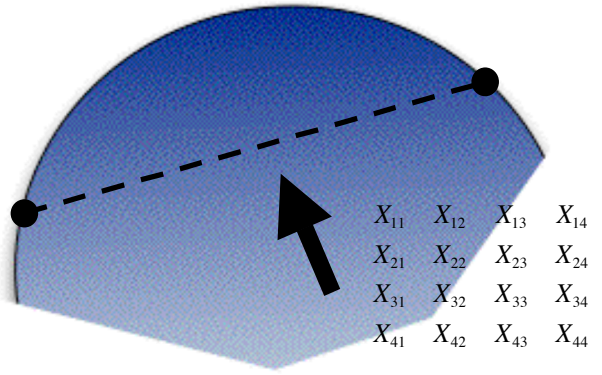
What does

nonlinear dimensionality reduction



have to do with

semidefinite programming?



Outline

- **Algorithm #1**

  Unfold data but preserve local distances.
  SDP enforces isometric constraints.

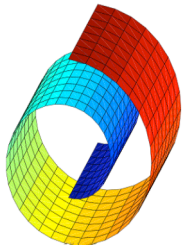
- **Algorithm #2**

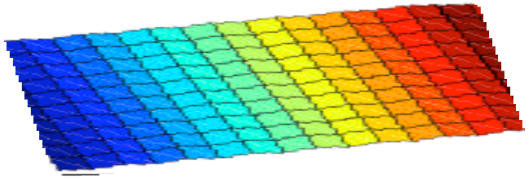
  Unfold data but preserve local angles.
  SDP learns conformal mapping.

Isometry

- **Intuitively**

  Whatever you can do to a sheet of paper without holes, tears, or self-intersections.



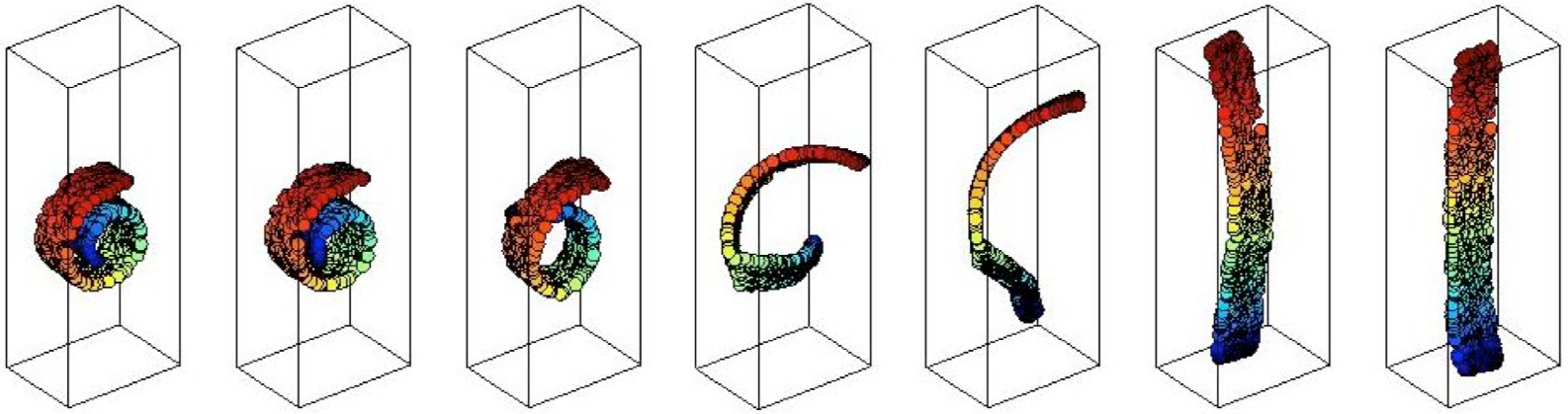


- **More formally**

  A smooth, invertible mapping that preserves distances and looks *locally* like a rotation plus translation.

Algorithm #1

What is being optimized by this sequence of isometries?



