Dimensionality Reduction

Sanjiv Kumar, Google Research, NY

EECS-6898, Columbia University - Fall, 2010

Curse of Dimensionality

Many learning techniques scale poorly with data dimensionality (d)

- Density estimation
  - For example, Gaussian Mixture Models (GMM) → need to estimate covariance matrices $O(d^2)$
- Nearest Neighbor Search $O(nd)$
  - Also, performance of trees and hashes suffers with high dimensionality
- Optimization techniques
  - First order methods scale $O(d)$ while second order $O(d^2)$
- Clustering, classification, regression,…

Curse of Dimensionality

Many learning techniques scale poorly with data dimensionality (d)

- Density estimation
  - For example, Gaussian Mixture Models (GMM) → need to estimate covariance matrices $O(d^2)$
- Nearest Neighbor Search $O(nd)$
  - Also, performance of trees and hashes suffers with high dimensionality
- Optimization techniques
  - First order methods scale $O(d)$ while second order $O(d^2)$
- Clustering, classification, regression,…

Data Visualization – hard to do in high-dimensional spaces

Dimensionality Reduction

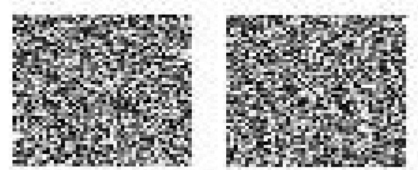
- Key Idea: Data dimensions in input space may be statistically dependent
  - possible to retain most of the information in input space in a lower dimensional space

Dimensionality Reduction

50 x 50 pixel faces

$\rightarrow \Re^{2500}$

50 x 50 pixel random images

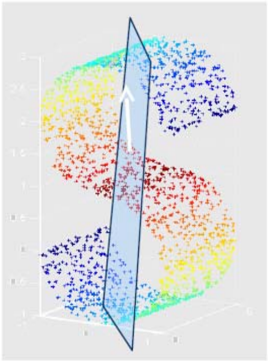


Space of face images significantly smaller than 256^{2500}

Want to recover the underlying low-dimensional space !

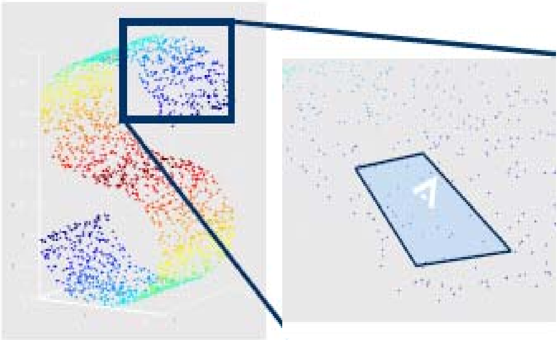
Dimensionality Reduction

Linear Techniques



- PCA, Metric MDS, Randomized projections
- Assume data lies in a subspace
- Work well in practice in many cases
- Can be a poor approximation for some data

Nonlinear Techniques



- Manifold learning methods
  - Kernel PCA, LLE, ISOMAP,…
- Assume local linearity of data
- Need densely sampled data as input
- Other approaches
  - Autoencoders (multi-layer Neural Networks),…
- Computationally more demanding than linear methods

Principal Component Analysis (PCA)

Two views but same solution

1. Want to find best linear reconstruction of the data that minimizes mean squared reconstruction error
2. Want to find best subspace that maximizes the projected data variance

data mean

Suppose input data $\{x_i\}_{i=1}^{n},\ x_i \in \Re^d$ is centered i.e., $x_i \leftarrow x_i - \mu$

Goal: To find a k-dim linear embedding y such that $k < d$

Principal Component Analysis (PCA)

Two views but same solution

1. Want to find best linear reconstruction of the data that minimizes mean squared reconstruction error
2. Want to find best subspace that maximizes the projected data variance

Suppose input data $\{x_i\}_{i=1}^n,\ x_i \in \Re^d$ is centered i.e., $x_i \leftarrow x_i - \mu$ data mean

Goal: To find a k-dim linear embedding y such that $k < d$

Reconstruction View $\tilde{x} = \sum_{j=1}^k y_j b_j = B\,y$ $\quad B \in \Re^{d\times k}, y \in \Re^{k\times 1}$

$$\arg\min_{B,y} \sum_{i=1}^n \|x_i - \tilde{x}_i\|_2^2 = \sum_{i=1}^n \|x_i - By_i\|_2^2 \text{ s.t. } B^T B = I$$

$$\boxed{\hat{B} = \arg\min_B \|X - BB^T X\|_F^2 \text{ s.t. } B^T B = I} \text{ and } \boxed{\hat{y} = \hat{B}^T X}$$

$d \times n$ data matrix

Solution: Get top k left singular vectors of X ($O(nd^2)$) and project data on them ($O(nkd)$)

Principal Component Analysis (PCA)

Max-Variance View

Want to find k-dim linear projection $y = B^T x$ such that

$$\boxed{\hat{B} = \arg\max_{B} Tr\left(B^T X X^T B\right) \quad \text{s.t.} \quad B^T B = I}$$

assuming data is centered

Solution: Get top k eigenvectors of XX^T ($O(nd^2+d^3)$) and project data ($O(nkd)$)

Left singular vectors of X = Eigenvectors of XX^T

Principal Component Analysis (PCA)

Max-Variance View

Want to find k-dim linear projection $y = B^T x$ such that

$$\boxed{\hat{B} = \arg\max_{B} Tr\left(B^T XX^T B\right) \quad \text{s.t.} \quad B^T B = I}$$

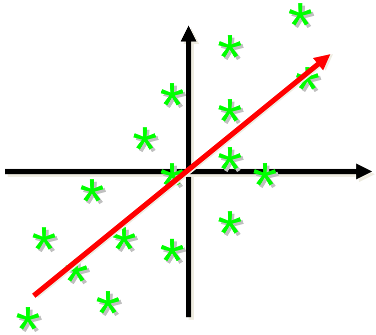
assuming data is centered

Solution: Get top k eigenvectors of XX^T ($O(nd^2+d^3)$) and project data ($O(nkd)$)

Left singular vectors of X = Eigenvectors of XX^T

Statistical assumption: Data is normally distributed

- More general versions (allow noise in the data)
  - Factor Analysis and Probabilistic PCA
- Can be extended to a nonlinear version using kernels



MultiDimensional Scaling (MDS)

Metric MDS: Given pairwise (Euclidean) distances among points, find a low-dim embedding that preserves the original distances

$$\hat{Y} = \arg\min_{Y} \sum_{i,j} \left(\left\| y_i - y_j \right\|_2 - d_{ij} \right)^2$$

$$x_i \in \Re^d, y_i \in \Re^k$$
$$X \in \Re^{d \times n}, Y \in \Re^{k \times n}$$
$$d_{ij} = \left\| x_i - x_j \right\|$$

MultiDimensional Scaling (MDS)

Metric MDS: Given pairwise (Euclidean) distances among points, find a low-dim embedding that preserves the original distances

$$\hat{Y} = \arg\min_{Y} \sum_{i,j} \left(\left\| y_i - y_j \right\|_2 - d_{ij} \right)^2$$

$$x_i \in \Re^d,\ y_i \in \Re^k$$
$$X \in \Re^{d \times n}, Y \in \Re^{k \times n}$$
$$d_{ij} = \left\| x_i - x_j \right\|$$

