A Geometric Perspective on Dimensionality Reduction

Deng Cai, Xiaofei He, Jiawei Han

University of Illinois at Urbana Champaign
Zhejiang University

SDM'09 Tutorial, April, 2009

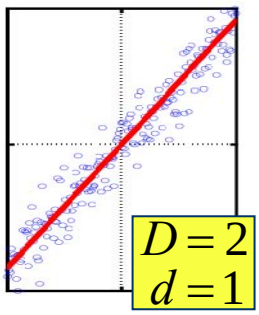
Dimensionality reduction

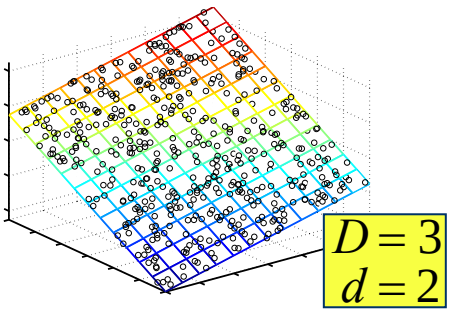
- Question

  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
  - Visualization of large networks

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

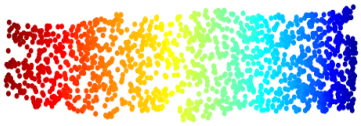




Does the data mostly lie in a subspace?

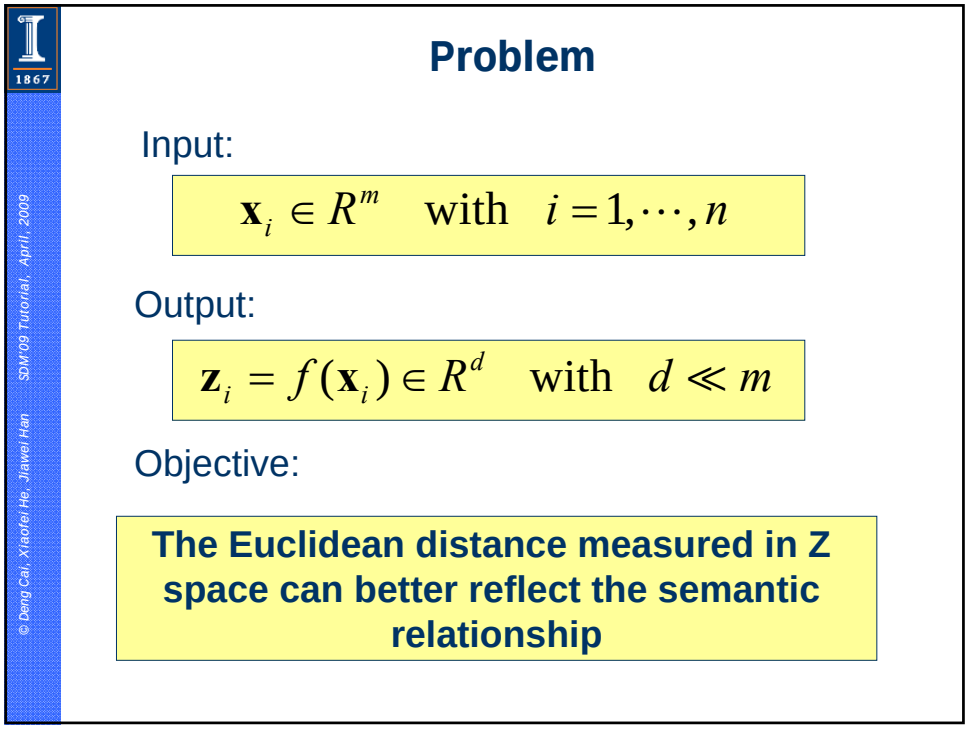
If so, what is its dimensionality?

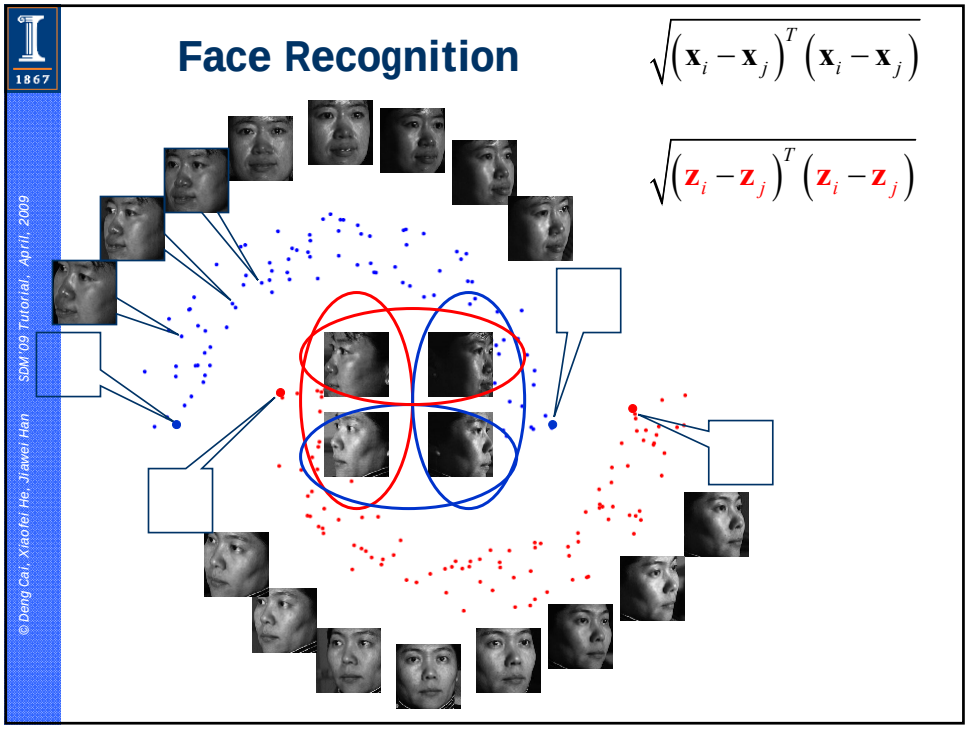
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