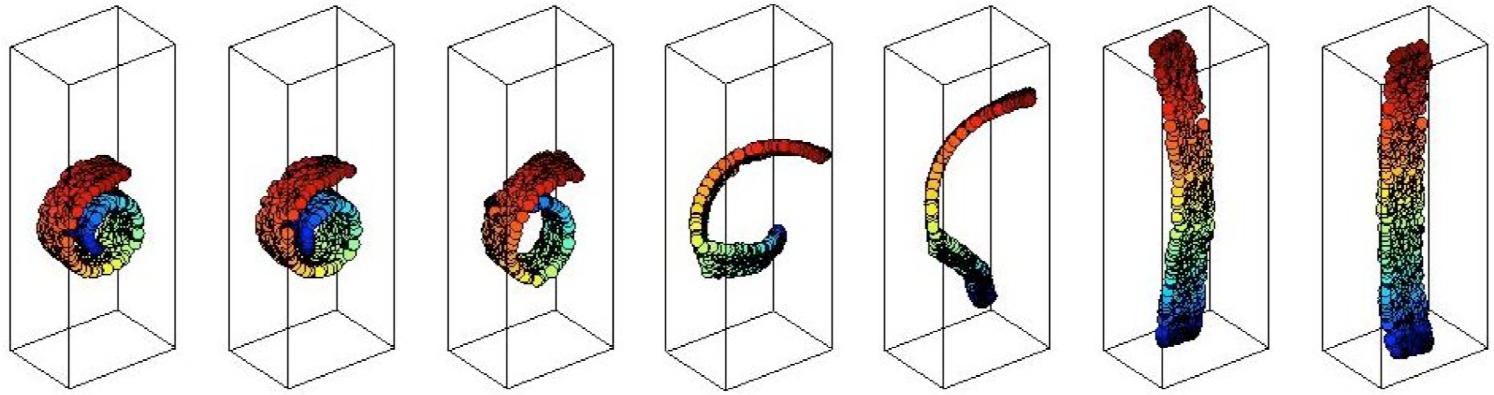
inputs $\{\vec{x}_i\}$

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

Naïve idea

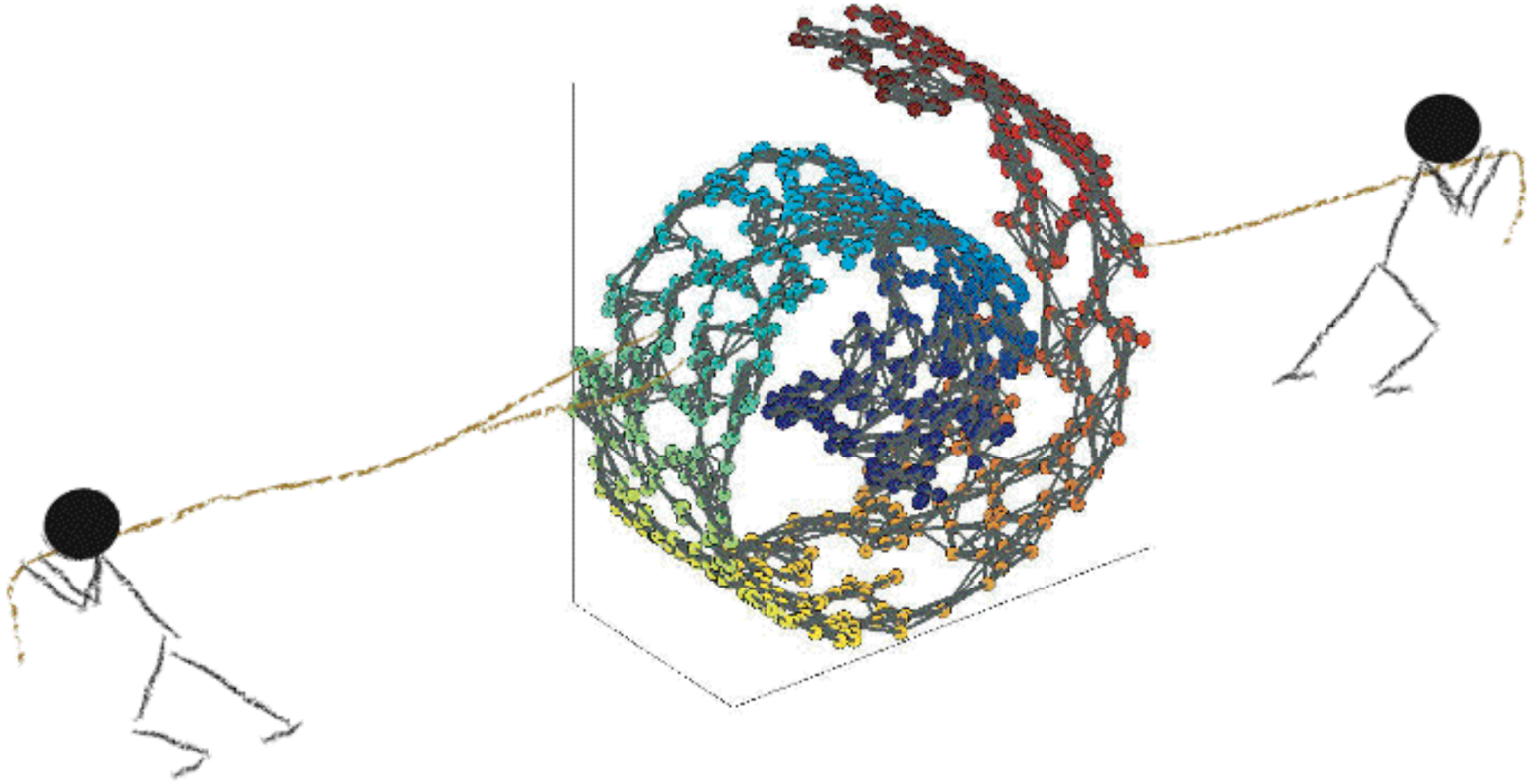
Minimize dimensionality (rank) subject to distance constraints?

NP-hard!



inputs $\{\vec{x}_i\}$ → **outputs** $\{\vec{y}_i\}$

“Maximum Variance Unfolding”



Generalizes PCA computation of “maximum variance subspace”.

Notation

- **Inputs (high dimensional)**

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

- **Outputs (low dimensional)**

  $\vec{y}_i \in \mathbb{R}^d$ where $d \ll D$

- **Goals**

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

Optimization

- **Quadratic programming**

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

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

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

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.

- **Semidefinite program**

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

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

(ii) $\sum_{ij} K_{ij} = 0$

(iii) $K \succ 0$ 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).

- **Easy vs hard**

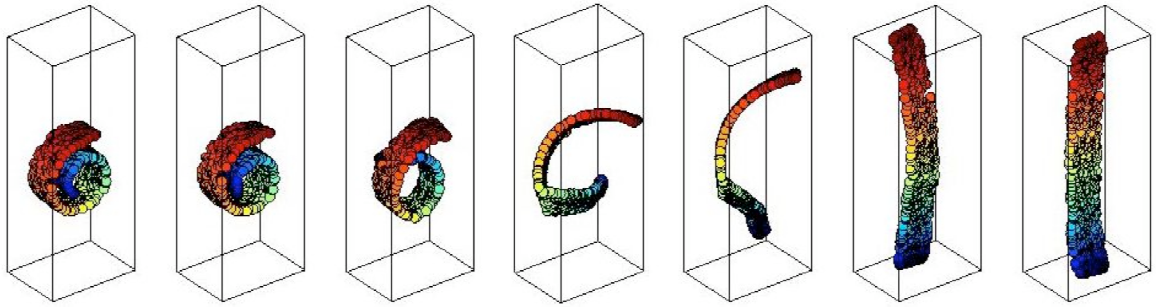
  The former is a convex optimization.
  The latter is an NP-hard optimization.

Does it work?

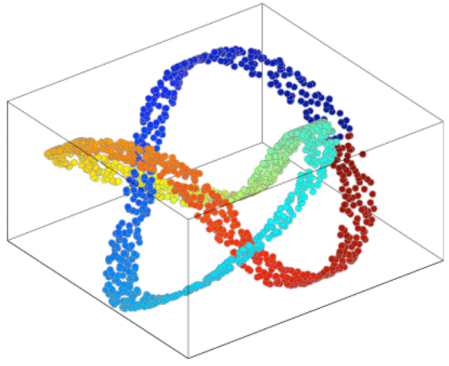
Surfaces

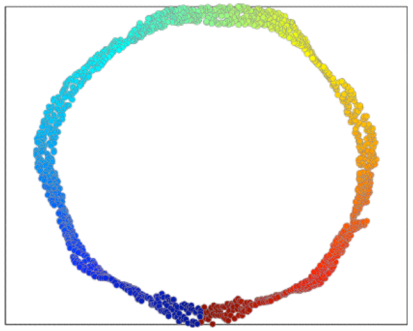
- **Swiss roll**

$N = 800$
$k = 6$



- **Trefoil knot**



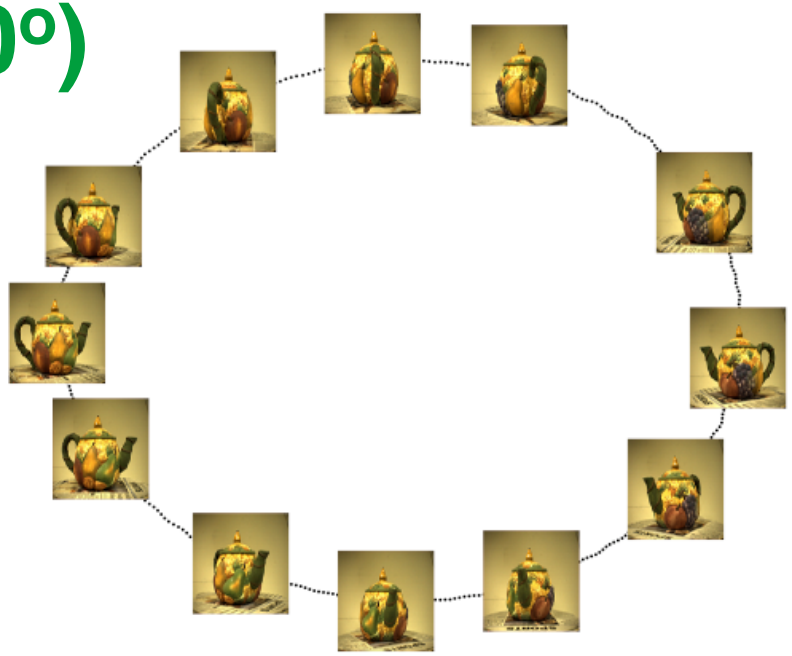


$N = 1617$
$k = 5$

Images of teapots

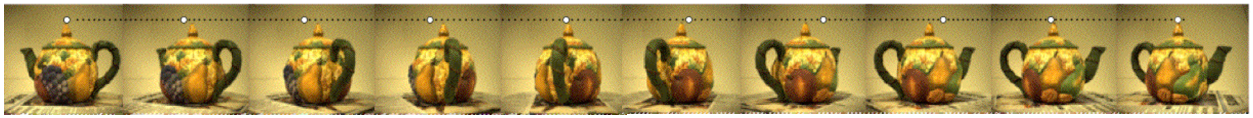
- **full rotation (360º)**

$$N = 400$$
$$k = 4$$
$$D = 23028$$



- **half rotation (180º)**

$$N = 200$$



Images are ordered by d=1 embedding.

