Spectral Methods for Dimensionality Reduction

Prof. Lawrence Saul

Dept of Computer & Information Science
University of Pennsylvania

UCLA IPAM Tutorial, July 11-14, 2005

Penn
UNIVERSITY of PENNSYLVANIA

Institute for Pure & Applied Mathematics
IPAM
University of California, Los Angeles

Dimensionality reduction

- **Inputs (high dimensional)**

  $$\vec{x}_i \qquad {}^{D} \text{ with } i = 1, 2, \ldots, n$$

- **Outputs (low dimensional)**

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

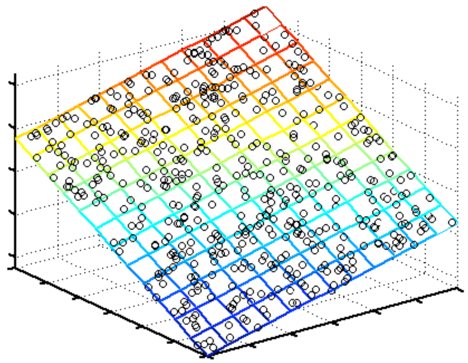
- **Goals**

  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?

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

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

Nonlinear Methods

- **Common framework**

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

  2) Derive matrix from graph weights.

  3) Derive embedding from eigenvectors.

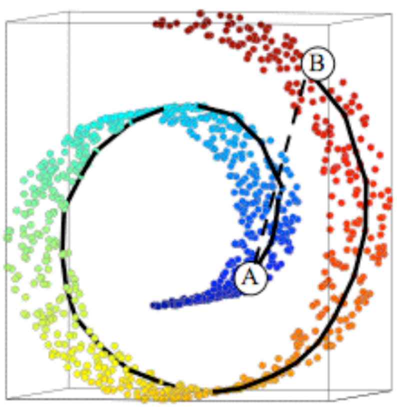
- **Varied solutions**

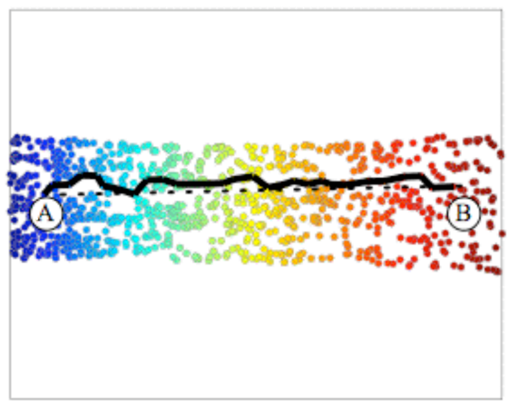
  Algorithms differ in step 2.

  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…

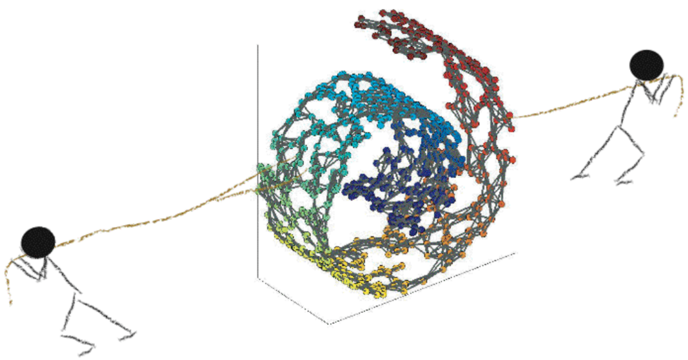
2000
Isomap, LLE

2002
Laplacian eigenmaps

2003
Hessian LLE

2004
Maximum variance unfolding

2005
Conformal eigenmaps



Maximize variance while respecting local distances, then apply MDS.