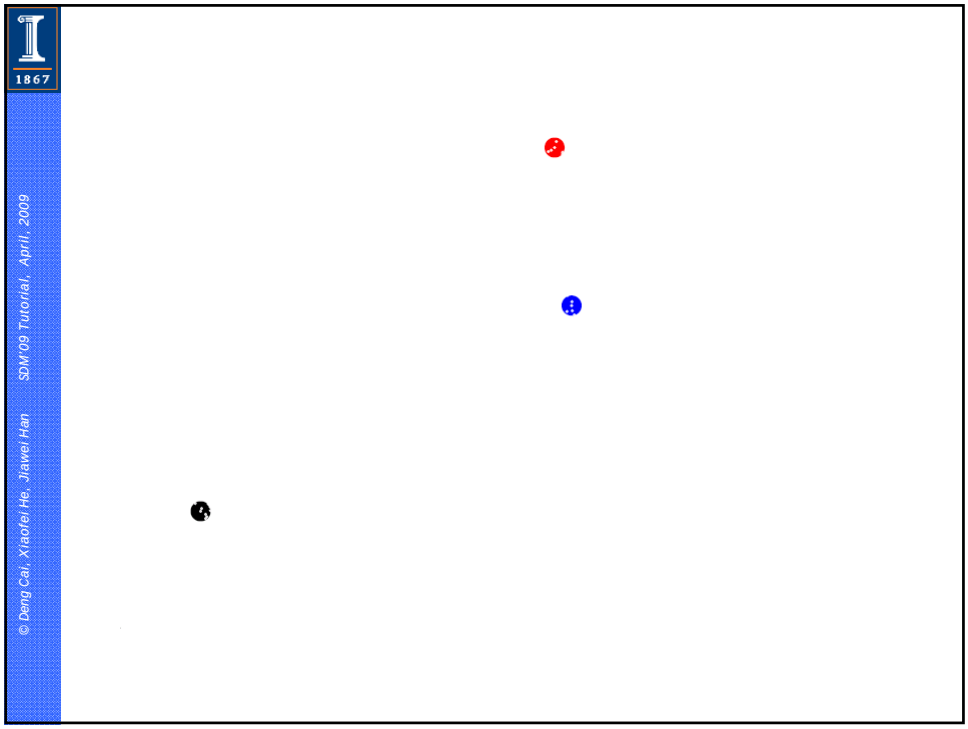


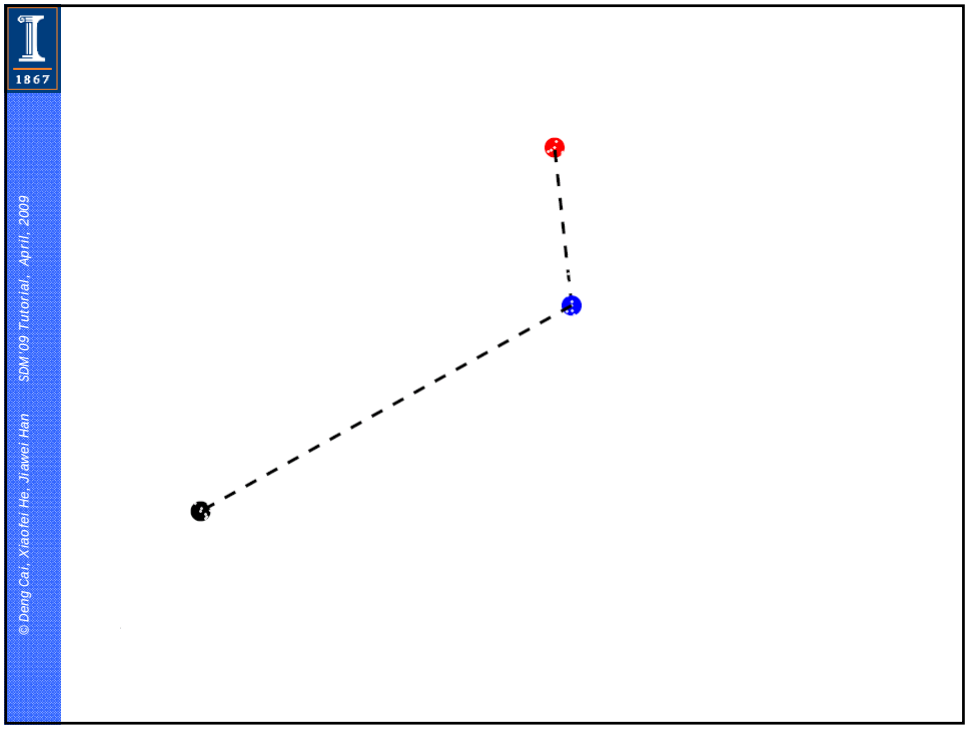
If the data are nonlinear distributed, how can we handle this?

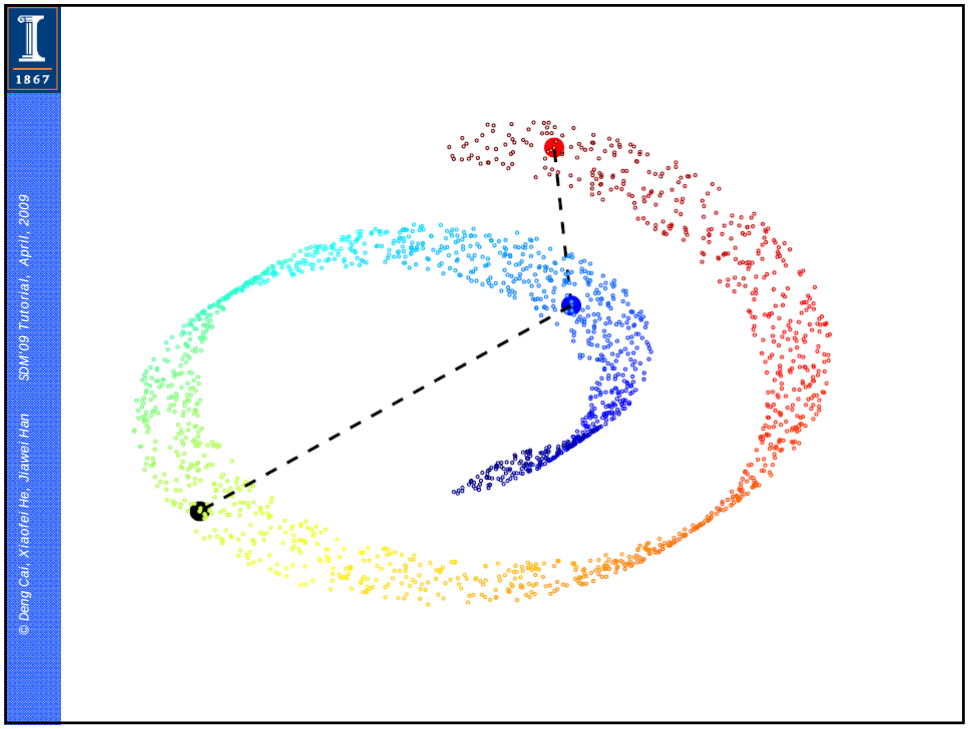
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


Problem
Input:
$\mathbf{x}_i \in R^m \quad \text{with} \quad i = 1, \cdots, n$
Output:
$\mathbf{z}_i = f(\mathbf{x}_i) \in R^d \quad \text{with} \quad d \ll m$
Objective:
The Euclidean distance measured in Z space can better reflect the semantic relationship
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


Face Recognition
$\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$
$\sqrt{(\mathbf{z}_i - \mathbf{z}_j)^T (\mathbf{z}_i - \mathbf{z}_j)}$
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

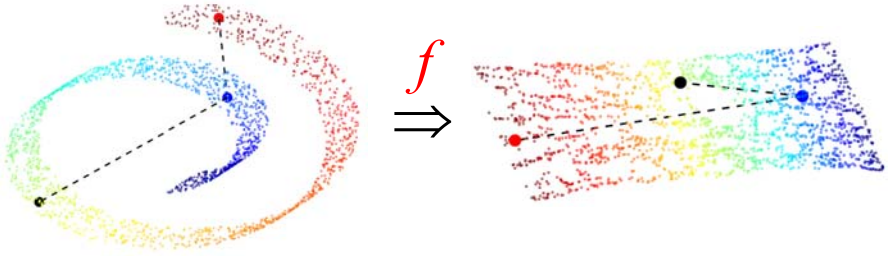

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Manifold-based Dimensionality Reduction



- Given high dimensional data sampled from a low dimensional manifold, how to compute a faithful embedding?
- How to find the projective function f ?
- How to efficiently find the projective function f ?

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

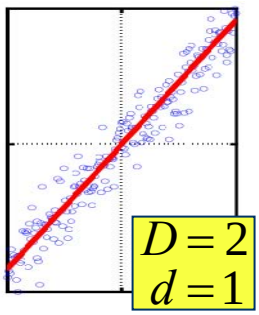
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Outline

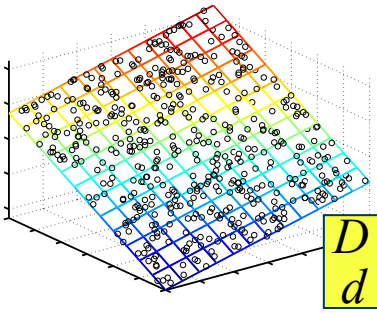
- Part 1 – Traditional linear methods
- Part 2 – Graph-based methods
- Part 3 – Linear and kernel extension
- Part 4 – Spectral regression for efficient computation
- Part 5 – Sparse subspace learning for feature selection

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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: $\sum_i \mathbf{x}_i = \mathbf{0}$
- Project into subspace:

$$\mathbf{y}_i = A^T \mathbf{x}_i \quad \text{with} \quad A^T A = I$$

- Maximize projected variance:

$$\text{var}(\mathbf{y}) = \frac{1}{n} \sum_i \| A^T \mathbf{x}_i \|^2$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Matrix eigen-decomposition

- Covariance matrix

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

- Spectral decomposition

$$C = \sum_{j=1}^{d} \lambda_j \mathbf{a}_j \mathbf{a}_j^{\mathrm{T}} \quad \text{with} \quad \lambda_1 \geq \cdots \geq \lambda_d \geq 0$$

- Maximum variance projection

$$A = [\mathbf{a}_1 \cdots \mathbf{a}_d]$$

Projects into subspace spanned by top *d* eigenvectors.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Interpreting PCA

- Eigenvectors:

  principal axes of maximum variance subspace.

- Eigenvalues:

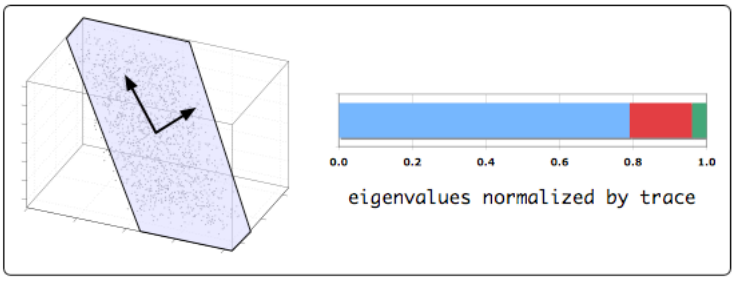
  projected variance of inputs along principle axes.