First, n x n distance matrix (D) is converted into a similarity matrix (K)

$$K = -\frac{1}{2} H\, D\, H \qquad D_{ij} = d_{ij}^2 \qquad H = I - \frac{1}{n} 1_n 1_n^T \qquad 1_n = \underbrace{[1,\ldots,1]}_{n \text{ entries}}^T$$

MultiDimensional Scaling (MDS)

Metric MDS: Given pairwise (Euclidean) distances among points, find a low-dim embedding that preserves the original distances

$$\hat{Y} = \arg\min_{Y} \sum_{i,j} \left(\|y_i - y_j\|_2 - d_{ij} \right)^2$$

$$x_i \in \Re^d, y_i \in \Re^k$$
$$X \in \Re^{d \times n}, Y \in \Re^{k \times n}$$
$$d_{ij} = \|x_i - x_j\|$$

First, n x n distance matrix (D) is converted into a similarity matrix (K)

$$K = -\frac{1}{2} H\, D\, H \qquad D_{ij} = d_{ij}^2 \qquad H = I - \frac{1}{n} 1_n 1_n^T \qquad 1_n = \underbrace{[1,\ldots,1]}_{n \text{ entries}}^T$$

Solution: Best k-dim ($k < d$) linear embedding Y given by

$$Y^T Y = K \approx U_k \Sigma_k U_k^T$$

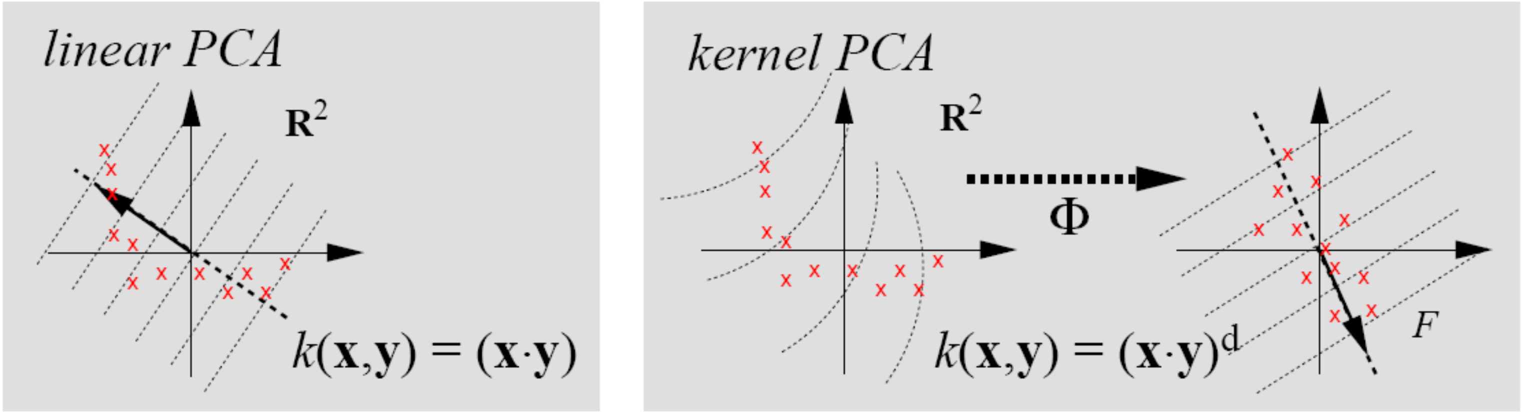
$$\boxed{Y = \Sigma_k^{1/2} U_k^T}$$

Embedding identical to that from PCA on X

Kernel (Nonlinear) PCA

Key Idea: Instead of finding linear projections in the original input space, do it in the (implicit) feature space induced by a mercer kernel

$$k(x,z) = \Phi(x)^T \Phi(x)$$



Scholkopf et al. [5]

Kernel (Nonlinear) PCA

Key Idea: Instead of finding linear projections in the original input space, do it in the (implicit) feature space induced by a mercer kernel

$k(x,z) = \Phi(x)^T \Phi(x)$

Let's focus on 1-dim projection,

	PCA	kernel PCA
Assumption: centered data	$\sum_{i=1}^n x_i = X1_n = 0$	$\sum_{i=1}^n \Phi(x_i) = \Phi(X)1_n = 0$
data covariance	$C = XX^T$	$C = \Phi(X)\Phi(X)^T$
best direction b	$Cb = \lambda b$	$\sum_{i=1}^n \Phi(x_i)\left(\Phi(x_i)^T b\right) = \lambda b$

$1_n = \underbrace{[1,\ldots,1]^T}_{n \text{ entries}}$

Kernel (Nonlinear) PCA

Key Idea: Instead of finding linear projections in the original input space, do it in the (implicit) feature space induced by a mercer kernel

$k(x,z) = \Phi(x)^T \Phi(x)$

Let's focus on 1-dim projection,

	PCA	**kernel PCA**
Assumption: centered data	$\sum_{i=1}^n x_i = X1_n = 0$	$\sum_{i=1}^n \Phi(x_i) = \Phi(X)1_n = 0$
data covariance	$C = XX^T$	$C = \Phi(X)\Phi(X)^T$
best direction b	$Cb = \lambda b$	$\sum_{i=1}^n \Phi(x_i)\left(\Phi(x_i)^T b\right) = \lambda b$

$1_n = \underbrace{[1,\ldots,1]^T}_{n \text{ entries}}$

$$b = \sum_{i=1}^n \Phi(x_i)\left(\Phi(x_i)^T b / \lambda\right) = \sum_{i=1}^n \alpha_i \Phi(x_i) \qquad \lambda \neq 0$$

$$= \Phi(X)\alpha$$

b lies in the span of mapped input points !

Kernel (Nonlinear) PCA

Key Idea: Instead of finding linear projections in the original input space, do it in the (implicit) feature space induced by a mercer kernel

$k(x,z) = \Phi(x)^T \Phi(x)$

Let's focus on 1-dim projection,

	PCA	**kernel PCA**	
Assumption: centered data	$\sum_{i=1}^n x_i = X1_n = 0$	$\sum_{i=1}^n \Phi(x_i) = \Phi(X)1_n = 0$	$1_n = \underbrace{[1,\ldots,1]^T}_{n \text{ entries}}$
data covariance	$C = XX^T$	$C = \Phi(X)\Phi(X)^T$	
best direction b	$Cb = \lambda b$	$\sum_{i=1}^n \Phi(x_i)\left(\Phi(x_i)^T b\right) = \lambda b$	

$$b = \sum_{i=1}^n \Phi(x_i)\left(\Phi(x_i)^T b/\lambda\right) = \sum_{i=1}^n \alpha_i \Phi(x_i) \quad \lambda \neq 0$$

$$= \Phi(X)\alpha$$

b lies in the span of mapped input points !

$$\Phi(X)\Phi(X)^T \Phi(X)\alpha = \lambda \Phi(X)\alpha$$

Premultiply by $\Phi(X)^T$ and replace $\Phi(X)^T \Phi(X) = K$

$$K^2\alpha = \lambda K\alpha \Rightarrow K\alpha = \lambda\alpha$$

K is positive-definite (otherwise, other solutions are not of interest)