In sixty seconds or less…

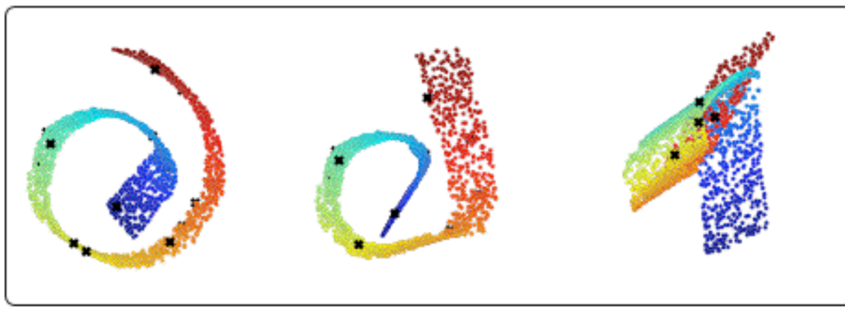
2000 Isomap, LLE

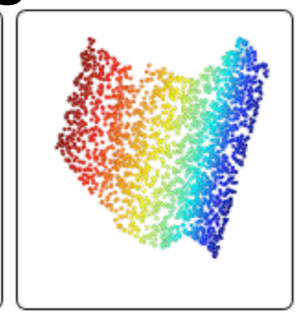
2002 Laplacian eigenmaps

2003 Hessian LLE

2004 Maximum variance unfolding

2005 Conformal eigenmaps





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

In sixty seconds or less…

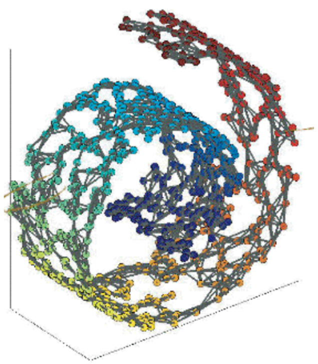
2000
Isomap, LLE

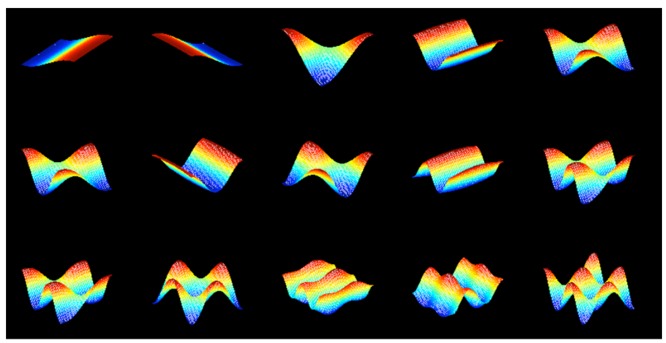
2002
Laplacian eigenmaps

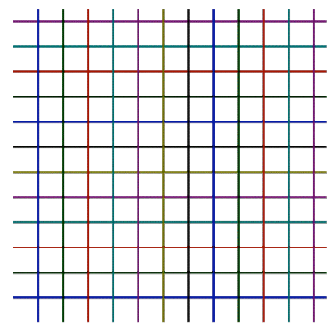
2003
Hessian LLE

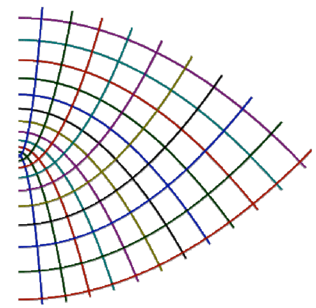
2004
Maximum variance unfolding

2005
Conformal eigenmaps









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

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

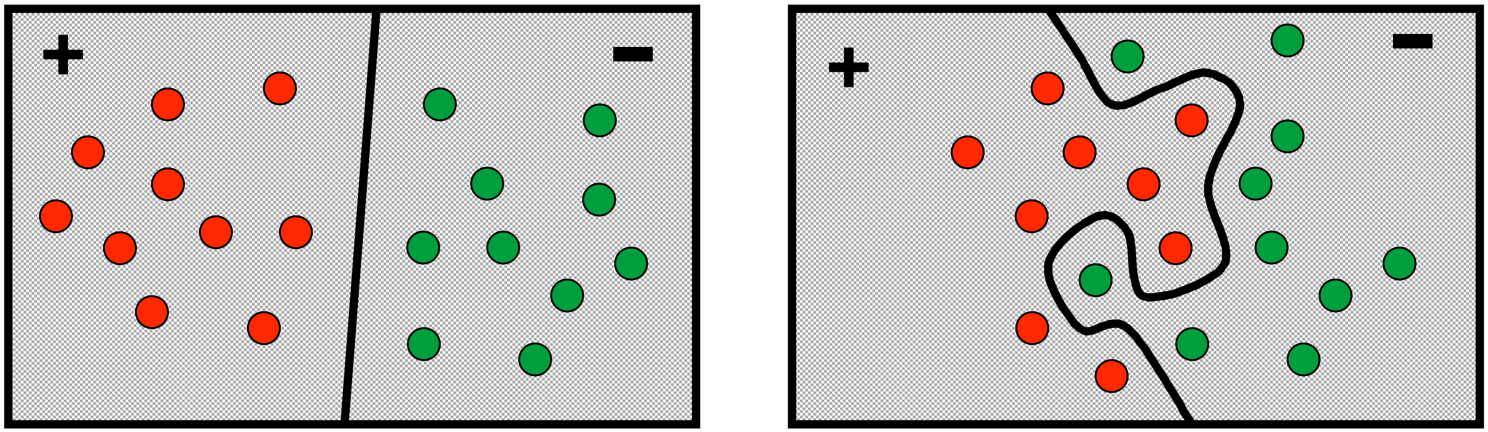
  http://www.cse.msu.edu/~lawhiu/manifold

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

Today

- Kernel methods in machine learning
  - Nonlinear versions of linear models
  - Ex: kernel classifiers, kernel PCA
  - Relation to manifold learning?
- Parting thoughts
  - Some interesting applications
  - Correspondences between manifolds
  - Open questions

Linear vs nonlinear



What computational price must we pay for nonlinear <u>classification</u>?