- Estimated dimensionality:

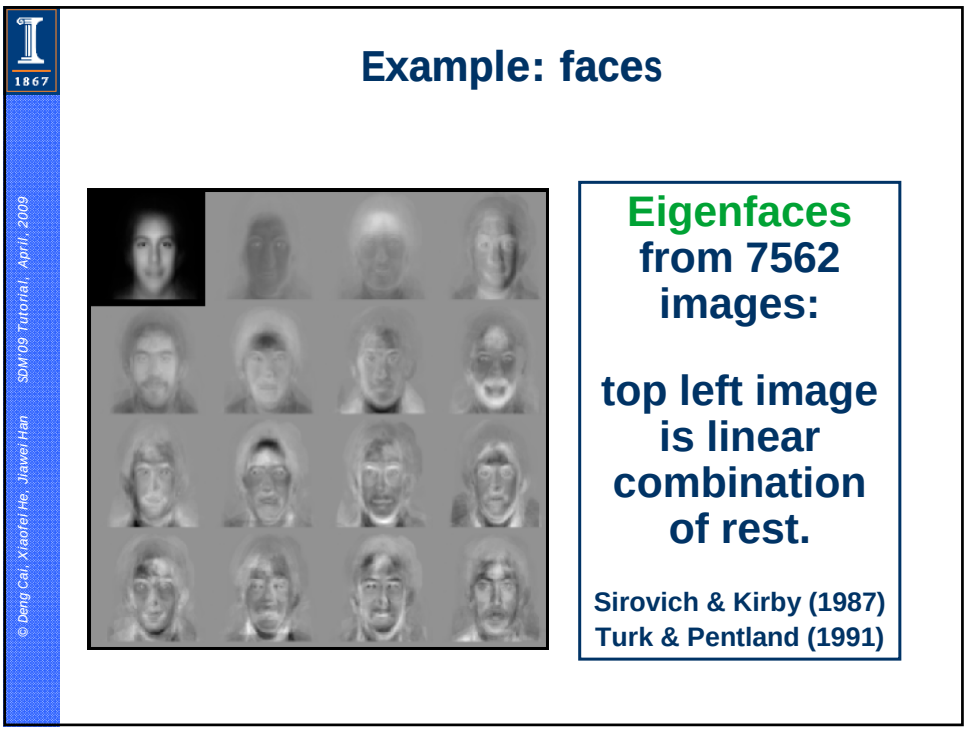
  number of significant (nonnegative) eigenvalues.

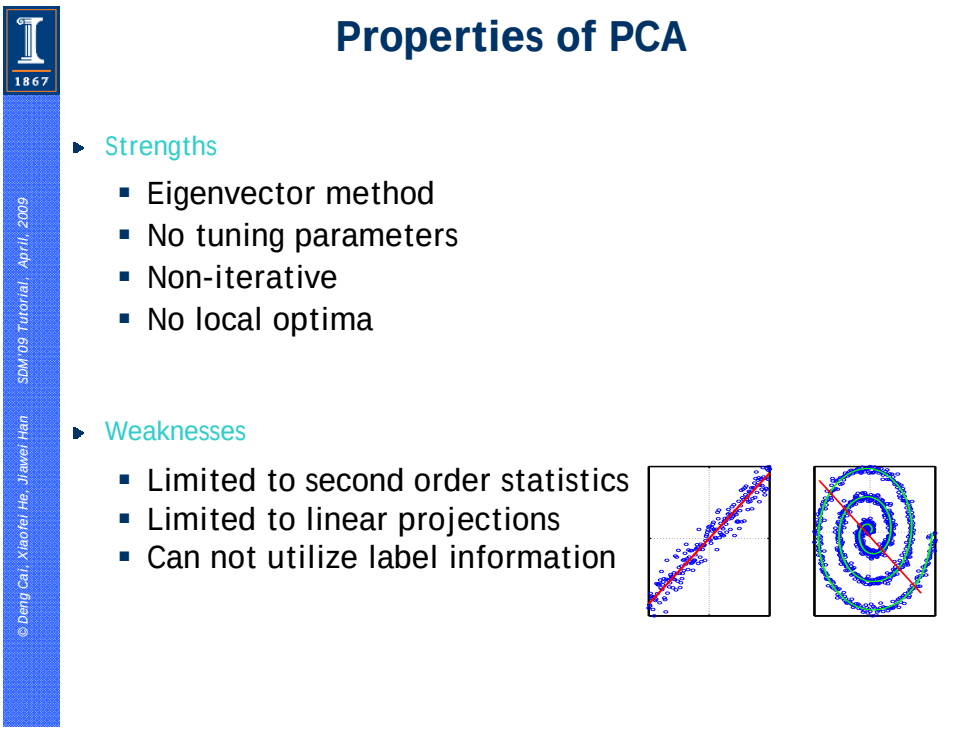
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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)
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


Properties of PCA
Strengths
Eigenvector method
No tuning parameters
Non-iterative
No local optima
Weaknesses
Limited to second order statistics
Limited to linear projections
Can not utilize label information
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

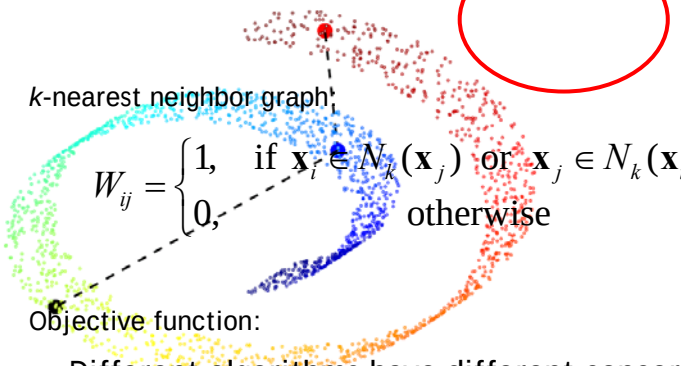
Outline

- Part 1 – Traditional linear methods
- Part 2 – Graph-based methods
- Part 3 – Linear and kernel extension
- Part 4 – Spectral regression for efficient computation
- Part 5 – Sparse subspace learning for feature selection

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

A Good Function

- If $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other, we hope $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ preserve the local structure (distance, similarity …)



- *k*-nearest neighbor graph:

$$W_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases}$$

- Objective function:
  - Different algorithms have different concerns

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Graph-based method #1

Isometric mapping of

data manifolds

(ISOMAP)

(Tenenbaum, de Silva, & Langford, 2000)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

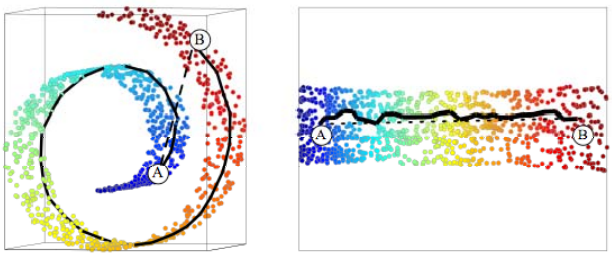
Isomap

- Key idea:

  Preserve geodesic distances as estimated along submanifold.

- Algorithm in a nutshell:

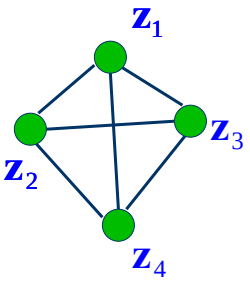
  Use geodesic instead of Euclidean distances in Multi-Dimensional Scaling (MDS).



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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 $\mathbf{z}_i$ such that $\|\mathbf{z}_i - \mathbf{z}_j\| \approx \Delta_{ij}$.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Metric Multidimensional Scaling

- Lemma

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

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

- Optimization

  Preserve dot products (proxy for distances).
  Choose vectors y_i to minimize:

$$\text{err}(\mathbf{z}) = \sum_{ij}\left(G_{ij} - \mathbf{z}_i^T\mathbf{z}_j\right)^2$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Matrix diagonalization

- Gram matrix "matching"

$$\text{err}(\mathbf{z}) = \sum_{ij} \left(G_{ij} - \mathbf{z}_i^T \mathbf{z}_j \right)^2$$

- Spectral decomposition

$$G = \sum_{j=1}^{d} \lambda_j \mathbf{a}_j \mathbf{a}_j^{\mathrm{T}} \quad \text{with} \quad \lambda_1 \geq \cdots \geq \lambda_d \geq 0$$

- Optimal approximation

$$Z = [\mathbf{z}_1 \cdots \mathbf{z}_n] = \left[\sqrt{\lambda_1}\mathbf{a}_1 \cdots \sqrt{\lambda_d}\mathbf{a}_d \right]^T$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

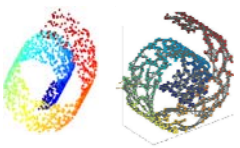
Step 1. Build adjacency graph.