Handwritten digits

$N = 638$

$k = 4$

$D = 256$

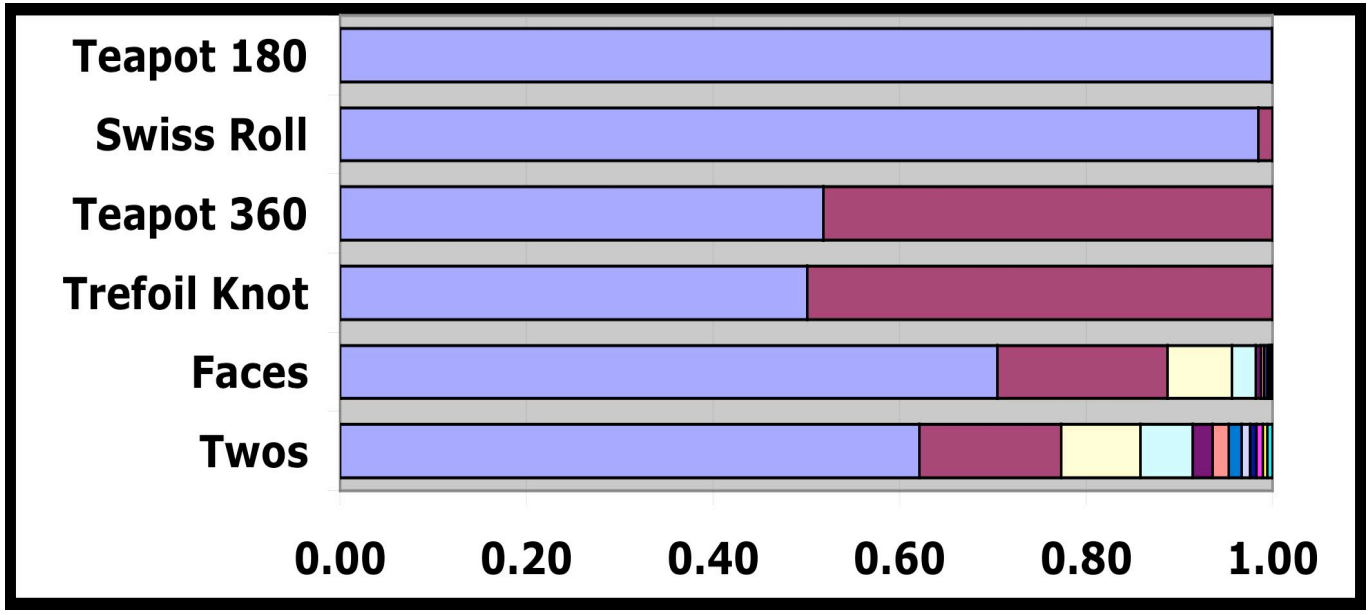
Images of faces

$N = 1000$

$k = 4$

$D = 560$

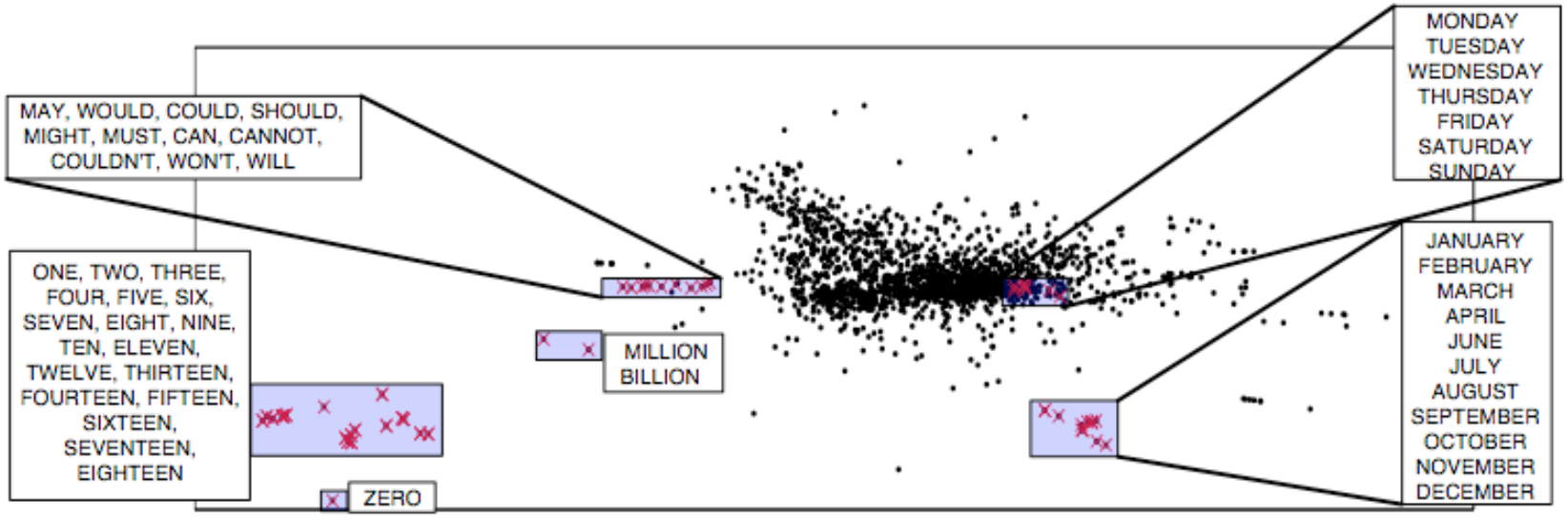
Eigenvalues from SDP

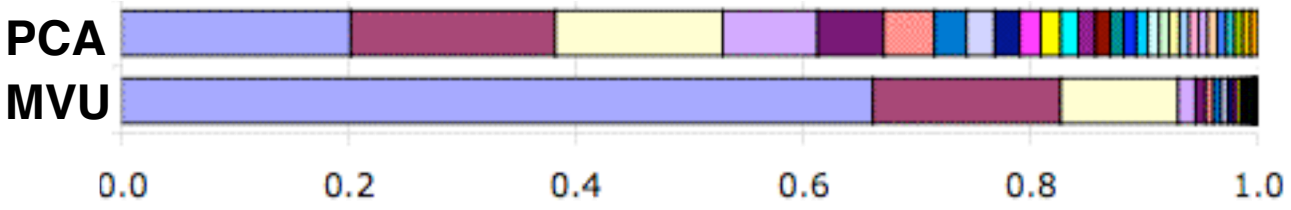


(normalized by trace)

 # eigenvalues reveals dimensionality

Word Co-occurrences





$N = 2000$

$k = 4$

$D = 60000$

Maximum variance unfolding

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

Landmark version

- **Gram matrix factorization**

  Factor $K = Q^T L Q$, where L is smaller Gram matrix between landmarks.