Kernel (Nonlinear) PCA

Main Computation: Find top k eigenvectors of kernel matrix

$$\boxed{K\alpha = \lambda\alpha} \qquad O(n^2 k)!$$

Final solution: $b_j = \Phi(X)\alpha_j$ but need to have unit-length

Kernel (Nonlinear) PCA

Main Computation: Find top k eigenvectors of kernel matrix

$$\boxed{K\alpha = \lambda\alpha} \qquad O(n^2 k)!$$

Final solution: $b_j = \Phi(X)\alpha_j$ but need to have unit-length

$$b_j^T b_j = 1 \Rightarrow \alpha_j^T K \alpha_j = 1$$

$$\Rightarrow \lambda_j \alpha_j^T \alpha_j = 1 \Rightarrow \|\alpha_j\|_2 = 1/\sqrt{\lambda_j}$$

Projection of a point x_i $\boxed{y_j = \Phi(x_i)^T b_j = (1/\sqrt{\lambda_j}) K_i^T \alpha_j}$

i^{th} column of K

Kernel (Nonlinear) PCA

Main Computation: Find top k eigenvectors of kernel matrix

$$\boxed{K\alpha = \lambda\alpha} \qquad O(n^2 k)!$$

Final solution: $b_j = \Phi(X)\alpha_j$ but need to have unit-length

$$b_j^T b_j = 1 \Rightarrow \alpha_j^T K \alpha_j = 1$$

$$\Rightarrow \lambda_j \alpha_j^T \alpha_j = 1 \Rightarrow \|\alpha_j\|_2 = 1/\sqrt{\lambda_j}$$

Projection of a point x_i

$$\boxed{y_j = \Phi(x_i)^T b_j = (1/\sqrt{\lambda_j}) K_i^T \alpha_j}$$

i^{th} column of K

What if we want to find a projection for a new point not seen during training ?

- Known as "out-of-sample" extension
- Not as straightforward as for linear PCA
- Can be thought of as adding another row and column in kernel Matrix
- To avoid recomputing eigendecomposition of extended Kernel matrix, use Nystrom method to approximate the new embedding (recall matrix approximations)

Centering in Feature Space

We assumed that data was centered in feature space

- Easy to do with features $\{x_i\}$
- How to do it in mapped feature space $\{\Phi(x_i)\}$ as explicit mapping may be unknown ?

Centering in Feature Space

We assumed that data was centered in feature space

- Easy to do with features $\{x_i\}$
- How to do it in mapped feature space $\{\Phi(x_i)\}$ as explicit mapping may be unknown ?

We want: $\overline{\Phi} = (1/n)\sum_{i=1}^{n}\Phi(x_i)$ $\quad\quad \Phi(x_i) \leftarrow \Phi(x_i) - \overline{\Phi}$

But we need data only through kernel matrix, so get “centered” kernel matrix

$$\boxed{\widetilde{K} = K - 1_{nn}K - K1_{nn} + 1_{nn}K1_{nn}}$$

$$(1_{nn})_{ij} = 1/n, \;\; i, j = 1,\ldots,n$$

Centering in Feature Space

We assumed that data was centered in feature space

- Easy to do with features $\{x_i\}$
- How to do it in mapped feature space $\{\Phi(x_i)\}$ as explicit mapping may be unknown ?

We want: $\overline{\Phi} = (1/n)\sum_{i=1}^{n}\Phi(x_i)$ $\qquad \Phi(x_i) \leftarrow \Phi(x_i) - \overline{\Phi}$

But we need data only through kernel matrix, so get "centered" kernel matrix

$$\boxed{\tilde{K} = K - 1_{nn}K - K1_{nn} + 1_{nn}K1_{nn}}$$

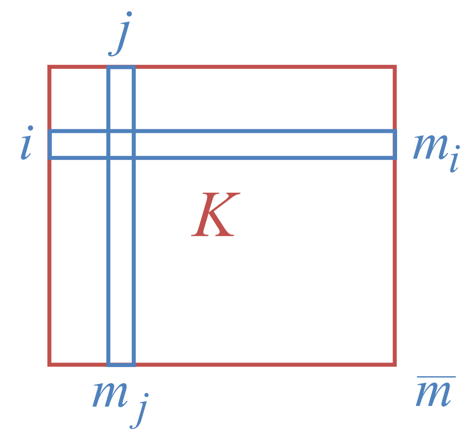
$$(1_{nn})_{ij} = 1/n, \;\; i, j = 1,\ldots,n$$

Interpretation $\tilde{K}_{ij} = K_{ij} - m_i - m_j + \overline{m}$

mean of i^{th} row

mean of j^{th} col

mean of all entries in K



Locally Linear Embedding (LLE)

Key Idea: Given sufficient samples, each data point and its neighbors are assumed to lie close to a locally linear patch.

- Try to reconstruct each data point from its t neighbors $O(n^2 d)$

$$x_i \approx \sum_{j \sim i} w_{ij} x_j \qquad j \sim i \text{ indicates neighbors of } i$$

Locally Linear Embedding (LLE)

Key Idea: Given sufficient samples, each data point and its neighbors are assumed to lie close to a locally linear patch.

- Try to reconstruct each data point from its t neighbors $O(n^2 d)$

$$x_i \approx \sum_{j \sim i} w_{ij} x_j \qquad j \sim i \text{ indicates neighbors of } i$$

- Learn the weights by solving $O(dnt^3)$

$$\arg\min_{w} \sum_i \left\| x_i - \sum_{j \sim i} w_{ij} x_j \right\|^2 \quad \text{s.t.} \quad \sum_{j \sim i} w_{ij} = 1$$

Locally Linear Embedding (LLE)

Key Idea: Given sufficient samples, each data point and its neighbors are assumed to lie close to a locally linear patch.

- Try to reconstruct each data point from its t neighbors $O(n^2 d)$

$$x_i \approx \sum_{j \sim i} w_{ij} x_j \qquad j \sim i \text{ indicates neighbors of } i$$

- Learn the weights by solving $O(dnt^3)$

$$\arg\min_{w} \sum_i \left\| x_i - \sum_{j \sim i} w_{ij} x_j \right\|^2 \text{ s.t. } \sum_{j \sim i} w_{ij} = 1$$