- 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.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

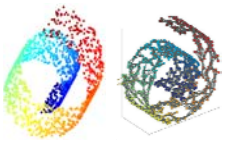
Building the graph

- 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.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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 $O(n^2 \log n + n^2 k)$.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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 $O(n^2d)$.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Summary

- Algorithm

  1) k nearest neighbors
  2) shortest paths through graph
  3) MDS on geodesic distances

- Impact

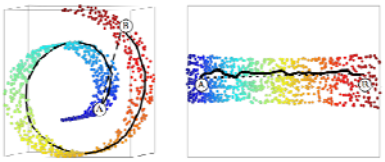
  Much simpler than neural nets, self-organizing maps, etc. Does it work?

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Examples

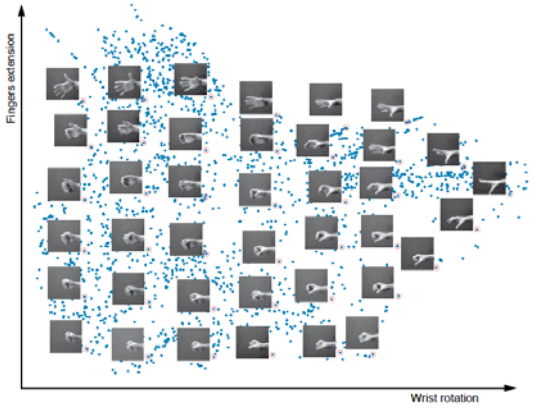
- Swiss roll

$n = 1024$
$k = 12$



- Wrist images

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

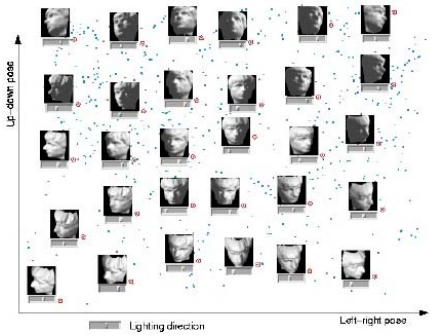


© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Examples

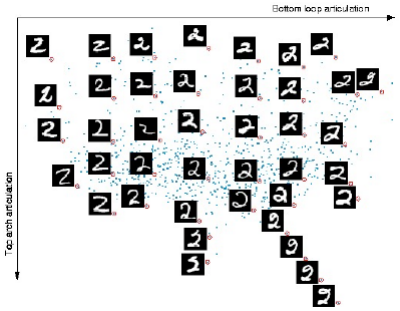
- Face images

$n = 698$
$k = 6$



- Digit images

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



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Graph-based method #2

Locally linear embedding

(LLE)

(Roweis & Saul, 2000)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

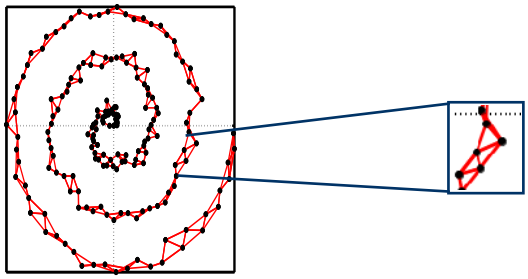
Locally linear embedding

- 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.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Step 2. Compute weights.

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



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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear reconstructions

- Local linearity

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

- Reconstruction errors

  Least squared errors should be small:

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Least squares fits

- Local reconstructions

  Choose weights to minimize:

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

- Constraints

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

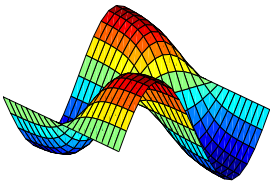
$$\sum_j W_{ij} = 1$$

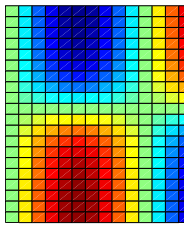
- Local invariance

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Step 3. "Linearization"

- Low dimensional representation

  Map inputs to outputs: $\mathbf{x}_i \in R^m$ to $\mathbf{z}_i \in R^d$

- Minimize reconstruction errors.

  Optimize outputs for fixed weights:

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

- Constraints

  Center outputs on origin: $\sum_i \mathbf{z}_i = \mathbf{0}$

  Impose unit covariance matrix: $\frac{1}{n} \sum_i \mathbf{z}_i \mathbf{z}_i^T = I.$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Sparse eigen-problem

- Quadratic form

$$\Psi(\mathbf{z}) = \sum_{ij} M_{ij} \left(\mathbf{z}_i^T \mathbf{z}_j \right) \text{ with } M = (I - W)^T (I - W)$$

- Rayleigh-Ritz quotient

Optimal embedding given by bottom d+1 eigenvectors.

- Solution

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Summary of LLE

- Three steps

1. Compute k-nearest neighbors.
2. Compute weights W_{ij}.
3. Compute outputs $\mathbf{z}_i$.

- Optimizations

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

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

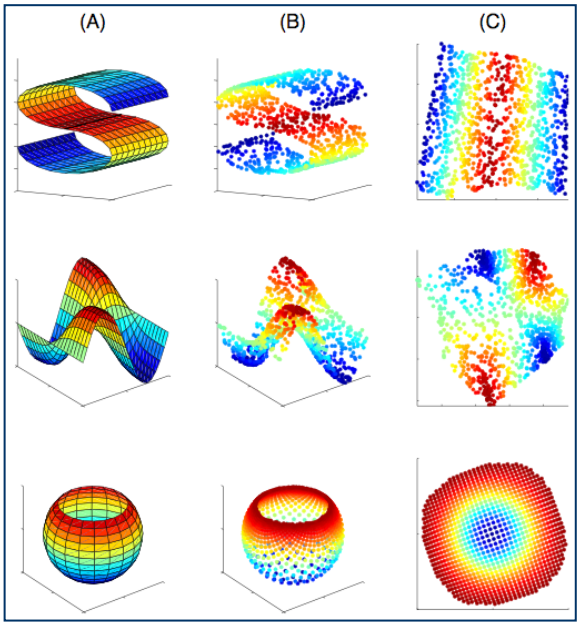
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Surfaces

n=1000
inputs

k=8
nearest
neighbors

m=3
d=2
dimensions



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

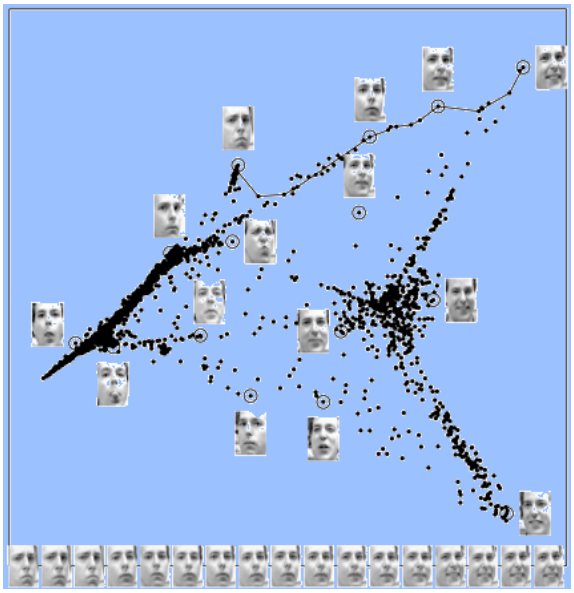
Pose and expression

n=1965
images

k=12
nearest
neighbors

m=560
pixels

d=2
(shown)



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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