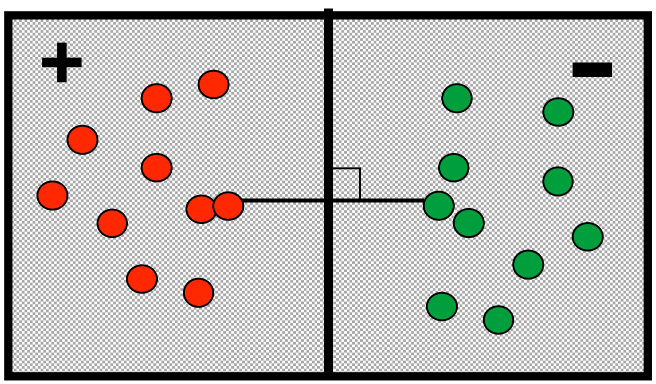
Linear classifier

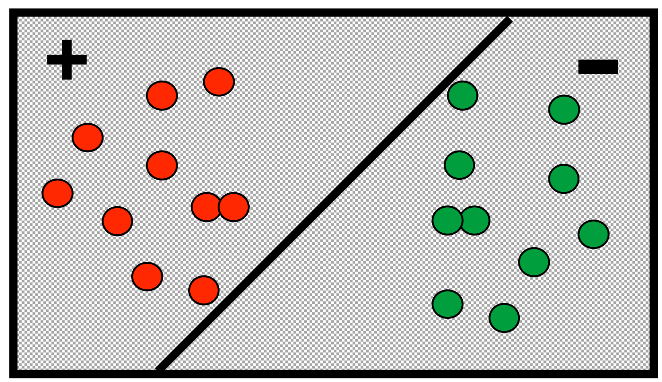
- **Training data**

inputs $\vec{x}_i \qquad {}^{D}$

outputs $y_i \qquad \{\ \ 1, +1\}$

- **Maximum margin hyperplane**

yes 

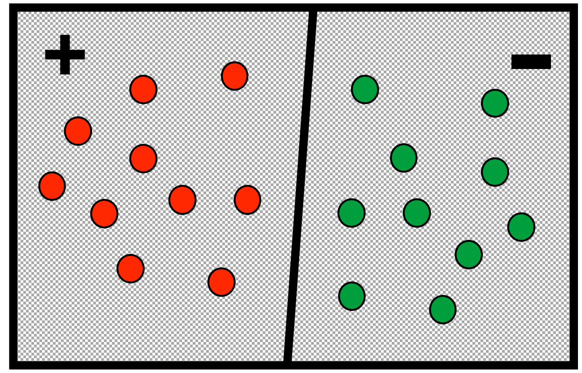


no

Convex optimization

- **Decision boundary**

$$y_i = \mathrm{sign}(\vec{w} \quad \vec{x}_i + b)$$



- **Maximum margin QP**

$$\min \|\vec{w}\|^2 \ \text{ such that } \ y_i(\vec{w} \quad \vec{x}_i + b) \quad 1$$

- **Hyperplane spanned by inputs**

$$\vec{w} = \quad {}_{i} \quad {}_{i} y_i \vec{x}_i$$

Problem is QP in coefficients $_i$.

Optimization

- **QP in coefficients**

$$\text{cost:} \quad \left\| \;{}_{ij} \quad {}_{i} \quad {}_{j} y_i y_j (\vec{x}_i \quad \vec{x}_j) \right\|^2$$

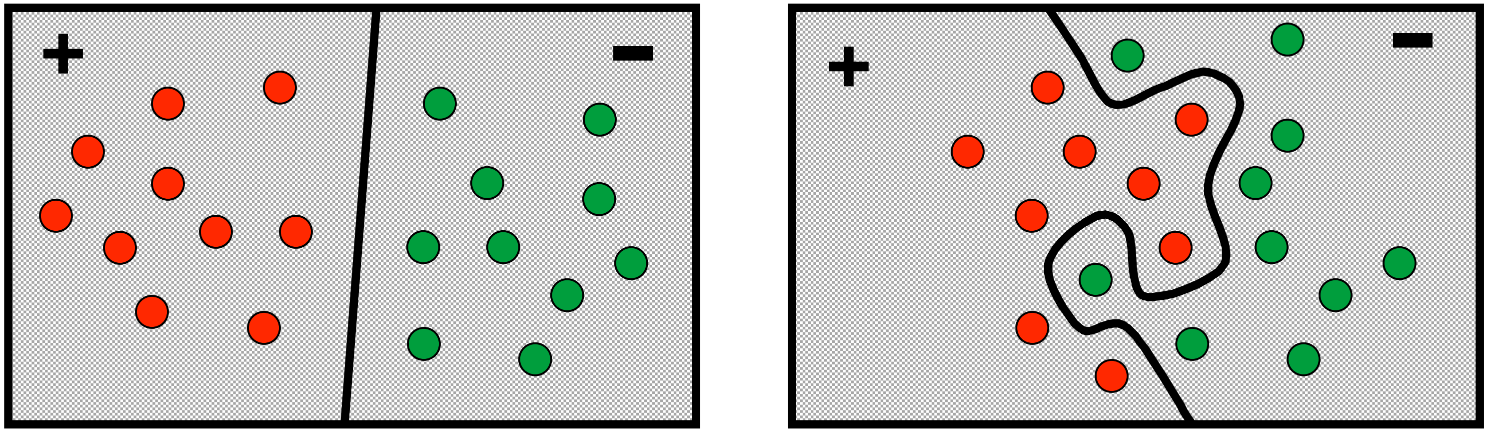
$$\text{constraints:} \quad y_i \qquad {}_{j} \quad {}_{j} y_j (\vec{x}_j \quad \vec{x}_i) + b \qquad 1$$

- **Inner products**

The optimization can be expressed purely in terms of inner products:

$$\boxed{G_{ij} = \vec{x}_i \quad \vec{x}_j}$$

Linear vs nonlinear



What computational price must we pay for nonlinear <u>classification</u>?