- **Constraint subsampling**

  Only necessary to enforce (otherwise violated) constraints in SDP solver.

- **Computational speedup**

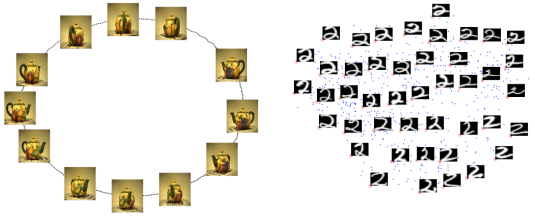
  Can solve up to n=20000, but still not as fast as landmark Isomap.

Weinberger & Saul, 2005

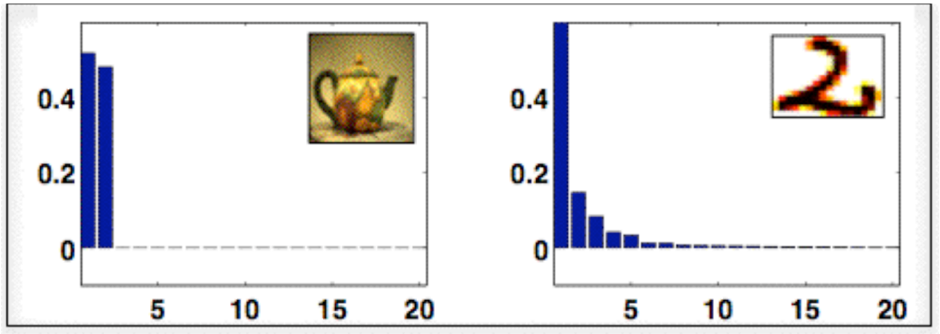
MVU versus Isomap

- **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”

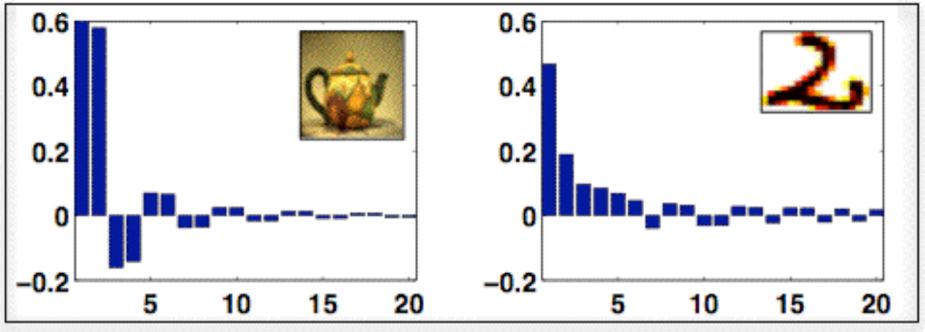
MVU versus Isomap



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"?

Outline

- **Algorithm #1**

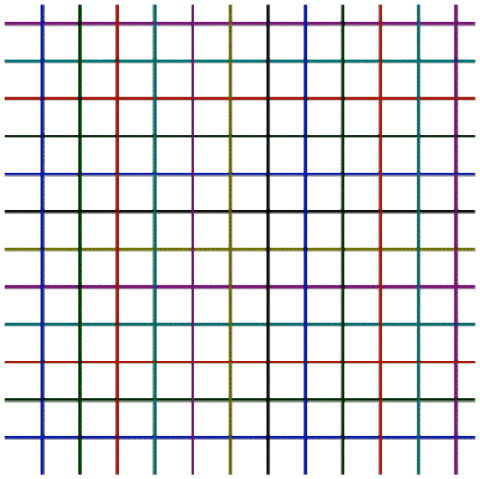
  Unfold data but preserve local distances.
  SDP enforces isometric constraints.

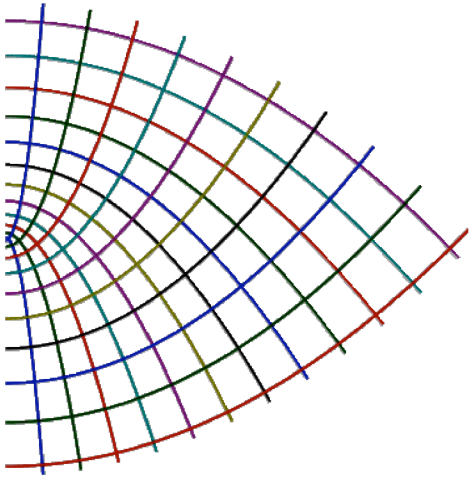
- **Algorithm #2**

  Unfold data but preserve local angles.
  SDP learns conformal mapping.

Motivation

- A conformal map between manifolds is continuous & locally angle-preserving.

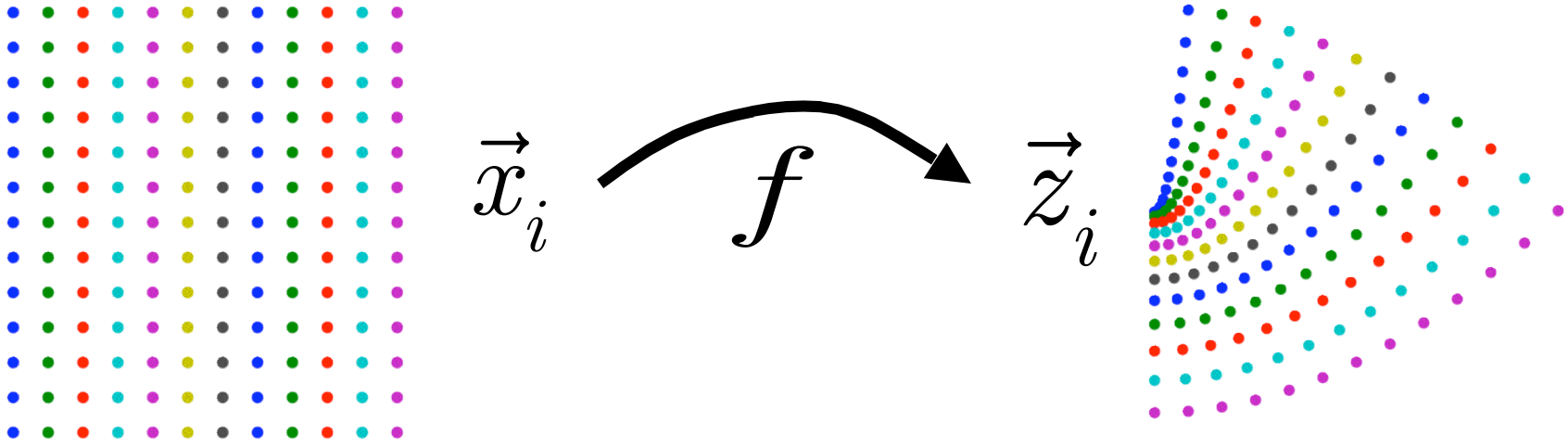




- Locally: rotation, translation, & scaling. Preserves shapes, not distances.

From manifolds to data sets

Given two data sets $\{\vec{x}_i\}_{i=1}^n$ and $\{\vec{z}_i\}_{i=1}^n$, do they appear to be related by a conformal mapping $\vec{z}_i = f(\vec{x}_i)$?