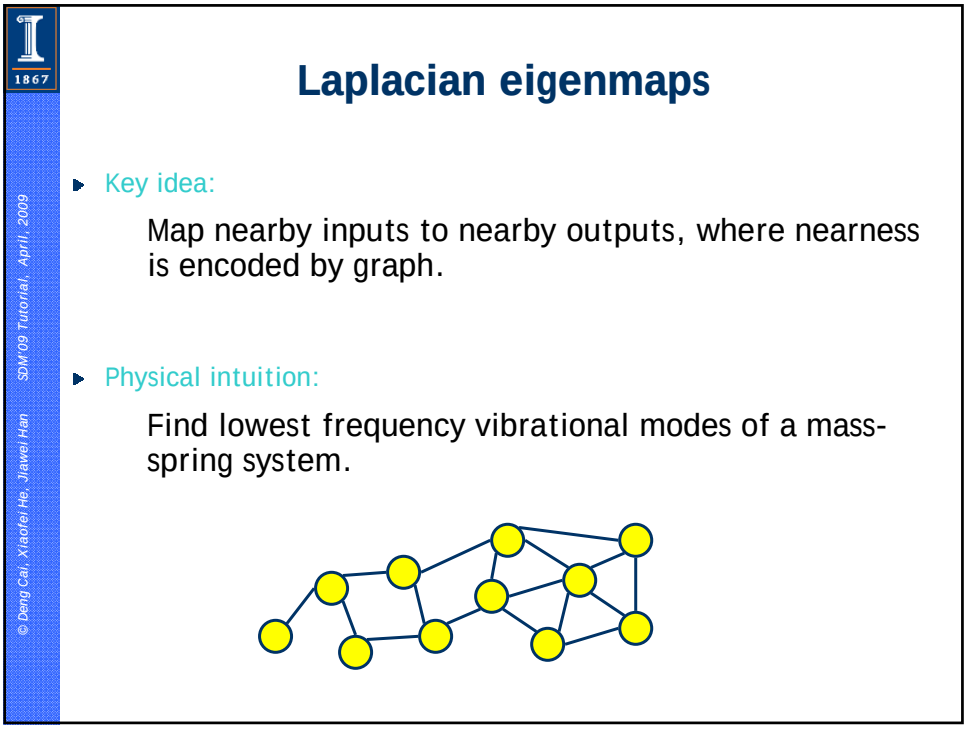
Graph-based method #3

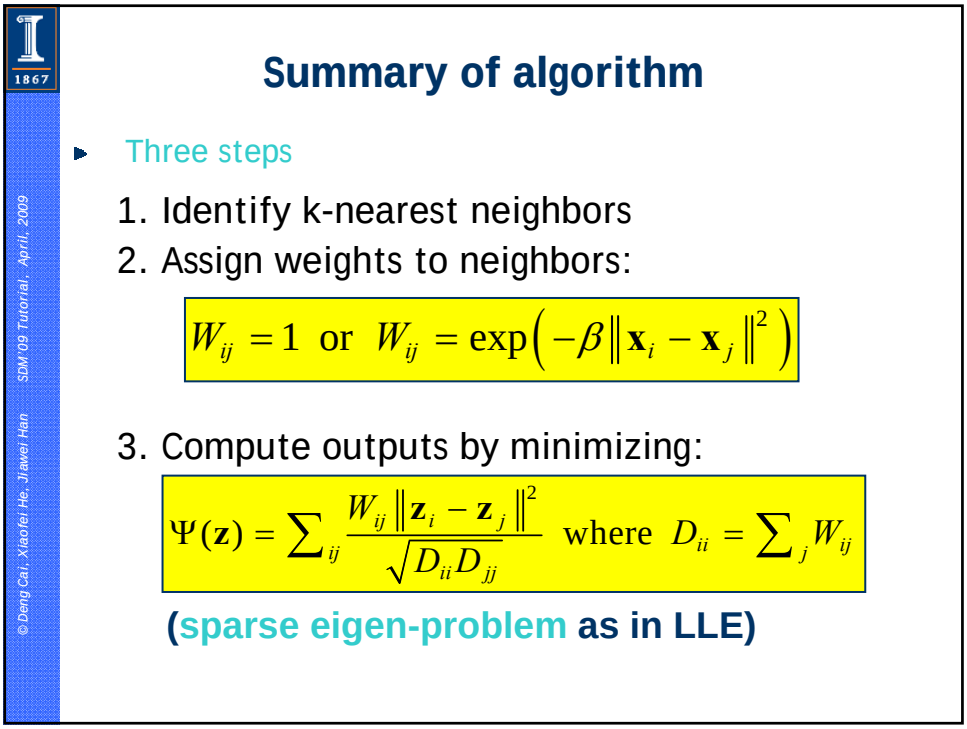
Laplacian Eigenmap

(LE)

(Belkin & Niyogi, 2001)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


1867
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009
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.


1867
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009
Summary of algorithm
Three steps
1. Identify k-nearest neighbors
2. Assign weights to neighbors:
$W_{ij} = 1$ or $W_{ij} = \exp\left(-\beta \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2\right)$
3. Compute outputs by minimizing:
$\Psi(\mathbf{z}) = \sum_{ij} \frac{W_{ij} \left\| \mathbf{z}_i - \mathbf{z}_j \right\|^2}{\sqrt{D_{ii} D_{jj}}}$ where $D_{ii} = \sum_j W_{ij}$
(sparse eigen-problem as in LLE)

Analysis on Manifolds

- Laplacian in 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

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Spectral graph theory

- Manifolds and graphs

  Weighted graph is discretized representation of manifold.

- Laplacian operators

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

$$\begin{aligned} \int_M \|\nabla f\|^2 &= \int_M f \Delta f \quad \text{(manifold)} \\ \sum_{ij} W_{ij} \left(f_i - f_j\right)^2 &= f^{\mathrm{T}} L f \quad \text{(graph)} \end{aligned}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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

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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

LLE (LE) 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

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

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.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

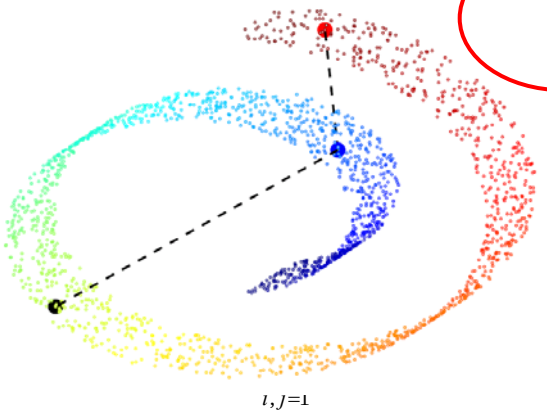
Outline

- Part 1 – Traditional linear methods
- Part 2 – Graph-based methods
- Part 3 – Linear and kernel extension
- Part 4 – Spectral regression for efficient computation
- Part 5 – Sparse subspace learning for feature selection

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

A Good Function

- If $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other, we hope $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ are also close.



$j \in N_k(\mathbf{x}_i)$

W_{ij}

$i,j=1$

- with certain constraint

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

No function is learned

$$\mathbf{x} \to f(\mathbf{x}) = y \qquad X \to f(X) = \mathbf{y}$$

$$\phi(\mathbf{y}) = \sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij}$$

$$= 2\sum_{i=1}^{n} y_i D_{ii} y_i - 2\sum_{i,j=1}^{n} y_i W_{ij} y_j$$

$$= 2\mathbf{y}^T (D - W)\mathbf{y} = 2\mathbf{y}^T L\mathbf{y}$$

$$\text{Constraint}: \quad \mathbf{y}^T D\mathbf{y} = 1$$

$$\min \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} = \max \frac{\mathbf{y}^T (D - L)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} = \max \frac{\mathbf{y}^T W\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

A Good Linear Function

- Consider a linear function f: $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$

$$\begin{aligned}\phi(f) &= \sum_{i,j=1}^{n}(f(\mathbf{x}_i)-f(\mathbf{x}_j))^2 W_{ij} \\ &= \sum_{i,j=1}^{n}(\mathbf{a}^T\mathbf{x}_i-\mathbf{a}^T\mathbf{x}_j)^2 W_{ij} \\ &= 2\sum_{i=1}^{n}\mathbf{a}^T\mathbf{x}_i D_{ii}\mathbf{x}_i^T\mathbf{a} - 2\sum_{i,j=1}^{n}\mathbf{a}^T\mathbf{x}_i W_{ij}\mathbf{x}_j^T\mathbf{a} \\ &= 2\mathbf{a}^T X(D-W)X^T\mathbf{a} = 2\mathbf{a}^T XLX^T\mathbf{a}\end{aligned}$$

$$\text{Constraint}: \quad \mathbf{a}^T XDX^T\mathbf{a} = 1$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Locality Preserving Projections (LPP)

[He & Niyogi, 03]

A good linear function $\mathbf{a}$:

$$\min \mathbf{a}^T XLX^T\mathbf{a}, \quad \text{s.t. } \mathbf{a}^T XDX^T\mathbf{a} = 1$$

$$\min \frac{\mathbf{a}^T XLX^T\mathbf{a}}{\mathbf{a}^T XDX^T\mathbf{a}} \qquad \max \frac{\mathbf{a}^T XWX^T\mathbf{a}}{\mathbf{a}^T XDX^T\mathbf{a}}$$

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}$$

- LPP is a linear version of Laplacian Eigenmap.
- The same trick can be applied on LLE and Isomap.
  - LLE → Neighborhood Preserving Embedding [He et. al 05]
  - Isomap → Isometric Projection [Cai et. al 06]

A Good Non-Linear Function

$$\max \frac{\mathbf{y}^T W \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} \rightarrow \max \frac{\mathbf{a}^T X W X^T \mathbf{a}}{\mathbf{a}^T X D X^T \mathbf{a}} \rightarrow \max \frac{\boldsymbol{\alpha}^T K W K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T K D K \boldsymbol{\alpha}}$$