Kernel trick

- **Kernel function**

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

- **Implicit mapping**

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

$$\boxed{K(\vec{x},\vec{x}\ ) = \vec{\ }\,(\vec{x})\quad \vec{\ }\,(\vec{x}\ )}$$

Example

- **Gaussian kernel**

  Measure similarity between inputs by the real-valued function:

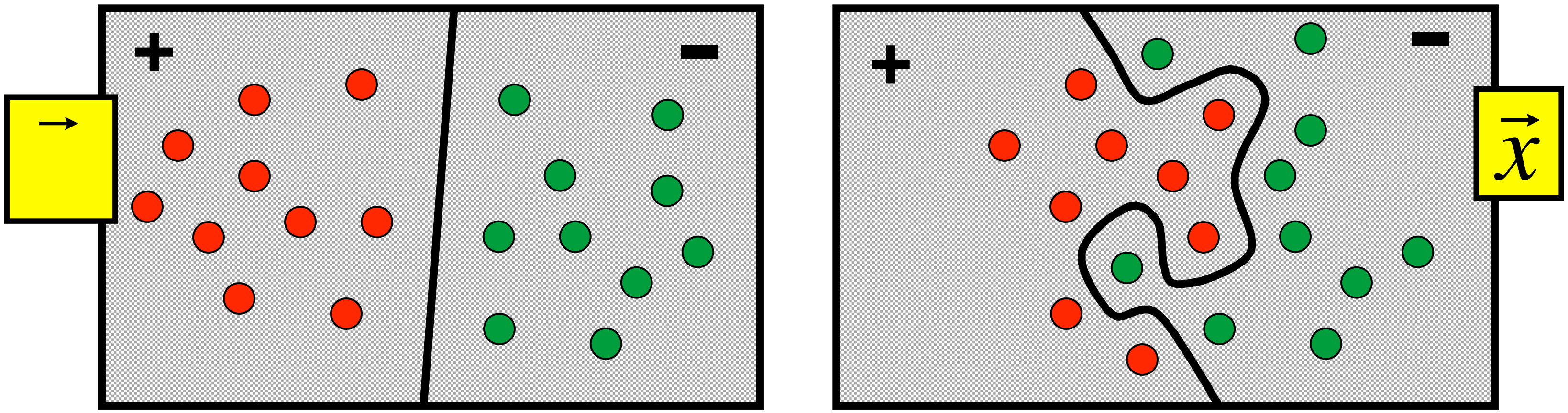
$$K(\vec{x},\vec{x}\ ) = \exp\left(\qquad \|\vec{x} \quad \vec{x}\ \|^2 \right)$$

- **Implicit mapping**

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

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

Nonlinear classification



Maximum margin hyperplane in feature space is nonlinear decision boundary in input space.

Old optimization

- **QP in coefficients**

$$\text{cost:} \quad \left\| \; {}_{ij} \quad {}_{i} \quad {}_{j} y_i y_j (\vec{x}_i \quad \vec{x}_j) \right\|^2$$

$$\text{constraints:} \quad y_i \qquad {}_{j} \quad {}_{j} y_j (\vec{x}_j \quad \vec{x}_i) + b \qquad 1$$

- **Inner products**

The optimization can be expressed purely in terms of inner products:

$$\boxed{G_{ij} = \vec{x}_i \quad \vec{x}_j}$$

New optimization

- **QP in coefficients**

$$\text{cost:} \quad \left\| \ {}_{ij} \ {}_{i} \ {}_{j} y_i y_j \boxed{K_{ij}} \right\|^2$$

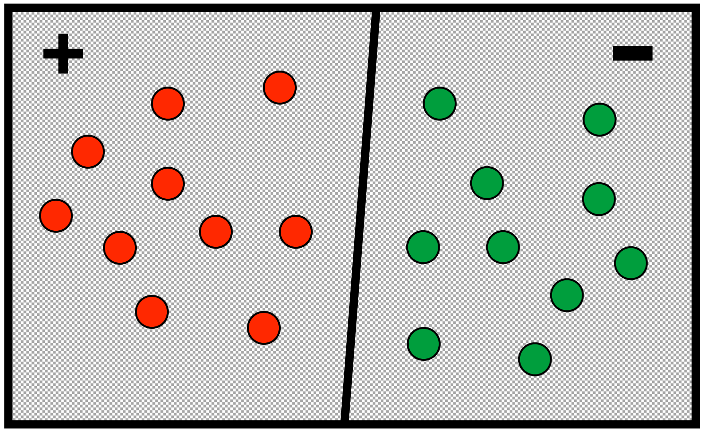
$$\text{constraints:} \quad y_i \ {}_{j} \ {}_{j} y_j \boxed{K_{ji}} + b \quad 1$$

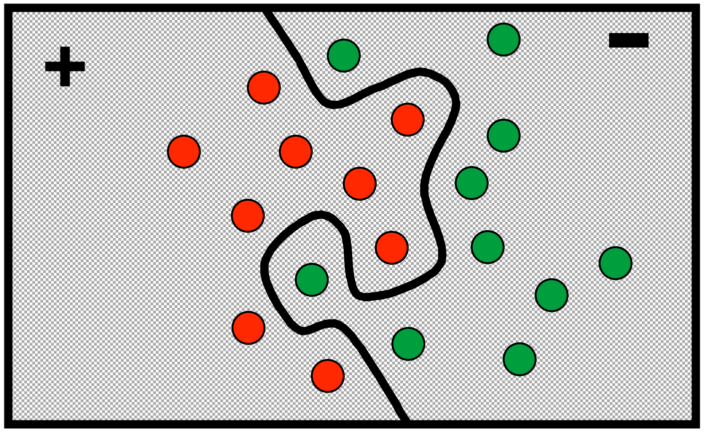
- **Inner products**

The optimization can be expressed purely in terms of the kernel matrix:

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

Linear vs nonlinear





What computational price must we pay for nonlinear <u>classification</u>? None.

Before vs after

- **Linear classifier**

  Compute decision boundary from maximum margin hyperplane.

- **Kernel trick** $\boxed{\vec{x}_i \cdot \vec{x}_j \rightarrow K(\vec{x}_i, \vec{x}_j)}$

  Substitute kernel function wherever inner products appear.