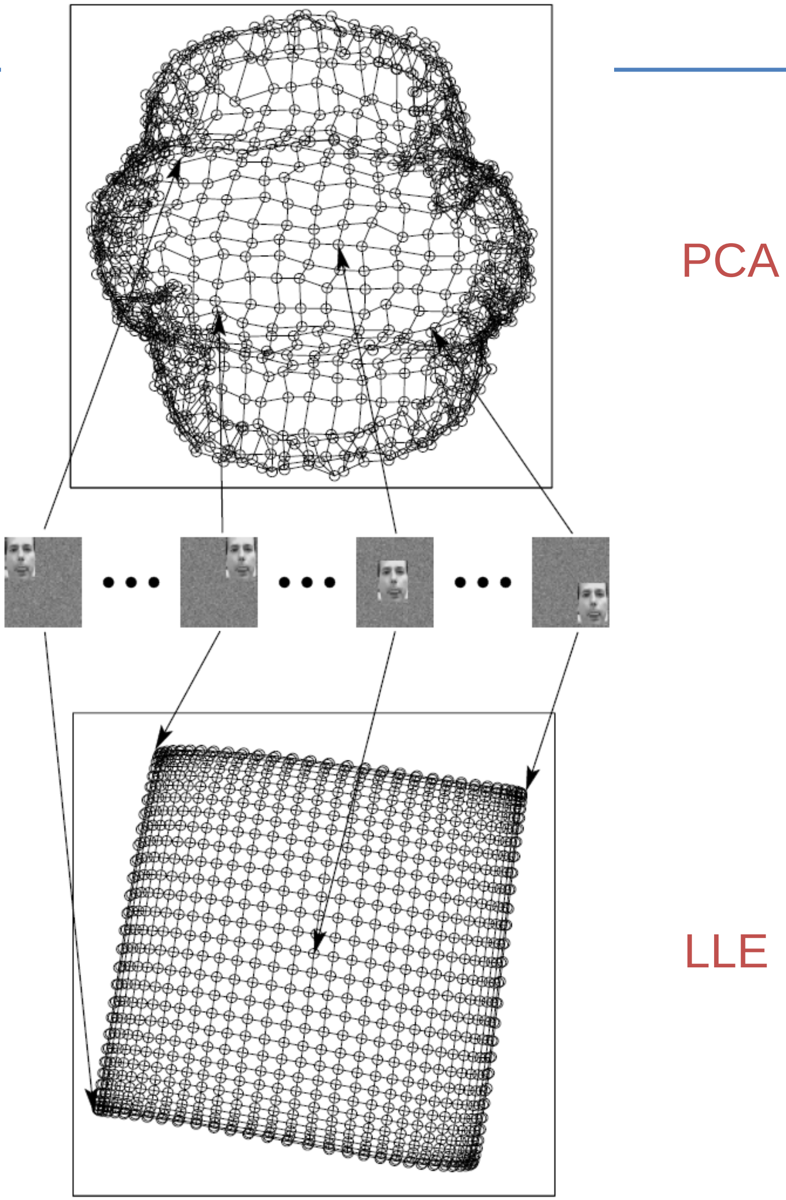
- Assumption: Same weights reconstruct the low-dim embedding also

$$\arg\min_{Y} \sum_i \left\| y_i - \sum_{j \sim i} w_{ij} y_j \right\|^2 \text{ s.t. } \sum_i y_i = 0 \quad (1/n) \sum_i y_i y_i^T = I$$

construct a sparse n x n matrix $M = (I - W)^T (I - W)$ Get bottom k eigenvectors ignoring the last $O(n^2 k)$

PCA vs LLE

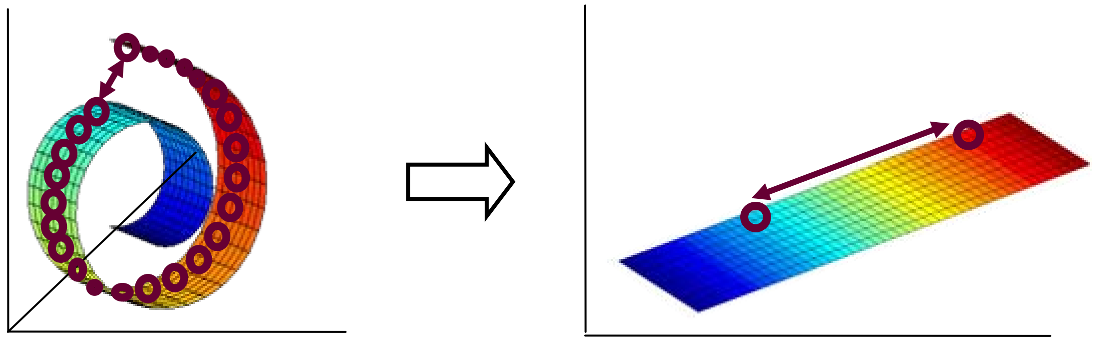
A face image translated in space against random background
n = 961, d = 3009, t = 4, k = 2



Roweis & Saul [5]

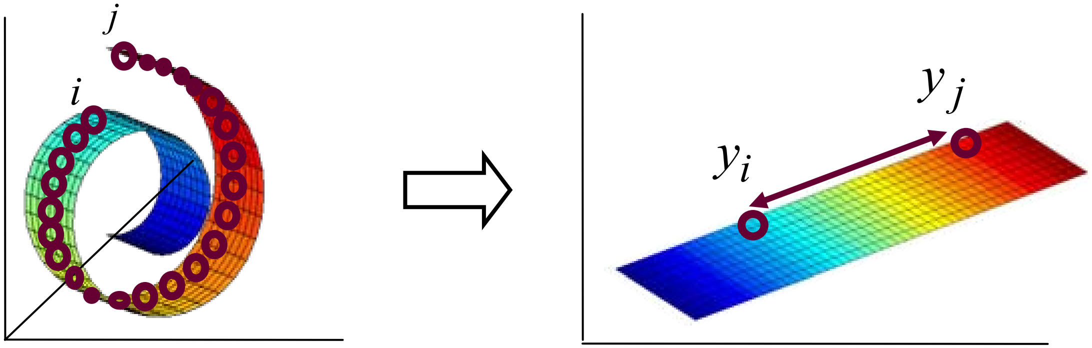
ISOMAP

Find the low-dimensional representation that best preserves geodesic distances between points → MDS with geodesic distances



ISOMAP

Find the low-dimensional representation that best preserves geodesic distances between points → MDS with geodesic distances



Output co-ordinates $\hat{Y} = \arg\min_{Y} \sum_{i,j} \left(\left\| y_i - y_j \right\|_2 - \Delta_{ij} \right)^2$

Geodesic distance

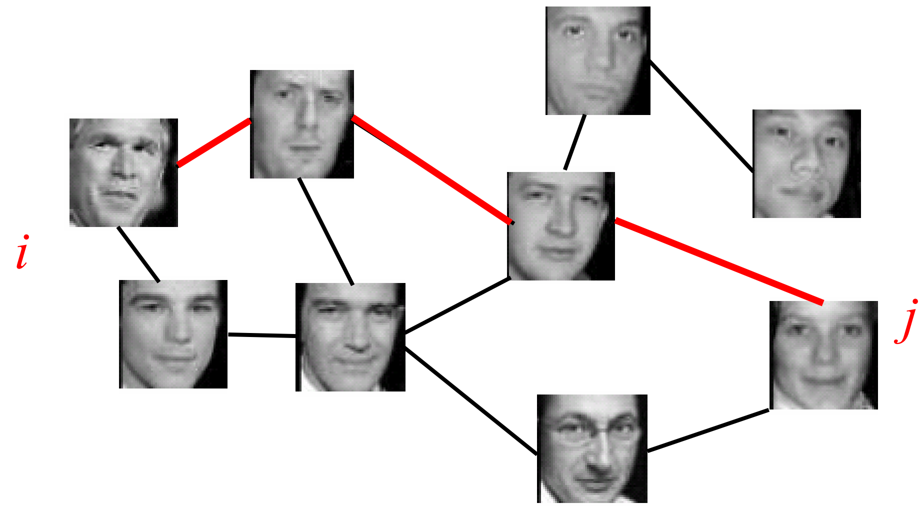
Recovers true (convex) manifold asymptotically !