- $\mathbf{x}$ -
  - Gaussian kernel $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2)$
  - Polynomial kernel $\mathcal{K}(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^d$
  - Sigmoid kernel $\mathcal{K}(\mathbf{x}, \mathbf{y}) = tanh(\mathbf{x}^T \mathbf{y} + \alpha)$
- A li = $X^T \mathbf{a}$
- A non-linear function f : $y = f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x})$

$$\mathbf{y} = f(X) = K\boldsymbol{\alpha}$$

Classical Representer Theorem

Kernel LPP [He & Niyogi, 03], Kernel Eigenmap [Brand, 03]

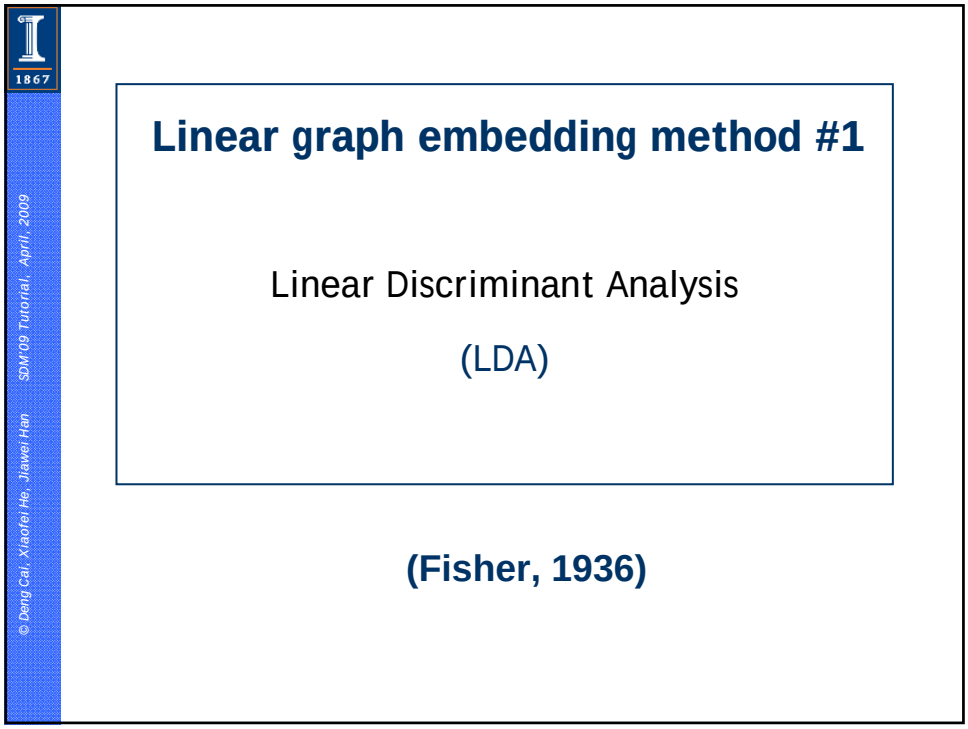
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

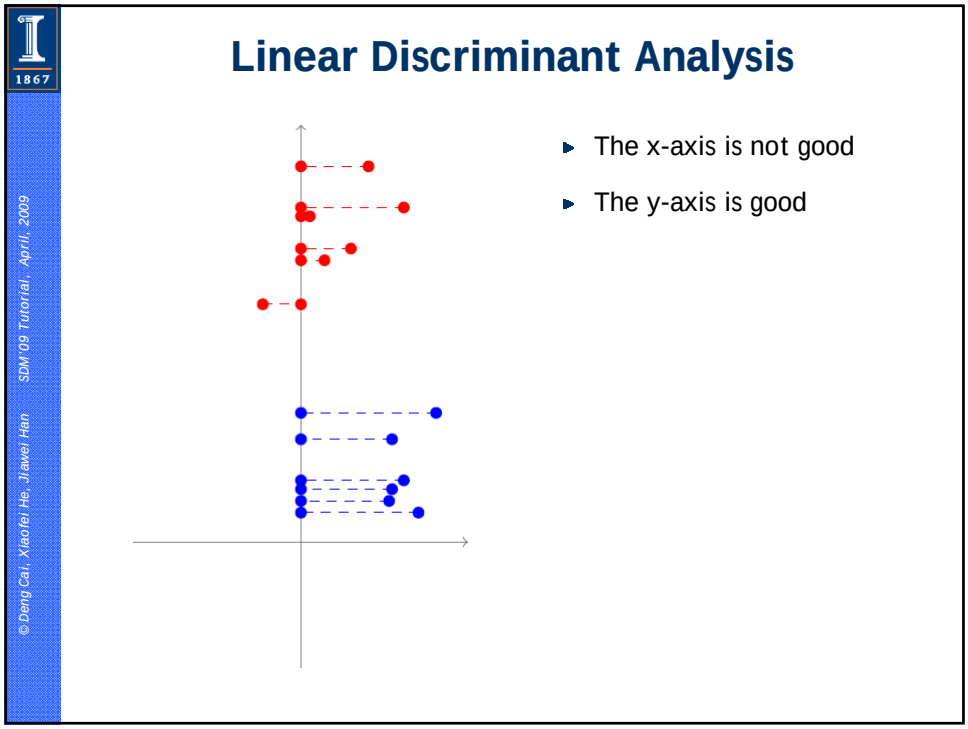
Generalized Eigenproblem

$$\max \frac{\mathbf{y}^T W \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} \quad \max \frac{\mathbf{a}^T X W X^T \mathbf{a}}{\mathbf{a}^T X D X^T \mathbf{a}} \quad \max \frac{\boldsymbol{\alpha}^T K W K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T K D K \boldsymbol{\alpha}}$$

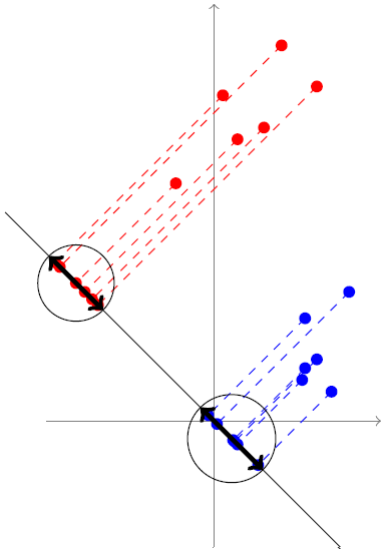
- *W* and *D* can be very flexible to characterize the statistical and geometric properties of the data.
  - A large family of algorithms.
  - Graph embedding
  - (Linear Graph Embedding)
  - (Kernel Graph Embedding)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009


1867
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009
Linear graph embedding method #1
Linear Discriminant Analysis
(LDA)
(Fisher, 1936)


1867
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009
Linear Discriminant Analysis
The x-axis is not good
The y-axis is good

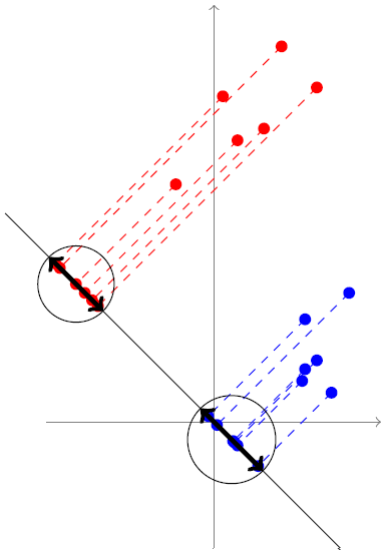
Linear Discriminant Analysis



- The x-axis is not good
- The y-axis is good
- An better direction
- Which direction is the best?
- Data points of different classes are far from each other
- Data points of the same class are close to each other

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear Discriminant Analysis



- Which direction is the best?

$$\mathbf{x} \rightarrow y = \mathbf{a}^T \mathbf{x}$$

$$D_b = \left(\overline{y}_1 - \overline{y}_2\right)^2$$

$$D_w = \sum_{c=1}^{2} \sum_{y \in C_c} \left(y - \overline{y}_c\right)^2$$

$$\mathbf{a}^* = \arg\max \frac{D_b}{D_w}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear Discriminant Analysis

$$\mathbf{a}^* = \arg\max \frac{D_b}{D_w} = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}}$$

$$D_b = \sum_{c=1}^{k} n_c \left(\bar{y}_c - \bar{y}\right)^2 = \sum_{c=1}^{k} n_c \left(\mathbf{a}^T \bar{\mathbf{x}}_c - \mathbf{a}^T \bar{\mathbf{x}}\right)^2$$

$$= \mathbf{a}^T \left(\sum_{c=1}^{k} n_c \left(\bar{\mathbf{x}}_c - \bar{\mathbf{x}}\right)\left(\bar{\mathbf{x}}_c - \bar{\mathbf{x}}\right)^T \right) \mathbf{a}$$