- **Nonlinear classifier**

  Optimization remains convex.
  Only heuristic is choosing the kernel.

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.

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

- **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!

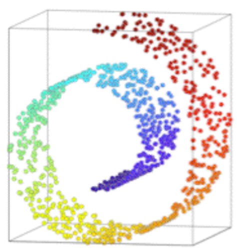
$$
\begin{aligned}
\left\|\vec{\ }_{i}\qquad\vec{\ }_{j}\right\|^{2} &= \left\|\vec{\ }_{i}\right\|^{2}+\left\|\vec{\ }_{j}\right\|^{2}\quad 2\,\vec{\ }_{i}\qquad\vec{\ }_{j}\\
&= K_{ii}+K_{jj}\quad 2K_{ij}\\
&\quad\ 2\quad\left\|\vec{x}_{i}\quad\vec{x}_{j}\right\|^{2}\ \text{for nearby inputs}
\end{aligned}
$$

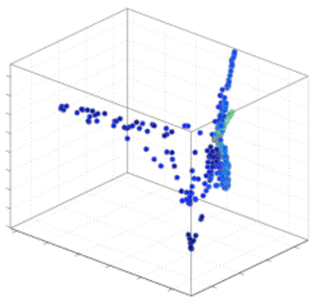
- **Manifold learning**

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

kPCA with Gaussian kernel

- **Swiss roll**





top three kernel principal components

$$K(\vec{x},\vec{x}\,) = \exp\Big(\quad \|\vec{x} \quad \vec{x}\,\|^2 \Big)$$

kPCA eigenvalues normalized by trace

- **Explanation**
  - Distant patches of manifold are mapped to orthogonal parts of feature space.
  - kPCA enumerates patches of radius $^{-1/2}$, fails terribly for dimensionality reduction.

kPCA and manifold learning

- **Generic kernels do not work**

  Gaussian $\boxed{K(\vec{x},\vec{x}\ ) = \exp\left(\quad \|\vec{x} \quad \vec{x}\ \|^2\right)}$

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

  Hyperbolic tangent $\boxed{K(\vec{x},\vec{x}\ ) = \tanh\left(\vec{x} \quad \vec{x}\ + \quad\right)}$

- **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 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:

$$\boxed{\frac{\quad}{t} = \quad L}$$

- **Metric space**

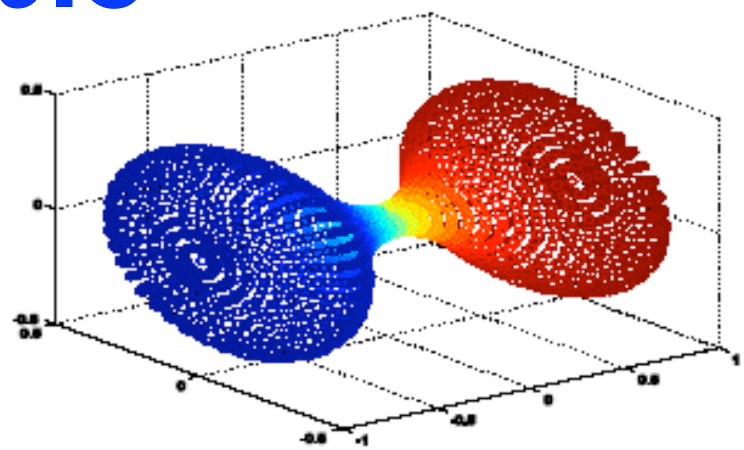
  Distances from pseudo-inverse are expected round-trip commute times:

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

Example

- **Barbell data set**

  Lobes are connected by bottleneck.

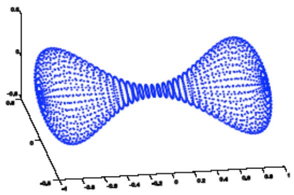


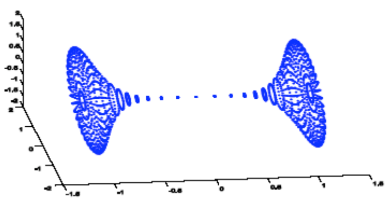
- **Comparison of induced geometries**

  + MVU will not alter barbell.

  + Laplacian will warp due to bottleneck.

  – Isomap will warp due to non-convexity.





(Coifman & Lafon)

Kernel methods

- **Unsupervised learning**

  Many spectral methods can be seen as learning a kernel matrix for kPCA.

- **Supervised learning**

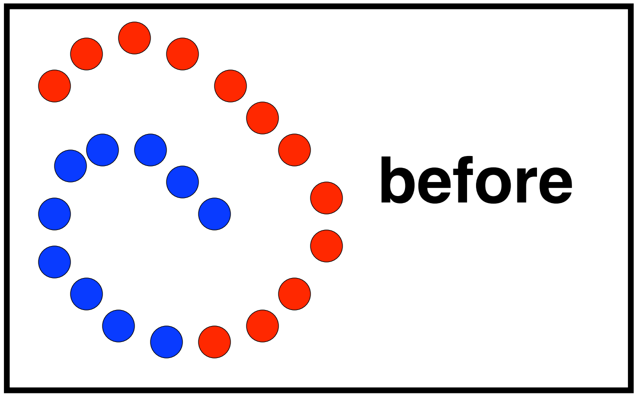
  Are these kernel matrices useful for classification?