ISOMAP

Given n input points:

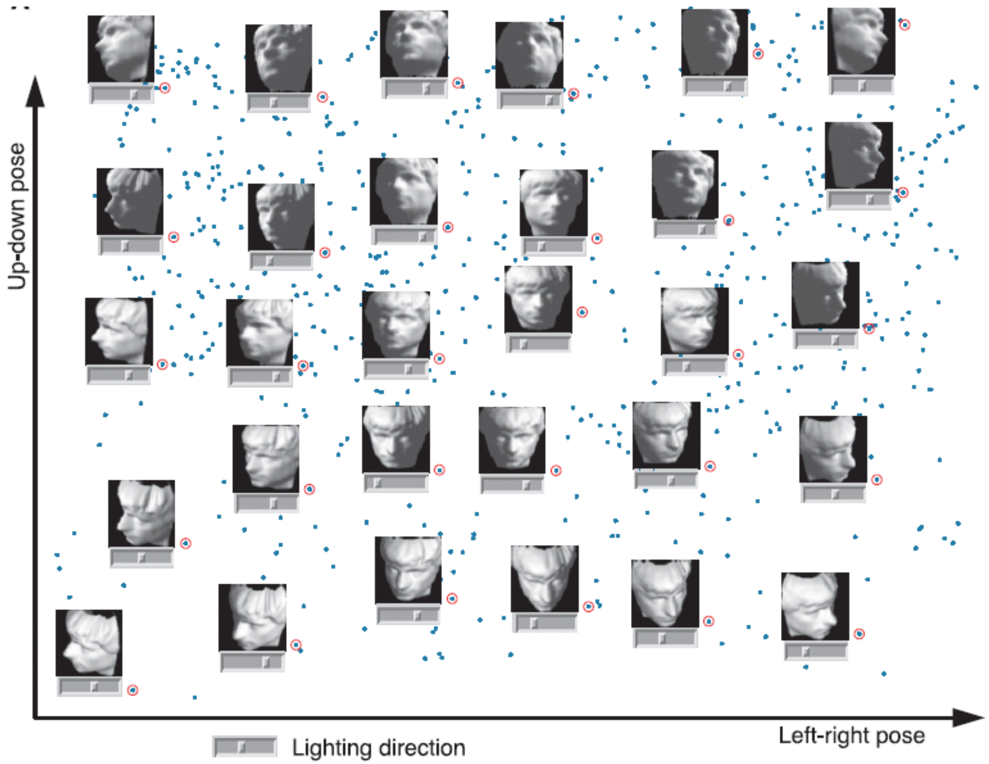
1. Find t nearest neighbors for each point : $O(n^2)$
2. Find shortest path distance for every (i, j), Δ_{ij} : $O(n^2 \log n)$
3. Construct $n \times n$ matrix K with entries as centered Δ_{ij}^2
   – K is a dense matrix
4. Optimal k reduced dims: $\Sigma_k^{1/2} U_k^T$ $O(n^2 k)$!

Eigenvalues Eigenvectors



ISOMAP Experiment

Face image taken with two pose variations (left-right and up-down), and 1-D illumination direction, d = 4096, n = 698



Tanenbaum et al. [7]

Issue:

- Quite sensitive to false edges in the graph ("short-circuit")
- One wrong edge may cause the shortest paths to change drastically
- Better to use expected commute time between two nodes → Laplacian Eigenmaps

Laplacian Eigenmaps

Minimize weighted distances between neighbors

$$\hat{Y} = \arg\min_{Y} \sum_{i \sim j} \left(\frac{W_{ij} \left\| y_i - y_j \right\|_2^2}{\sqrt{D_{ii} D_{jj}}} \right) \qquad D_{ii} = \sum_j W_{ij}$$

Another formulation

$$\hat{Y} = \arg\min_{Y} Tr[Y^T L Y] \qquad L = D - W$$

s.t $\quad Y^T D Y = I$

Laplacian Eigenmaps

Minimize weighted distances between neighbors

1. Find t nearest neighbors for each point : $O(n^2)$

2. Compute weight matrix W:
$$W_{ij} = \begin{cases} \exp(-\|x_i - x_j\|^2/\sigma^2) & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$

3. Compute normalized laplacian
$$K = I - D^{-1/2} W D^{-1/2} \quad \text{where} \quad D_{ii} = \sum_j W_{ij}$$

4. Optimal k reduced dims: U_k

Bottom eigenvectors of K ignoring last

$O(n^2 k)$ but can do much faster using Arnoldi's/Lanczos method since matrix is sparse

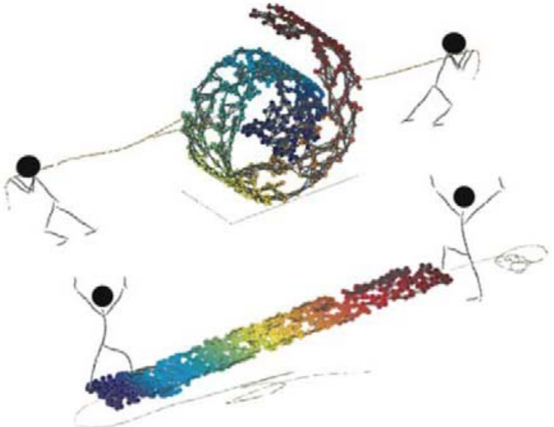
Maximum Variance Unfolding (MVU)

Key Idea: Find embedding with maximum variance that Preserves angles and lengths for edges between nearest neighbors

Angles/distances preservation constraint

$$\left\| y_i - y_j \right\|^2 = \left\| x_i - x_j \right\|^2$$

If there is an edge (i, j) in the graph formed by pairwise connecting all t nearest neighbors



Weinberger and Saul [12]

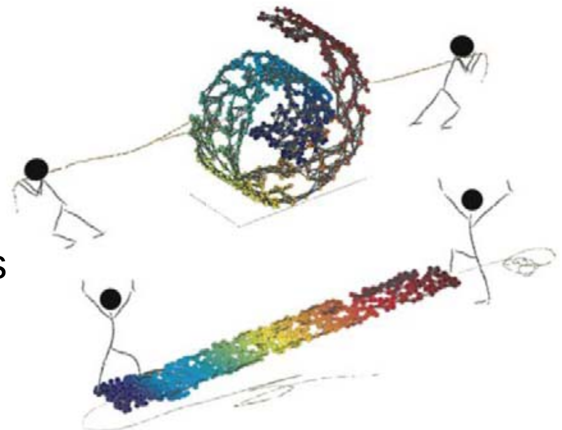
Maximum Variance Unfolding (MVU)

Key Idea: Find embedding with maximum variance that Preserves angles and lengths for edges between nearest neighbors

Angles/distances preservation constraint

$$\left\| y_i - y_j \right\|^2 = \left\| x_i - x_j \right\|^2$$

If there is an edge (i, j) in the graph formed by pairwise connecting all t nearest neighbors



Weinberger and Saul [12]

Centering constraint (for translational invariance)

$$\sum_i y_i = 0$$

Optimization Criterion

– Maximize squared pairwise distances between embeddings

$$\arg\max_Y \sum_{i,j} \left\| y_i - y_j \right\|^2 \qquad \text{s.t. above constraints}$$

Same as maximizing variance of the outputs !