Between class scatter matrix S_b

$$D_w = \sum_{c=1}^{2} \sum_{y \in C_c} \left(y - \bar{y}_c\right)^2 = \sum_{c=1}^{k} \sum_{\mathbf{x} \in C_c} \left(\mathbf{a}^T \mathbf{x} - \mathbf{a}^T \bar{\mathbf{x}}_c\right)^2$$

$$= \mathbf{a}^T \left(\sum_{c=1}^{k} \sum_{\mathbf{x} \in C_c} \left(\mathbf{x} - \bar{\mathbf{x}}_c\right)\left(\mathbf{x} - \bar{\mathbf{x}}_c\right)^T \right) \mathbf{a}$$

Within class scatter matrix S_w

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear Discriminant Analysis

$$\mathbf{a}^* = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}}$$

$$= \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T \left(S_w + S_b\right) \mathbf{a}} = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}}$$

$$S_t = S_b + S_w = \sum_{i=1}^{n} \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)^T$$

Total scatter matrix S_t

Covariance matrix if divided by n

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Graph Embedding Perspective of LDA

- Assume the data are centered. $\bar{\mathbf{x}} = \mathbf{0}$

$$X = \left[X^{(1)}, X^{(2)} \cdots, X^{(k)} \right] = \left[\mathbf{x}_1^{(1)} \cdots \mathbf{x}_{n_1}^{(1)}, \cdots, \mathbf{x}_1^{(k)} \cdots \mathbf{x}_{n_k}^{(k)} \right]$$

$$S_t = \sum_{i=1}^{n} \left(\mathbf{x}_i - \bar{\mathbf{x}} \right) \left(\mathbf{x}_i - \bar{\mathbf{x}} \right)^T = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T = XX^T$$

$$S_b = \sum_{c=1}^{k} n_c \left(\bar{\mathbf{x}}_c - \bar{\mathbf{x}} \right) \left(\bar{\mathbf{x}}_c - \bar{\mathbf{x}} \right)^T = \sum_{c=1}^{k} n_c \bar{\mathbf{x}}_c \bar{\mathbf{x}}_c^T$$

$$= \sum_{c=1}^{k} n_c \left(\frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{x}_i^{(c)} \right) \left(\frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{x}_i^{(c)} \right)^T$$

$$= \sum_{c=1}^{k} \frac{1}{n_c} \left(\sum_{i=1}^{n_c} \mathbf{x}_i^{(c)} \right) \left(\sum_{i=1}^{n_c} \mathbf{x}_i^{(c)} \right)^T = \sum_{c=1}^{k} X^{(c)} W^{(c)} \left(X^{(c)} \right)^T$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Graph Embedding Perspective of LDA

$$W^{(c)} = \begin{bmatrix} 1/n_c & \cdots & 1/n_c \\ \vdots & \ddots & \vdots \\ 1/n_c & \cdots & 1/n_c \end{bmatrix} \qquad W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(k)} \end{bmatrix}$$

$$W_{ij} = \begin{cases} 1/n_c, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ both belong to the } c\text{-th class} \\ 0, & \text{otherwise} \end{cases}$$

$$S_b = \sum_{c=1}^{k} X^{(c)} W^{(c)} \left(X^{(c)} \right)^T = XWX^T$$

$$S_t = \sum_{i=1}^{n} \left(\mathbf{x}_i - \bar{\mathbf{x}} \right) \left(\mathbf{x}_i - \bar{\mathbf{x}} \right)^T = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T = XX^T$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Graph Embedding Perspective of LDA

$$\mathbf{a}^* = \arg\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}} = \arg\max \frac{\mathbf{a}^T X W X^T \mathbf{a}}{\mathbf{a}^T X X^T \mathbf{a}}$$

$$W_{ij} = \begin{cases} 1/n_c, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ both belong to the } c\text{-th class} \\ 0, & \text{otherwise} \end{cases}$$

$$D_{ii} = \sum_{j=1}^{m} W_{ij} = 1 \qquad D = I$$

Linear Discriminant Analysis is a linear graph embedding approach with a specifically designed supervised graph.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear graph embedding method #2

Semi-supervised Discriminant Analysis

(SDA)

(Cai et. al, 2007)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Utilizing unlabeled data in LDA?

- *n* labeled examples, *N-n* unlabeled examples

  X : labeled examples Z : all the examples

  - LDA only uses the labeled examples

$$\mathbf{a}^* = \max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}} = \arg\max \frac{\mathbf{a}^T X W_l X^T \mathbf{a}}{\mathbf{a}^T X X^T \mathbf{a}}$$

- *Can we utilize the unlabeled examples for better estimation of* **a* ?**

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Utilizing unlabeled data in LDA

- Basic idea: combine LDA and LPP.
  - Construct a k-nearest neighbor graph *W*

$$J(\mathbf{a}) = \frac{1}{2}\sum_{i,j=1}^{N}(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j)^2 W_{ij} = \mathbf{a}^T Z L Z^T \mathbf{a}$$

$$\max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a} + \alpha J(\mathbf{a})} \Rightarrow \max \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a} + \alpha \mathbf{a} X L X^T \mathbf{a}}$$

$$\Rightarrow \max \frac{\mathbf{a}^T X W_l X^T \mathbf{a}}{\mathbf{a}^T X X^T \mathbf{a} + \alpha \mathbf{a} Z L Z^T \mathbf{a}}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Utilizing unlabeled data in LDA

$$\max \frac{\mathbf{a}^T X W_l X^T \mathbf{a}}{\mathbf{a}^T X X^T \mathbf{a} + \alpha \mathbf{a} Z L Z^T \mathbf{a}} \quad \Rightarrow \max \frac{\mathbf{a}^T Z W Z^T \mathbf{a}}{\mathbf{a}^T Z (I + \alpha L) Z^T \mathbf{a}}$$

$$\tilde{W} = \begin{bmatrix} W_l & 0 \\ 0 & 0 \end{bmatrix}_{N \times N} \qquad \tilde{I} = \begin{bmatrix} I_l & 0 \\ 0 & 0 \end{bmatrix}_{N \times N}$$

$$X W_l X^T = \begin{bmatrix} X & X_u \end{bmatrix} \begin{bmatrix} W_l & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X^T \\ X_u^T \end{bmatrix} = Z \tilde{W} Z^T$$

$$X X^T = \begin{bmatrix} X & X_u \end{bmatrix} \begin{bmatrix} I_l & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X^T \\ X_u^T \end{bmatrix} = Z \tilde{I} Z^T$$

- Semi-supervised Discriminant Analysis (SDA)
  - Another linear graph embedding algorithm

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

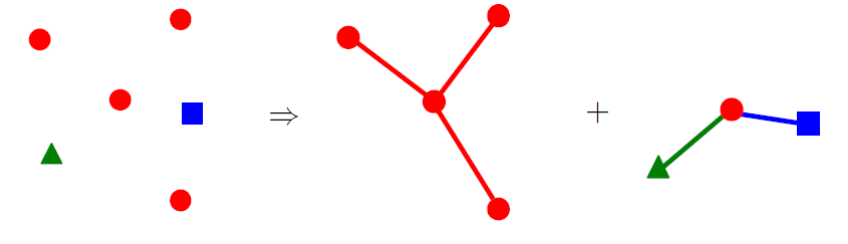
Linear graph embedding method #3

Marginal Fisher Analysis (MFA)

Local discriminant embedding (LDE)

[Yan et. al, 2005], [Chen et. al, 2005]

Exploiting the discriminant structure in local



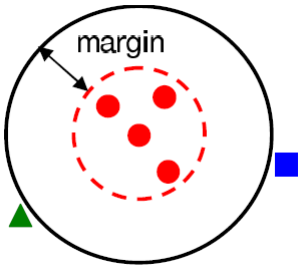
kNN graph

within-class neighbor graph

between-class neighbor graph

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Exploiting the discriminant structure in local



Maximizing the distances in between-class graph at the same time minimizing the distances in within-class graph, the margin between different classes is maximized.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Within-class and between-class neighbor graph