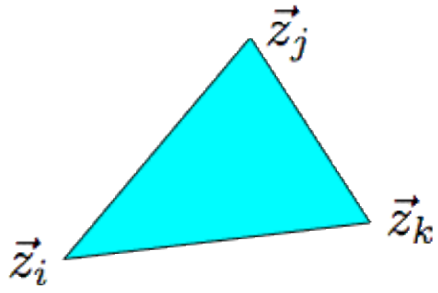


Measure local (dis)similarity.

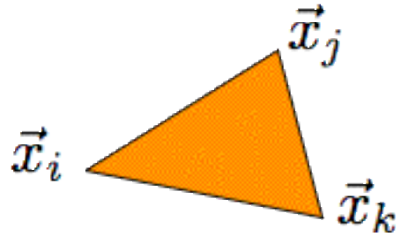
- Build k-nearest neighbor graph.
- Compare edges in triangulated graph.
- Are lengths equal (up to local scaling)?

$$\mathcal{D} = \sum_{ijk} \sum_{ij \; ik} \left\| \vec{z}_j - \vec{z}_k \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \Bigg)^2$$

vertices of triangles



scaling factor at $\vec{x}_i$



Optimal scaling factors

$$\mathcal{D}(s) = \sum_{ijk} \left(\left\| \vec{z}_{ij} - \vec{z}_{ik} \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \right)^2$$

- **Least squares fits**

  Solving for the scaling factors is easy if the inputs and outputs are fixed.

- **Analytically solvable**

  Solution can be written in closed form.
  Scaling factors are always nonnegative.

Unsupervised learning

Given inputs $\{\vec{x}_i\}$, how can we compute nontrivial outputs $\{\vec{z}_i\}$ and scaling parameters $\{s_i\}$ such that angles are maximally preserved?

Problem is ill-posed: $\min_{\{z,s\}} \mathcal{D}(z,s)$

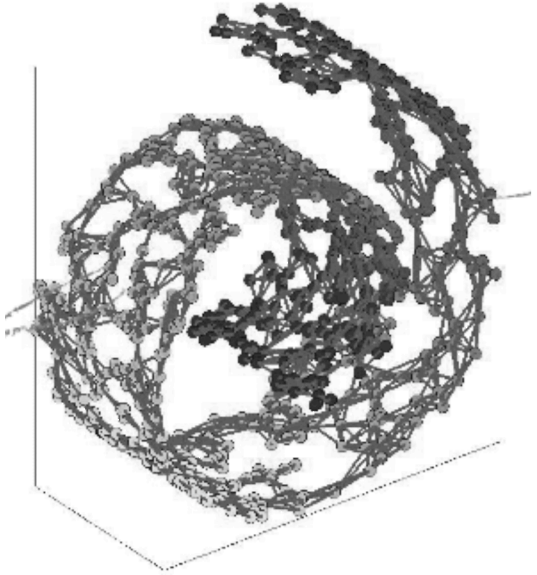
$$\mathcal{D}(z,s) = \sum_{ijk} \left(\sum_{ij} \sum_{ik} \left\| \vec{z}_j - \vec{z}_k \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \right)^2$$

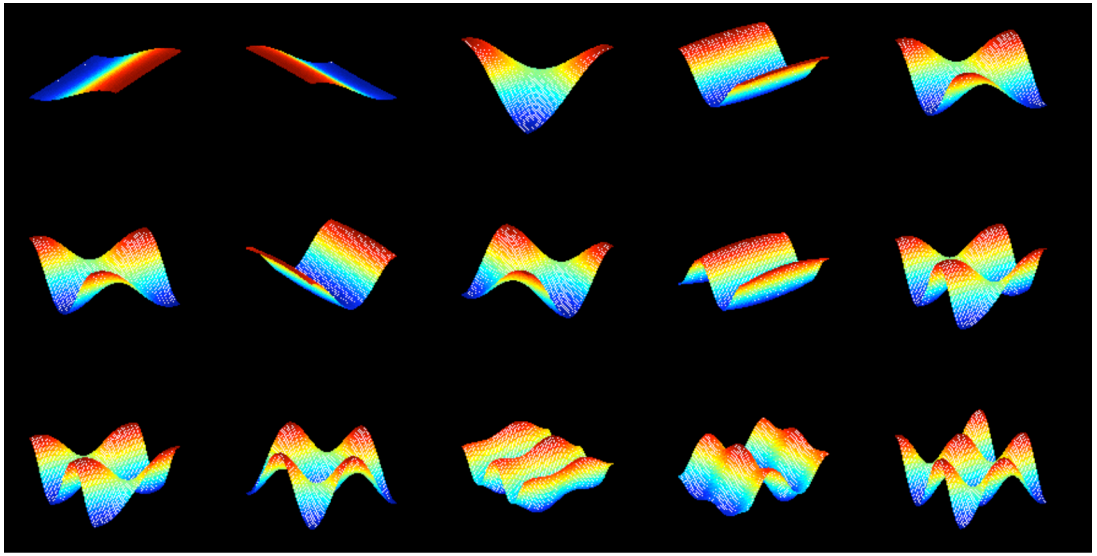
Smooth functions on graphs

- **Spectral graph theory**

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

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





Linear parameterization

- **Partial basis expansion**

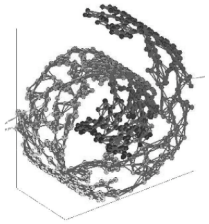
  Express outputs $\{\vec{z}_i\}$ as linear combination of $\{\vec{y}_i\}$, the m smoothest eigenvectors of graph Laplacian.

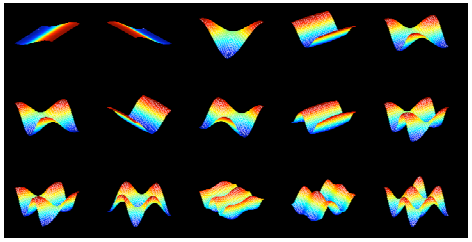
$$\vec{z}_i = \mathbf{L}\vec{y}_i \text{ with } \mathbf{L} \quad \mathcal{R}^m \;\; m$$

- **Dimensionality reduction**

  To reduce dimensionality, map inputs $\vec{x}_i \quad \mathcal{R}^D$ to $\vec{z}_i \quad \mathcal{R}^m$ with $m \ll D$.

Basis expansion





- **Regularizes solution**

  Conformal map should be smooth, with small contributions from higher eigenvectors of graph Laplacian.

- **Handles imprecision**

  Linear transformation can unmix eigenvectors that were not resolved by eigensolver.

$$\vec{z}_i = \mathbf{L}\vec{y}_i \text{ with } \mathbf{L} \quad \mathcal{R}^{m \;\; m}$$

Maximally angle-preserving map

- **Optimization**

  Minimize dissimilarity of triangles in kNN-graphs of inputs and outputs.

$$\mathcal{D}(z,s) = \sum_{ijk} \left\| \vec{z}_j - \vec{z}_k \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \quad ^2$$

- **Parameterization**

  Expand (non-zero) solution in terms of partial basis from graph Laplacian.

$$\vec{z}_i = \mathbf{L}\vec{y}_i \text{ with } \mathbf{L} \quad \mathcal{R}^{m \quad m}$$

Is this optimization tractable?