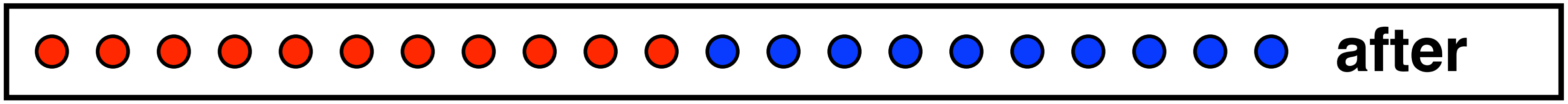
  Is learning manifold structure useful for classification?

An empirical question…

- **Best case scenario**

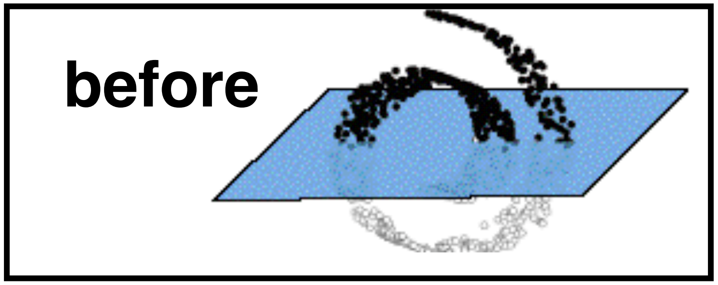
  Classification labels "follow" manifold.

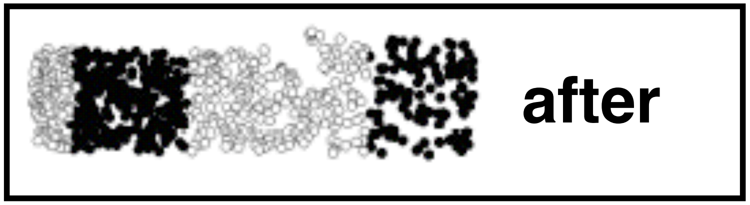




- **Worst case scenario**

  Classification labels "ignore" manifold.





Classification on manifolds

- **Empirically**

  Class boundaries are correlated with (but not completely linearized by) manifold coordinates.

- **How to exploit manifold structure?**

  How to integrate graph-based spectral methods into classifiers?

Semi-supervised learning

- **Problem**

  How to learn a classifier from few labeled but many unlabeled examples?

- **Solution**

  Learn manifold from unlabeled data.
  Optimize decision boundaries to:

  (1) classify labeled data correctly
  (2) vary smoothly along manifold

[Zhu et al, 2004; Belkin et al, 2004]

So far…

- **Algorithms**

  Isomap, LLE, Laplacian eigenmaps, maximum variance unfolding, etc.

- **Kernel methods**
  - **Manifold learning as kernel PCA**
  - **Graph-based kernels for classification**

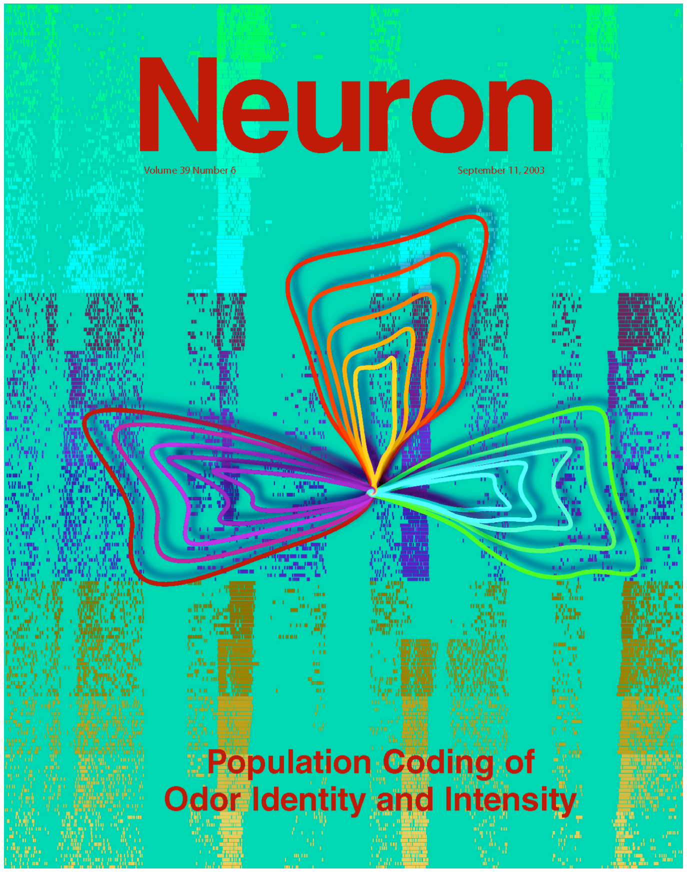
Interesting applications?