Maximum Variance Unfolding (MVU)

Reformulation: Using a kernel K, such that $K_{ij} = y_i^T y_j$

Angles/distances preservation

$$K_{ii} - 2K_{ij} + K_{jj} = d_{ij}^2 = \left\| x_i - x_j \right\|^2$$

Maximum Variance Unfolding (MVU)

Reformulation: Using a kernel K, such that $K_{ij} = y_i^T y_j$

Angles/distances preservation

$$K_{ii} - 2K_{ij} + K_{jj} = d_{ij}^2 = \left\| x_i - x_j \right\|^2$$

Centering constraint

$$\sum_i y_i = 0 \Rightarrow \left\| \sum_i y_i \right\|^2 = \sum_{ij} K_{ij} = 0$$

Symmetric Positive-Definite constraint

$K \succeq 0$ Semi-Definite Program ! O(n^3+c^3)

\# of constraints

Maximum Variance Unfolding (MVU)

Reformulation: Using a kernel K, such that $K_{ij} = y_i^T y_j$

Angles/distances preservation

$$K_{ii} - 2K_{ij} + K_{jj} = d_{ij}^2 = \left\| x_i - x_j \right\|^2$$

Centering constraint

$$\sum_i y_i = 0 \Rightarrow \left\| \sum_i y_i \right\|^2 = \sum_{ij} K_{ij} = 0$$

Symmetric Positive-Definite constraint

$K \succeq 0$ Semi-Definite Program ! $O(n^3 + c^3)$

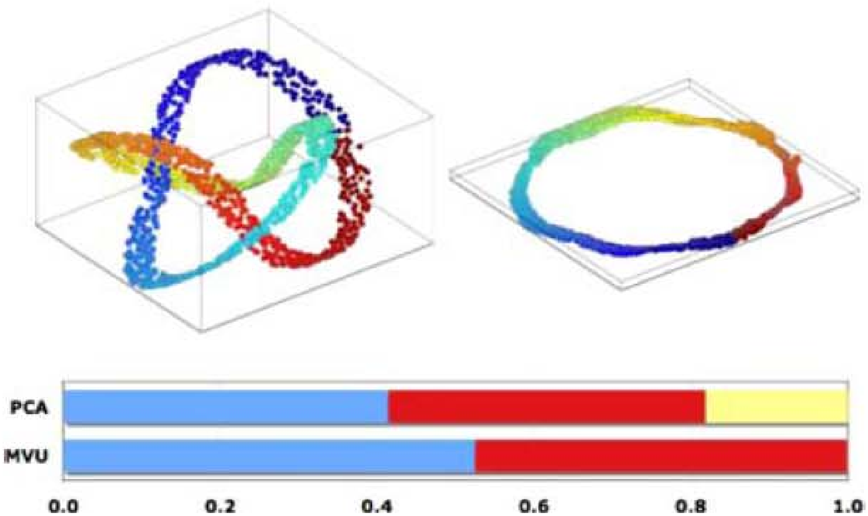
\# of constraints

Max-variance objective function $Tr(K)$

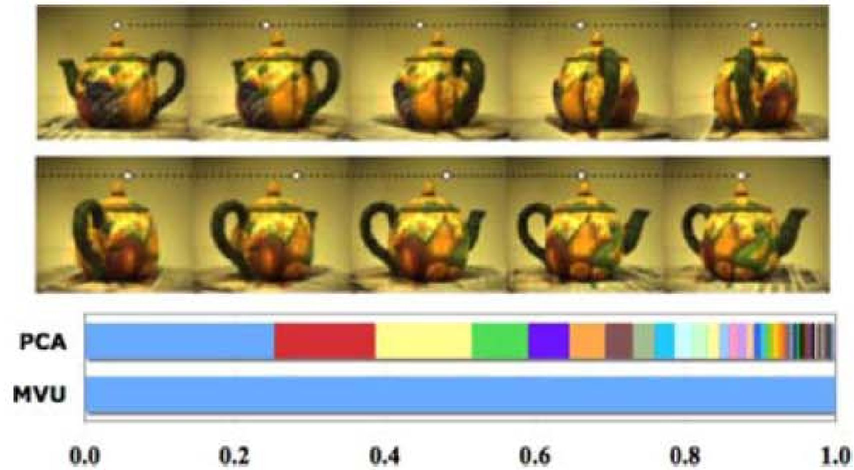
Final solution $Y = \Sigma_k^{1/2} U_k^T$ Top k eigenvalues and eigenvectors of K

Can relax the hard constraints via slack variables !

PCA vs MVU



Trefoil knot, n = 1617, d = 3, t = 5, k = 2



A teapot viewed rotated 180 deg in a plane, n = 200, d = 23028, t = 4, k = 1

Weinberger and Saul [12]

Large-Scale Face Manifold Learning

Construct Web dataset

- Extracted 18M faces from 2.5B internet images
- ~15 hours on 500 machines
- Faces normalized to zero mean and unit variance

Graph construction

- Approx Nearest Neighbor – Spill Trees
- 5 NN, ~2 days   Can be done much faster using appropriate hashes !

Talwalkar, Kumar, Rowley [13]

Neighborhood Graph Construction

Connect each node (face) with its neighbors

Is the graph connected?

- Depth-First-Search to find largest connected component
- 10 minutes on a single machine
- Largest component depends on number of NN (t)

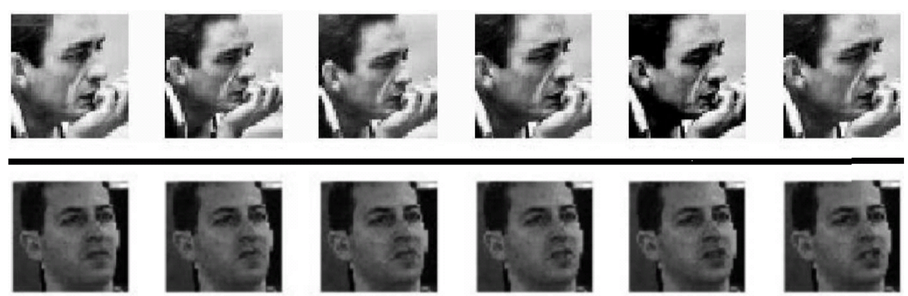
t	# Comp	% Largest
1	4.3M	0.03 %
2	285K	80.1 %
3	277K	82.2 %
5	275K	83.1 %

Talwalkar, Kumar, Rowley [13]

Samples from connected components

From Largest Component

From Smaller Components



Talwalkar, Kumar, Rowley [13]

Graph Manipulation

Approximating Geodesics

- Shortest paths between pairs of face images
- Computing for all pairs infeasible $O(n^2 \log n)$!

Key Idea: Need only a few columns of K for sampling-based spectral decomposition

- require shortest paths between a few (l) nodes and all other nodes
- 1 hour on 500 machines (l = 10K)

Computing Embeddings (k = 100)

- Nystrom: 1.5 hours, 500 machine
- Col-Sampling: 6 hours, 500 machines
- Projections: 15 mins, 500 machines

Talwalkar, Kumar, Rowley [13]