- **Cost function**

  Dissimilarity cost is quartic in linear transformation and scaling factors.

$$\mathcal{D}(L,s) = \sum_{ijk} \left(\left\| L(\vec{y}_j - \vec{y}_k) \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \right)^2$$

- **Constraint**

  Degenerate (zero) solution is prevented by quadratic constraint.

$$\mathrm{Tr}(\mathbf{L}^{\mathrm{T}}\mathbf{L}) = 1$$

Eliminating scaling factors

$$\mathcal{D}(L,s) = \sum_{ijk} \left(\left\| L(\vec{y}_j - \vec{y}_k) \right\|^2 - s_i \left\| \vec{x}_j - \vec{x}_k \right\|^2 \right)^2$$

- **Least squares fits**

  Scaling factors can be eliminated in terms of $\vec{x}_i$, $\vec{y}_i$, and L.

- **Linearity**

  Optimal scaling factors are linear in the positive semidefinite (psd) matrix $L^T L$.

Cast optimization as SDP

- **Eliminate scaling factors**

  Cost function can be expressed solely in terms of psd matrix $L^T L$.

- **Recognize quadratic form**

  Cost function is quadratic form in vector $\vec{\mathbf{v}}$ = vec$(L^T L)$.

- **Schur complement "trick"**

$$\text{If } M = \begin{matrix} A & B \\ C & D \end{matrix} \succ 0, \text{ then } A - BD^{-1}C \succ 0.$$

Semidefinite programming

- **Optimization is SDP in** $\mathbf{P} = \mathbf{L}^T\mathbf{L}$

Minimize t such that:

(i) $\mathrm{P} \succ 0$,

(ii) $\mathrm{trace}(\mathrm{P}) = 1$,

(iii) $\begin{pmatrix} \mathrm{I} & \mathrm{Sv} \\ (\mathrm{Sv})^{\mathrm{T}} & t \end{pmatrix} \succ 0.$

- **Size of SDP:**
  - **independent of # inputs (n)**
  - **independent of dimensionality (D)**
  - ***m*² unknowns, where *m*«*n* and *m*«*D***

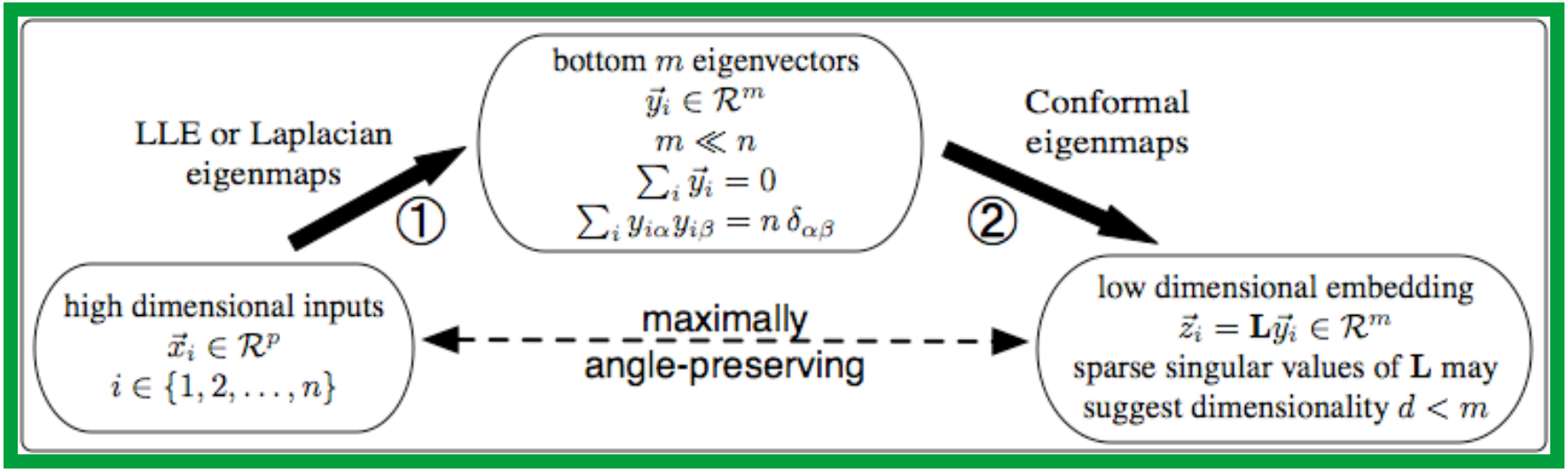
From solution of SDP in $\mathbf{P}=\mathbf{L}^{\mathrm{T}}\mathbf{L}$

- To recover angle-preserving map:

$$\mathbf{L} = \mathbf{P}^{1/2}$$
$$\vec{z}_i = \mathbf{L}\vec{y}_i$$

- To estimate dimension of manifold:
  - examine singular values of $\mathbf{L}$.
  - gap suggests dimensionality $d < m$.
  - intuitively, dimensionality ≈ rank($\mathbf{L}$).

Summary of algorithm



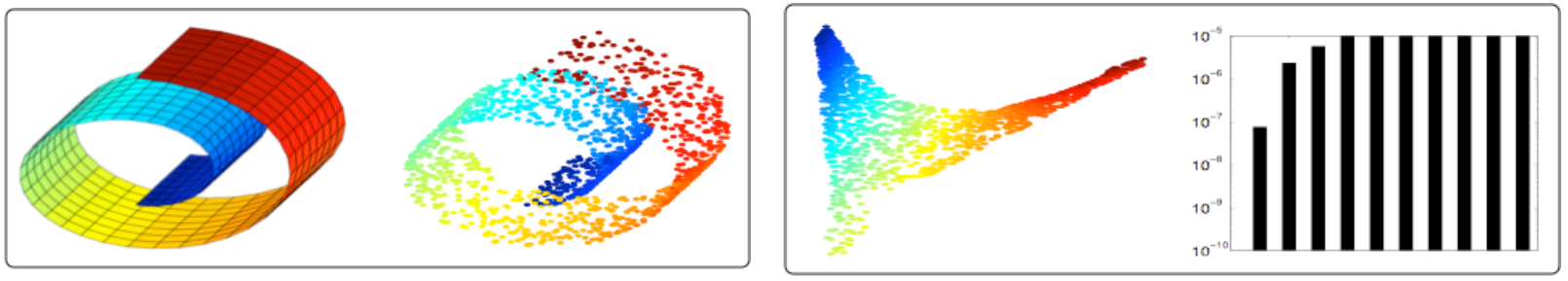
1) **Run LLE or Laplacian eigenmaps**
2) **Solve small SDP for maximally angle-preserving embedding**

LLE & graph Laplacian

Pros and cons

+ Sparse matrices scale well
+ Embeddings preserve proximity
– Topologically not geometrically faithful
– How to estimate dimensionality?
– Hard to resolve eigenvectors

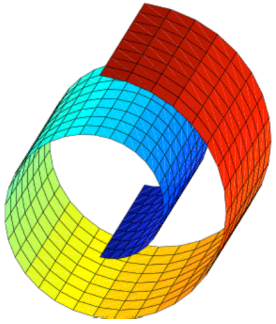
Example: Swiss roll



Comparison

- **LLE and graph Laplacian**
  - Solve sparse eigenvalue problem.
  - Output bottom $d \ll n$ eigenvectors.
- **Conformal eigenmaps (CE)**
  - Use $m \ll n$ eigenvectors as partial basis.
  - Expand solution in partial basis.
  - Solve SDP for angle-preserving map.
  - Eigenvalues yield dimensionality $d < m$.
  - Extra computation is modest (~3 min).