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)

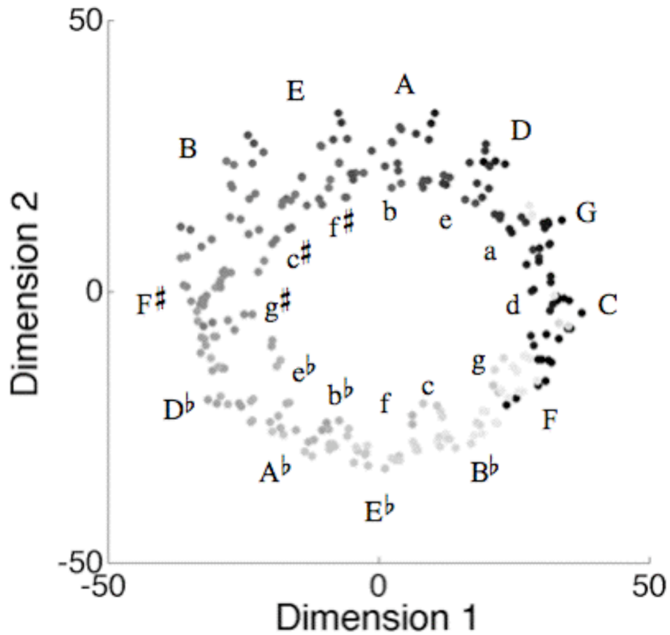


Visualization

- Tonal pitch space

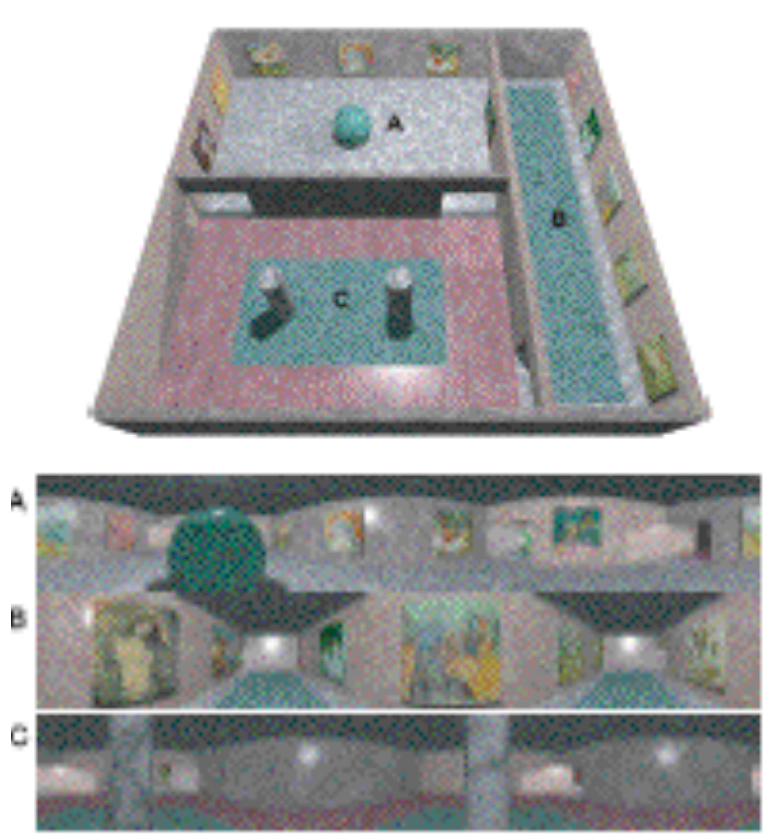
  Music theorists have defined distance functions between harmonies, such as C/C, C/g, C/C#, etc.

  (Burgoyne & Saul, 2005)

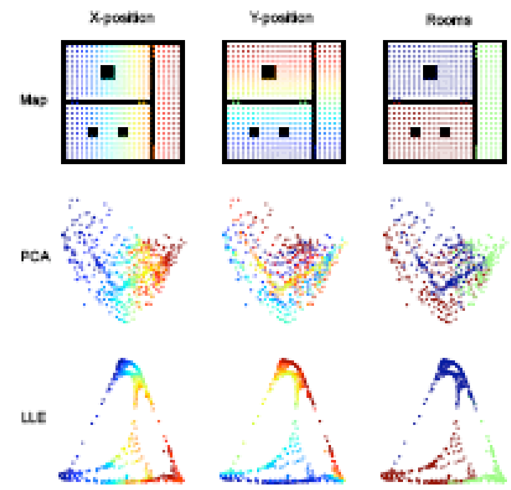


Circle of fifths (from MVU)

Robot localization (Ham , Lin, & Lee, 2005)

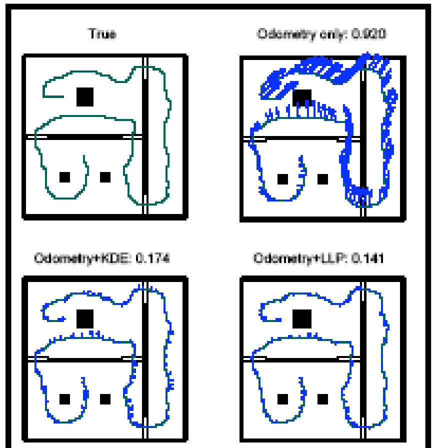


simulated environment and panoramic views



PCA

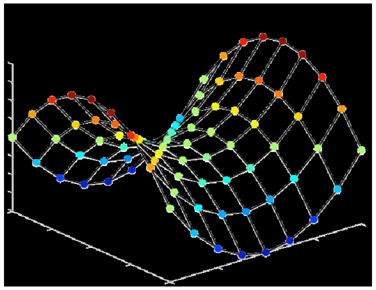
LLE



Supervised, improved by Bayesian filtering of odometer readings

Novelty detection?

(suggested to me this week)



Suppose that "normal" configurations lie on or near manifold?