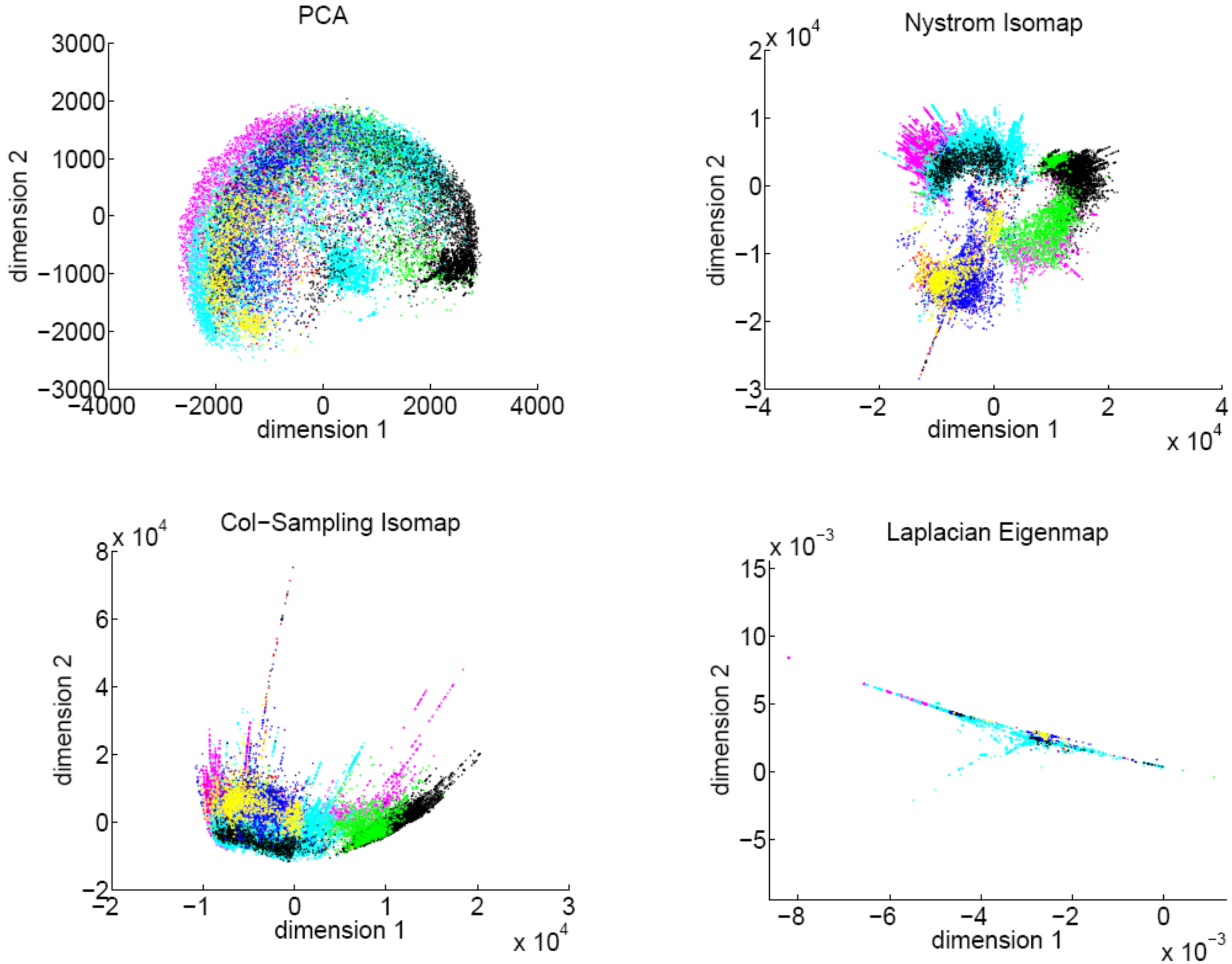
CMU-PIE Dataset

68 people, 13 poses, 43 illuminations, 4 expressions

35,247 faces detected by a face detector

Classification and clustering on poses

Optimal 2D embeddings



Talwalkar, Kumar, Rowley [13]

Clustering

K-means clustering after transformation (k = 100)

– K fixed to be the same as number of classes

Two metrics

Purity - points within a cluster come from the same class

Accuracy - points from a class form a single cluster

Methods	Purity (%)	Accuracy (%)
PCA	54.6 ($\pm$1.3)	46.8 ($\pm$1.3)
Nyström Isomap	59.9 ($\pm$1.5)	53.7 ($\pm$4.4)
Col-Sampling Isomap	56.5 ($\pm$0.7)	49.4 ($\pm$3.8)
Laplacian Eigenmap	39.3 ($\pm$4.9)	74.7 ($\pm$5.1)

Matrix K is not guaranteed to be positive semi-definite in Isomap !

- Nystrom: EVD of W (can ignore negative eigenvalues)
- Col-sampling: SVD of C (signs are lost) !

Talwalkar, Kumar, Rowley [13]

Experiments - Classification

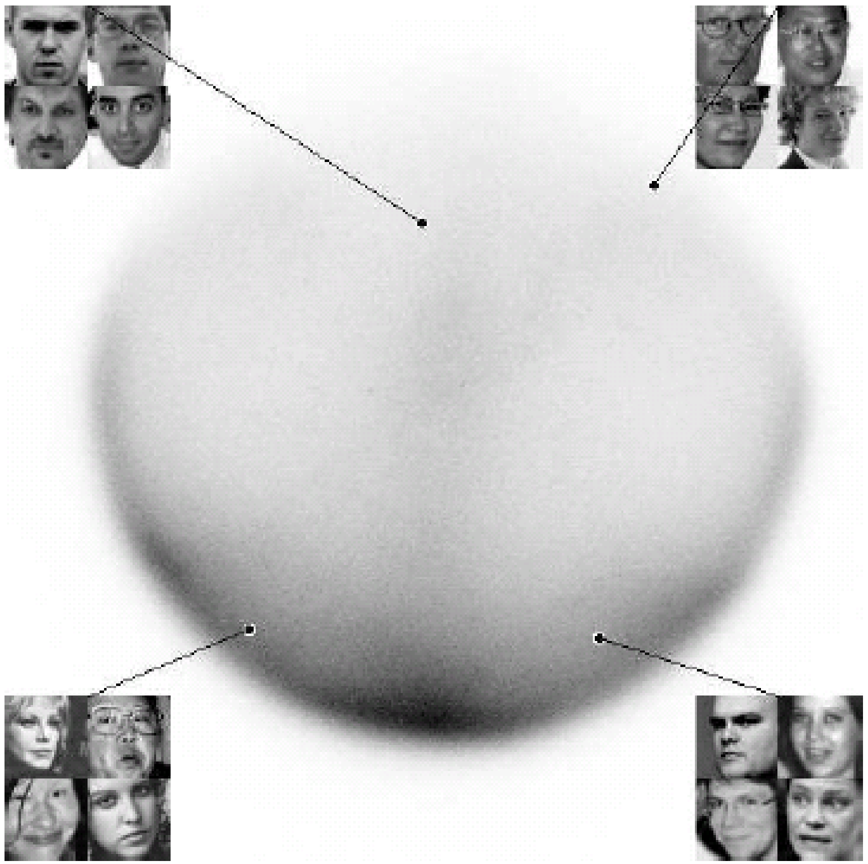
K-Nearest Neighbor Classification after Embedding

(%) Classification error for 10 random splits

Methods	$K = 1$	$K = 3$
Nyström Isomap	11.0 ($\pm$0.5)	14.0 ($\pm$0.6)
Col-Sampling Isomap	12.0 ($\pm$0.4)	15.3 ($\pm$0.6)
Laplacian Eigenmap	12.7 ($\pm$0.7)	16.6 ($\pm$0.5)

Talwalkar, Kumar, Rowley [13]