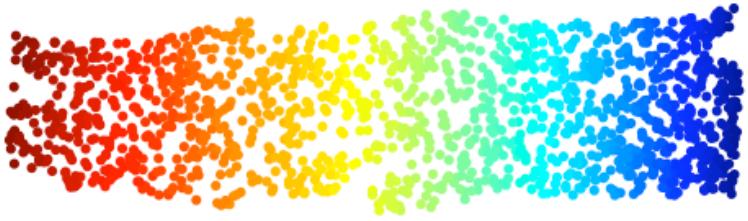
Swiss roll

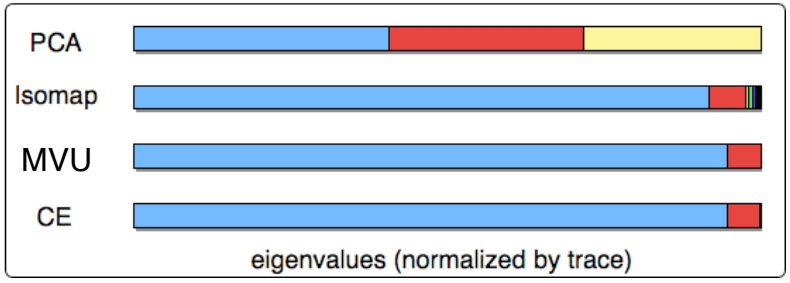


$n = 1000$ inputs
$k = 6$ nearest neighbors
$m = 10$ basis vectors (LLE)

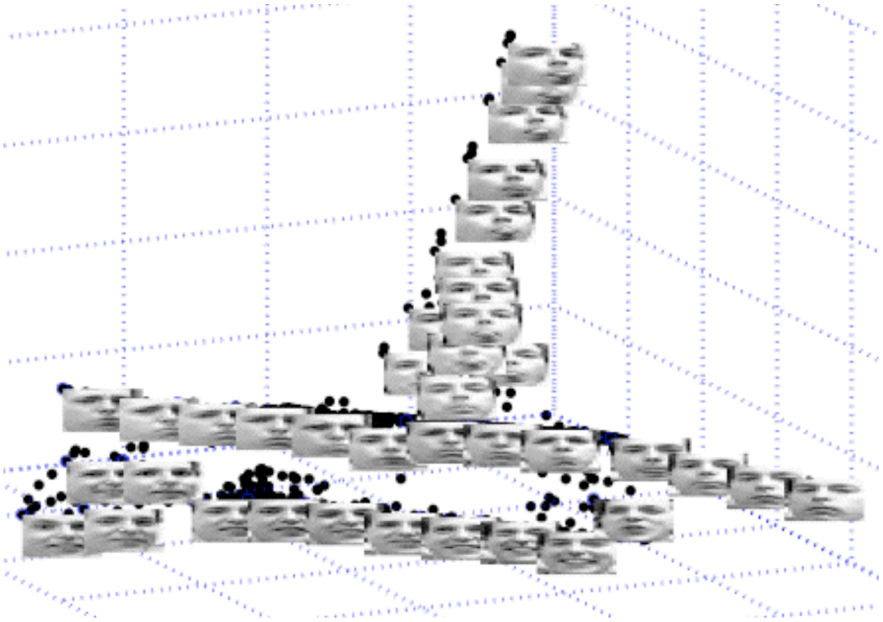
LLE

conformal

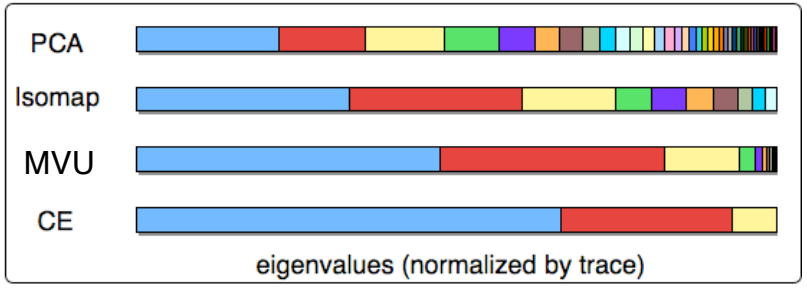




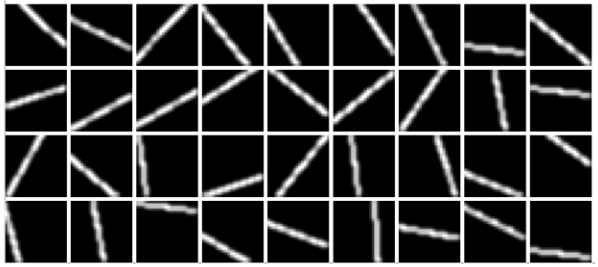
Images of Faces

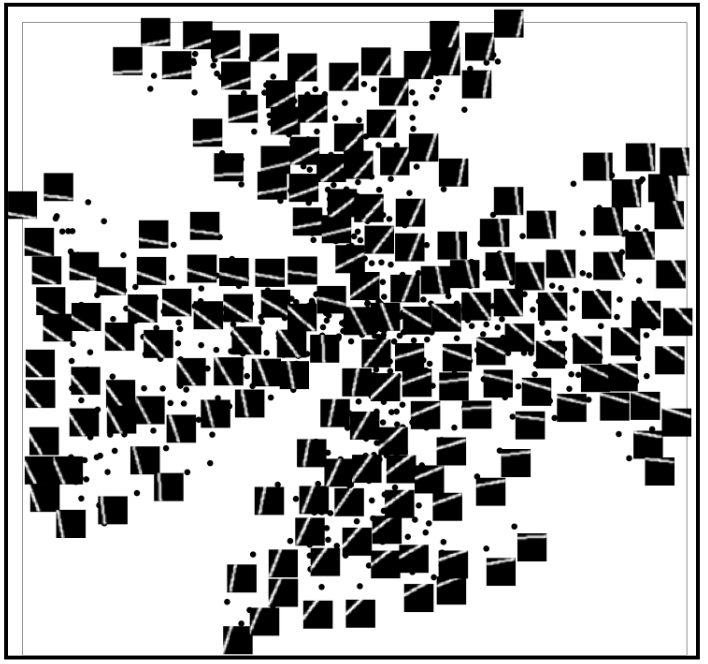


$n = 983$ inputs
$k = 8$ nearest neighbors
$m = 10$ basis vectors (LLE)