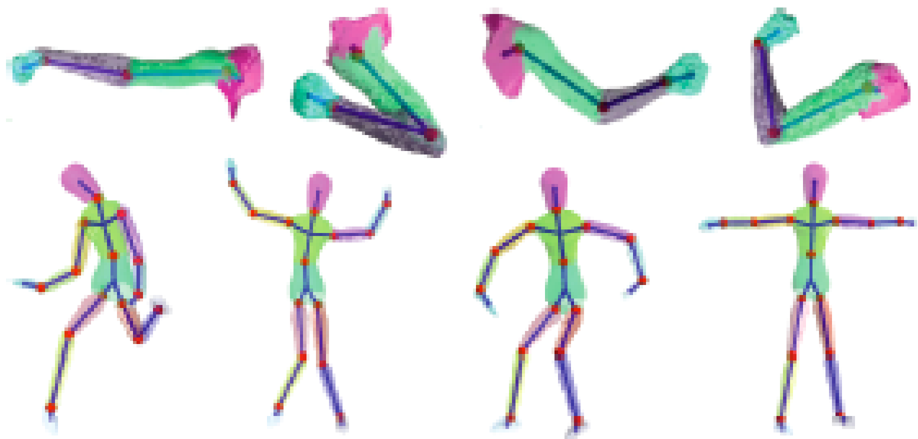
- **Network monitoring**

  How to detect that a network is about to crash?

- **Hyperspectral images**

  How to detect anomalies in a large digital library of images?

Surface registration?



Better methods by exploiting manifold structure?

Unsupervised registration of non-rigid surfaces from 3D laser scans

(figure from Anguelov et al, 2005)

Learning correspondences

✓ **So far:**

How to perform nonlinear dimensionality reduction on a single data set?

- **An interesting generalization:**

How to perform nonlinear dimensionality reduction on multiple data sets?

(Ham, Lee, & Saul, 2003, 2005)

Image correspondences

Images of objects at same pose are in correspondence.

http://www.bushorchimp.com

Correspondences

- **Out of one, many:**

  Many data sets share a common manifold structure.

- **Examples:**
  - **Facial expressions, vocalizations, joint angles of different subjects**
  - **Multimodal input: audiovisual speech, terrain images and inertial sensors**

How can we use this?

Learning from examples

- **Given:**

  n_1 **examples of object 1 in** D_1 **dimensions**

  n_2 **examples of object 2 in** D_2 **dimensions**

  n **labeled correspondences (**$n \ll n_1+n_2$**)**

- **Matrix form:**

$D_1 \times n$	$D_1 \times n_1$	?
$D_2 \times n$	?	$D_2 \times n_2$

Learning correspondences

$D_1 \times n$	$D_1 \times n_1$	?
$D_2 \times n$	?	$D_2 \times n_2$

- **Fill in the blanks:**

  How to map between objects?
  How to exploit shared structure?

- **Difficult nonlinear regression**

  Must learn shared low dimensional manifold to avoid overfitting.

Spectral method

$D_1 \times n$	$D_1 \times n_1$	?
$D_2 \times n$	?	$D_2 \times n_2$

- **Uncoupled graph Laplacians**

  Two separate problems: size $(n+N_1)$ for object 1, size $(n+N_2)$ for object 2.

- **Coupled graph Laplacian**

  Map matched inputs to same output.
  One problem of size $(n+N_1+N_2)$.

Multiple objects

- **Given:**

  n_i **examples of** i^{th} **object in** D_i **dimensions**
  n **labeled correspondences (**$n \ll \sum_i N_i$**)**

- **Matrix form:**

	?			$D_1 \times n_1$	?	?
		?		?	$D_2 \times n_2$	?
			?	?	?	$D_3 \times n_3$

Richer and more general than traditional framework for semisupervised learning…

Image correspondences

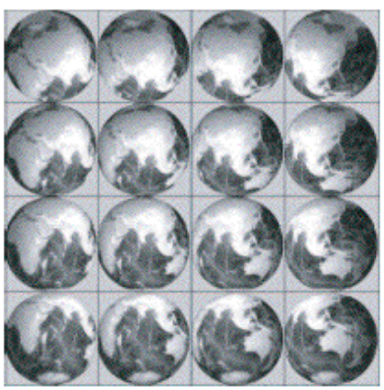
- **Partially labeled examples**

  841 images of student
  698 images of statue
  900 images of earth

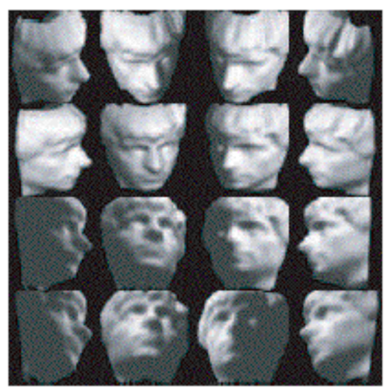
  25 labeled correspondences

- **From coupled graph Laplacian:**

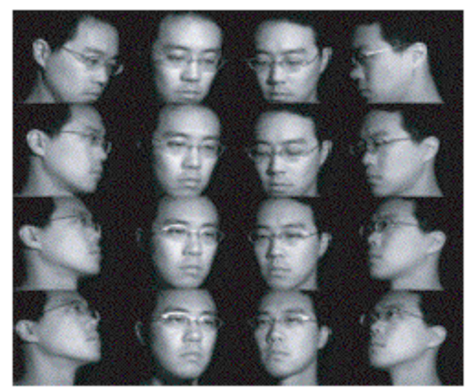
Queries (samples)



Match 1



Match 2



Elements of Manifold Learning

- **Statistics**
  - Discrete sampling of continuous pdf
  - High dimensional data analysis
- **Geometry**
  - Isometric (distance-preserving) maps
  - Conformal (angle-preserving) maps
- **Computation**
  - Spectral decompositions of graphs
  - Semidefinite programming

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)