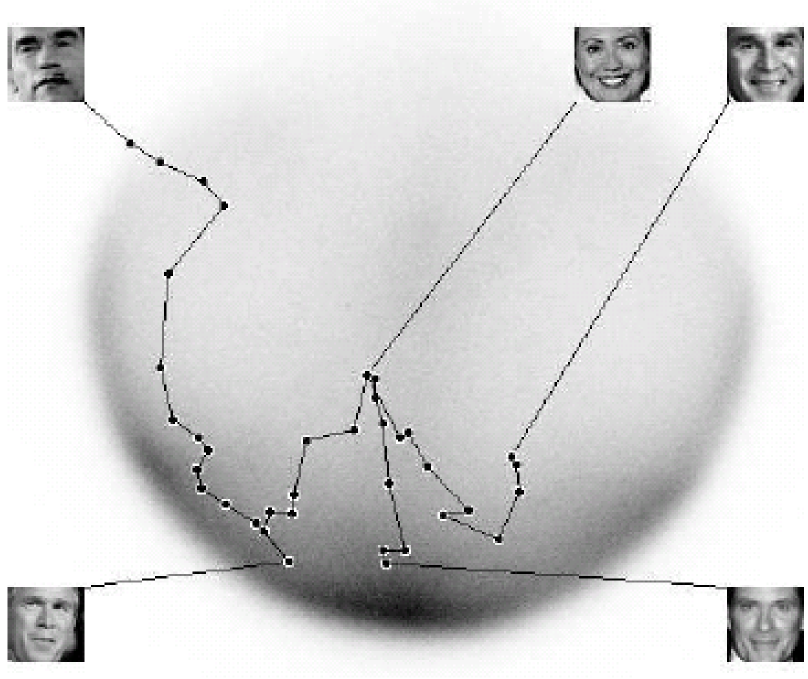
18M-Manifold in 2D



Nystrom Isomap

Talwalkar, Kumar, Rowley [13]

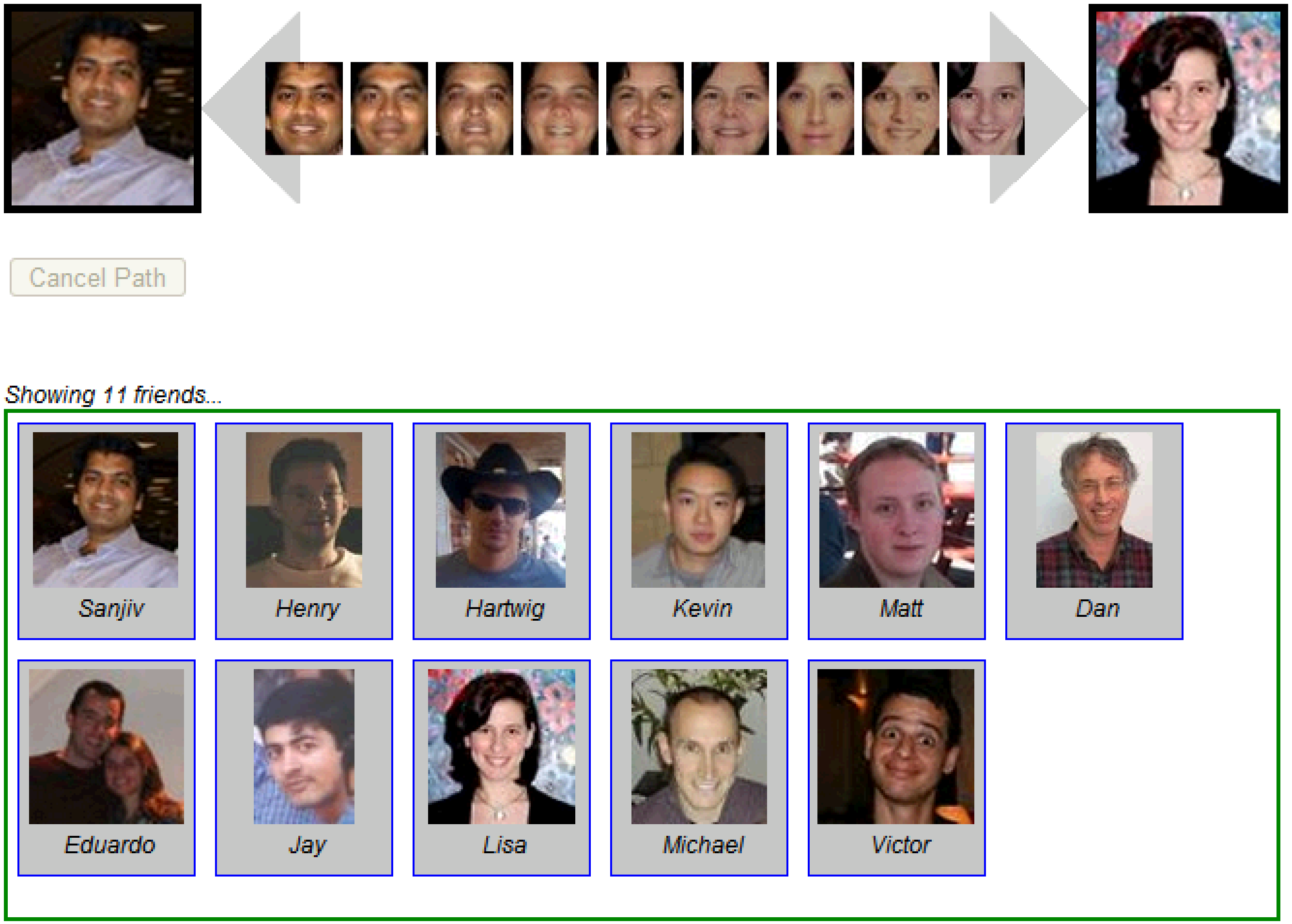
Shortest Paths on Manifold



18M samples not enough!

Talwalkar, Kumar, Rowley [13]

People Hopper Interface



Orkut Gadget

Manifold Learning – Open Questions

- Does a manifold really exist for a given dataset?
- Is it really connected or convex?
- Instead of lying on a manifold, may be data lives in small clusters in different subspaces?
- Any practical benefits of nonlinear dimensionality reduction (manifold learning) in clustering/classification?
  - Most of the results on toy data, no real practical utility so far
  - In practice, PCA enough to give most of the benefits (if any)
- Instead of looking for yet another manifold learning method, better to focus on solving if a manifold exists and how to quantify that

References

1. K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space". *Philosophical Magazine* **2** (6): 559–572. 1901.
2. C. Spearman, “General Intelligence, Objectively Determined and Measured,” American Journal of Psychology, 1904. (factor analysis)
3. I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag. pp. 487, 1986.
4. T. Cox, & M. Cox. Multidimensional scaling. Chapman & Hall, 1994.
5. B. Schölkopf, A. Smola, K.-R. Muller, Kernel Principal Component Analysis, In: Bernhard Schölkopf, Christopher J. C. Burges, Alexander J. Smola (Eds.), Advances in Kernel Methods-Support Vector Learning, 1999, MIT Press Cambridge, MA, USA, 327–352.
6. S. T. Roweis and L. K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” *Science,* December 2000.
7. J. B. Tenenbaum, V. de Silva and J. C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” *Science* 290 (5500): 2319-2323, 2000.
8. M. Belkin and P. Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, Advances in Neural Information Processing Systems 14, 2001, p. 586-691.
9. D. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data" Proc Natl Acad Sci U S A. 2003 May 13; 100(10): 5591–5596.
10. Y. Bengio, J F Paiement, P. Vincent, O. Delalleau, N. Le Roux, M. Ouimet, “Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering,” NIPS, 2004.
11. G. E. Hinton* and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, 2006, Vol. 313. no. 5786, pp. 504 - 507.
12. K. Q. Weinberger and L. K. Saul, “Unsupervised Learning of Image Manifolds by Semidefinite Programming,” International Journal of Computer Vision (IJCV), 70(1), 2006.
13. A. Talwalkar, S. Kumar and H. Rowley, “Large Scale Manifold Learning,” CVPR, 2008.
14. B. Shaw and T. Jebara, “Structure Preserving Embedding”, ICML, 2009.