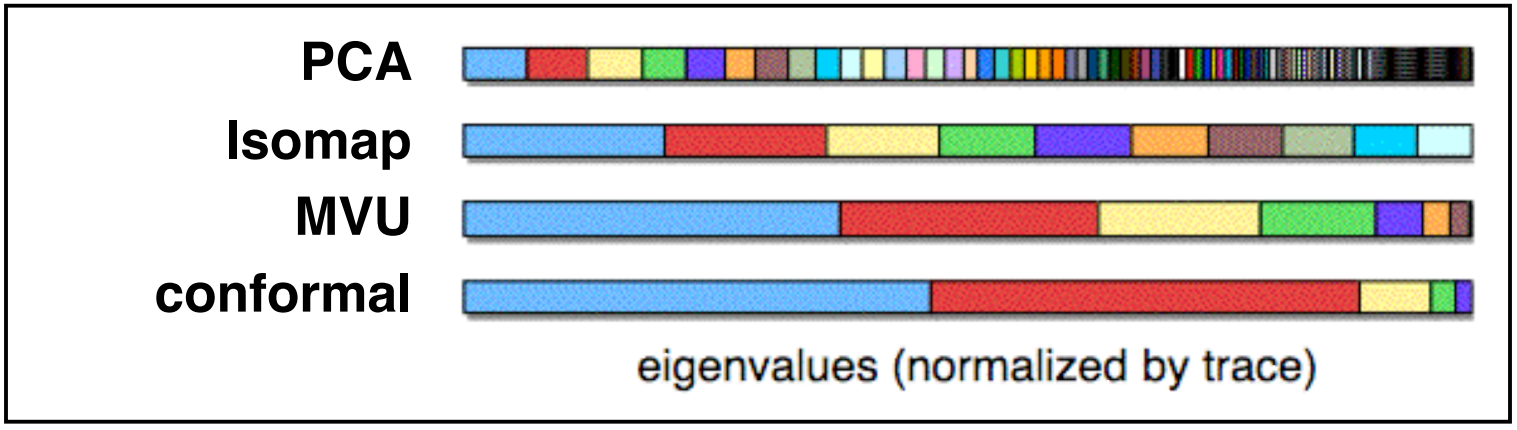


Images of Oriented Edges





$n = 2016$ images
$D = 24 \quad 24$ resolution
$m = 10$ (graph Laplacian)
$k = 10$ nearest neighbors



Open questions

- **Which basis functions?**

  LLE is based on symmetries; graph Laplacian on smoothness. Others?

- **Angle-vs-distance preserving?**

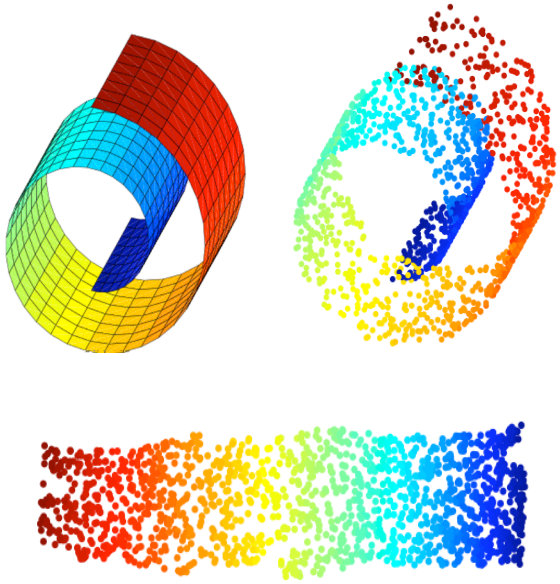
  More aggressive dimensionality reduction, but at what price?

- **Other types of learning?**

  Graph Laplacians have many uses. Where else can SDPs help?

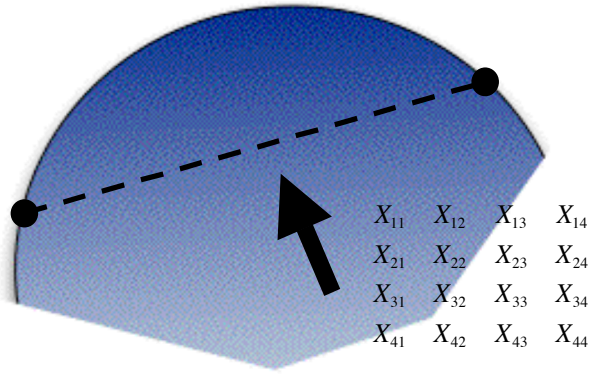
What does

nonlinear dimensionality reduction

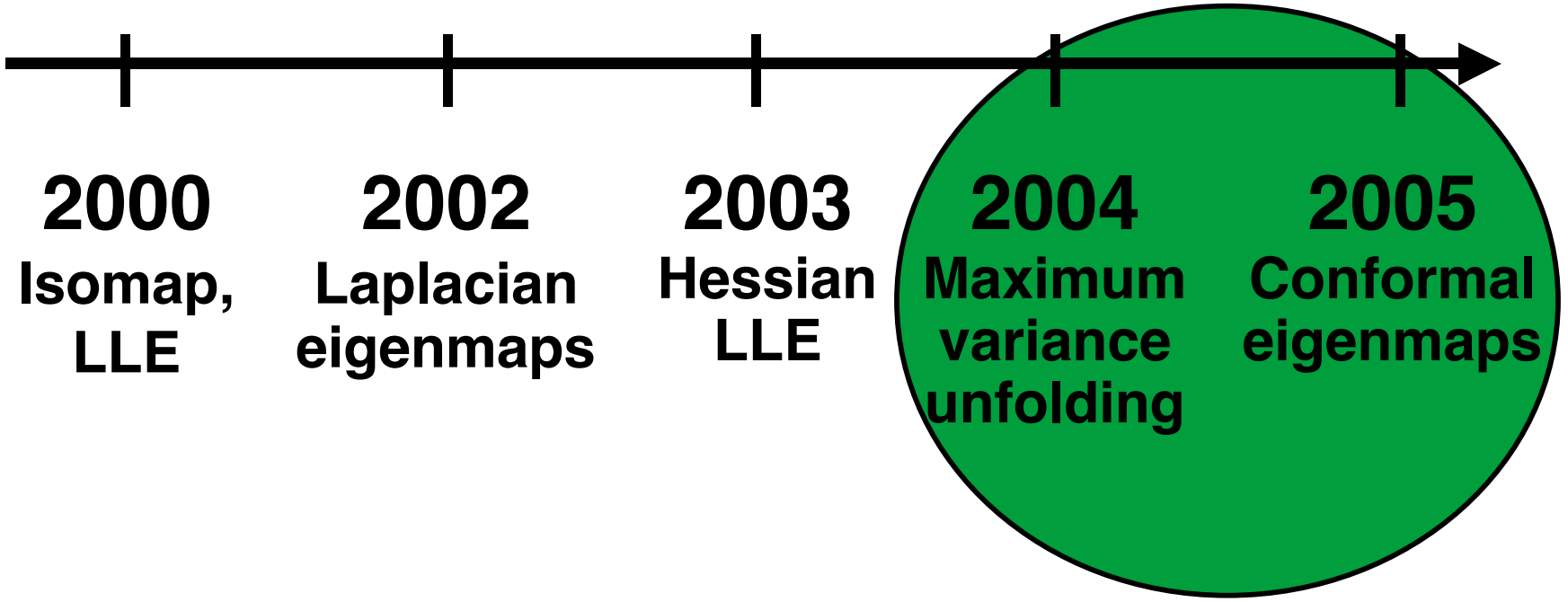


have to do with

semidefinite programming?



Manifold learning



SDPs have greatly expanded the types of optimizations we can perform.

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**

  LLE was inspired by properties of conformal maps. SDPs can enforce them.

Tomorrow…

- **Kernel methods in machine learning**
  - Nonlinear versions of linear models
  - Ex: kernel classifiers, kernel PCA
  - Relation to manifold learning?
- **More open problems**
  - Correspondences between manifolds
  - Spherical & toroidal geometries
  - Applications (vision, graphics, speech)

See you tomorrow…