$l(\mathbf{x}_i)$: class label of $\mathbf{x}_i$

$N(\mathbf{x}_i)$: k nearest neighbors of $\mathbf{x}_i$

$$N_b(\mathbf{x}_i) = \left\{ \mathbf{x}_j^i \mid \mathbf{x}_j^i \in N(\mathbf{x}_i), l(\mathbf{x}_i) \neq l(\mathbf{x}_j^i) \right\}$$

$$N_w(\mathbf{x}_i) = \left\{ \mathbf{x}_j^i \mid \mathbf{x}_j^i \in N(\mathbf{x}_i), l(\mathbf{x}_i) = l(\mathbf{x}_j^i) \right\}$$

between-class neighbor graph

$$W_{b,ij} = \begin{cases} 1 & \mathbf{x}_j \in N_b(\mathbf{x}_i) \text{ or } \mathbf{x}_i \in N_b(\mathbf{x}_j) \\ 0 & \text{otherwise.} \end{cases}$$

within-class neighbor graph

$$W_{w,ij} = \begin{cases} 1 & \mathbf{x}_j \in N_w(\mathbf{x}_i) \text{ or } \mathbf{x}_i \in N_w(\mathbf{x}_j) \\ 0 & \text{otherwise.} \end{cases}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Within-class and between-class neighbor graph

$$\mathbf{a}^* = \arg\max \sum_{i,j=1}^{n} (\mathbf{a}^T \mathbf{x}_i - \mathbf{a}^T \mathbf{x}_j)^2 W_{b,ij} = \arg\max \mathbf{a}^T X L_b X^T \mathbf{a}$$

at the same time

$$\mathbf{a}^* = \arg\min \sum_{i,j=1}^{n} (\mathbf{a}^T \mathbf{x}_i - \mathbf{a}^T \mathbf{x}_j)^2 W_{w,ij} = \arg\min \mathbf{a}^T X L_w X^T \mathbf{a}$$

$$\mathbf{a}^* = \arg\max \frac{\mathbf{a}^T X L_b X^T \mathbf{a}}{\mathbf{a}^T X L_w X^T \mathbf{a}}$$

- Marginal Fisher Analysis (MFA)
  - Another linear graph embedding algorithm

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Linear Graph Embedding Algorithms

- Locality Preserving Projection [He & Niyogi, 2003]
- Linear Discriminant Analysis [Fisher, 1936]
- Neighborhood Preserving Embedding [He et. al, 2005]
- Marginal Fisher Analysis [Yan et. al, 2005]
- Local discriminant embedding [Chen et. al, 2005]
- Augmented Relation Embedding [Lin et. al, 2005]
- Isometric Projection [Cai et. al, 2006]
- Semantic Subspace Projection [Yu & Tian, 2006]
- Locally Sensitive Discriminant Analysis [Cai et. al, 2007]
- Semi-supervised Discriminant Analysis [Cai et. al, 2007]
- .....

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Generalized Eigen-problem for LGE

$$\max \frac{\mathbf{a}^T X W X^T \mathbf{a}}{\mathbf{a}^T X D X^T \mathbf{a}}$$

$$X W X^T \mathbf{a} = \lambda X D X^T \mathbf{a}$$

- Time: $O\left(mnt + t^3\right), \quad t = \min(m, n)$
- Memory: $O\left(mn\right)$
- Does there exist efficient way to solve the optimization problem?

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Generalized Eigen-problem for KGE

$$\max \frac{\boldsymbol{\alpha}^T KWK\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T KDK\boldsymbol{\alpha}}$$

$$KWK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha}$$

$$\frac{9}{2}n^3 + O\left(n^2 m\right) \quad \text{multiplication}$$

- Does there exist efficient way to solve the optimization problem?

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Outline

- Part 1 – Traditional linear methods
- Part 2 – Graph-based methods
- Part 3 – Linear and kernel extension
- Part 4 – Spectral regression for efficient computation
- Part 5 – Sparse subspace learning for feature selection

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Decomposing The Eigen-problem

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}$$

Theorem

Let $\mathbf{y}$ *be the eigenvector of eigen-problem* $W\mathbf{y} = \lambda D\mathbf{y}$ *with eigenvalue* λ. *If* $X^T\mathbf{a} = \mathbf{y}$, *then* $\mathbf{a}$ *is the eigenvector of eigen-problem*

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}$$

with the same eigenvalue λ.

Proof.

We have $W\mathbf{y} = \lambda D\mathbf{y}$, thus,

$$XWX^T\mathbf{a} = XW\mathbf{y} = X\lambda D\mathbf{y} = \lambda XD\mathbf{y} = \lambda XDX^T\mathbf{a}$$

□

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Two-step Approach

1. Solve the eigen-problem $W\mathbf{y} = \lambda D\mathbf{y}$ to get $\mathbf{y}$.
2. Find $\mathbf{a}$ which satisfies the linear equations $X^T\mathbf{a} = \mathbf{y}$
   - Least square fit $\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} \left(\mathbf{a}^T\mathbf{x}_i - y_i\right)^2 + \alpha \|\mathbf{a}\|^2$
   - When *n* < *m*, we can use regularization
     - Ridge regression

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Decomposing The Eigen-problem

$$KWK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha}$$

Theorem

Let $\mathbf{y}$ *be the eigenvector of eigen-problem* $W\mathbf{y} = \lambda D\mathbf{y}$ *with eigenvalue* λ*. If* $K\boldsymbol{\alpha} = \mathbf{y}$*, then* $\boldsymbol{\alpha}$ *is the eigenvector of eigen-problem*

$$KWK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha}$$

with the same eigenvalue λ*.*

Proof.

We have $W\mathbf{y} = \lambda D\mathbf{y}$, thus,

$$KWK\boldsymbol{\alpha} = KW\mathbf{y} = K\lambda D\mathbf{y} = \lambda KD\mathbf{y} = \lambda KDK\boldsymbol{\alpha}$$

□

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Two-step Approach: Spectral Regression

1. Solve the eigen-problem $W\mathbf{y} = \lambda D\mathbf{y}$ to get $\mathbf{y}$.
2. Find $\boldsymbol{\alpha}$ which satisfies the linear equations $K\boldsymbol{\alpha} = \mathbf{y}$
   - When K is singular $(K + \delta I)\boldsymbol{\alpha} = \mathbf{y}$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Advantages of The Two-step Approach

- Both *W* and *D* are sparse matrices and the top eigenvectors can be efficiently calculated (Lanczos algorithm).
  - If LDA graph is used, the eigen-problem becomes trivial
- There exist many efficient iterative algorithms (e.g., LSQR) that can efficiently handle very large scale least squares problems.
- The regularization techniques can be easily incorporated.
- Incremental algorithms

Spectral Analysis + Regression = Spectral Regression

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Step 1: Eigenvectors of W, D

$$W\mathbf{y} = \lambda D\mathbf{y}$$

General graph: Lanczos algorithm.

LDA graph:

$$W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(k)} \end{bmatrix} \qquad W^{(c)} = \begin{bmatrix} 1/n_c & \cdots & 1/n_c \\ \vdots & \ddots & \vdots \\ 1/n_c & \cdots & 1/n_c \end{bmatrix}$$

- There are exactly *c* eigenvectors of *W* with the same eigenvalue 1. These eigenvectors are:

$$\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} m_i}, \underbrace{1, \cdots, 1}_{m_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{c} m_i}]^T \quad k = 1, \cdots, c$$

Step 2: Ridge Regression

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \alpha \|\mathbf{a}\|^2 \right)$$

- The closed form solution:

$$\mathbf{a} = \left(XX^T + \alpha I \right)^{-1} X\mathbf{y}$$

- When $\alpha > 0$, $\mathbf{a}$ will not satisfy the linear equations system and will not be the eigenvector of the eigen-problem:

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

SR ? Generalized Eigenvector Solution

Theorem

*Let **y** be the eigenvector of W, if **y** is in the space spanned by row vectors of X, the corresponding projective function **a** calculated in SR will be the eigenvector of the corresponding eigen-problem as α deceases to zero.*

Proof.

Please see our paper for a complete proof. □

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

SR ?= Generalized Eigenvector Solution

- For high dimensional data, we will often have $n < m$. In this case the sample vectors are usually linearly independent.

Theorem

If the sample vectors are linearly independent, i.e., $rank(X) = m$, all the projective functions calculated by SR are the eigenvectors of eigen-problem $XWX^T\mathbf{a} = \lambda XX^T\mathbf{a}$ as α deceases to zero. Thus, the solutions of SR are identical to the solutions given by directly solving the generalized eigen-problem.

Proof.

Please see our paper for a complete proof. □

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

KSR ?= Generalized Eigenvector Solution

$$(K + \delta I)\boldsymbol{\alpha} = \mathbf{y}$$

Lemma

Suppose that $\mathbf{x}_1, \cdots, \mathbf{x}_m$ are distinct points, and $\sigma \neq 0$. The matrix K given by

$$K_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$$

has full rank.

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Compute the Solution using Cholesky Decomposition

$$(K+\delta I)\boldsymbol{\alpha}=\mathbf{y} \qquad \boldsymbol{\alpha}=(K+\delta I)^{-1}\mathbf{y} \qquad \left(\frac{5}{6}n^3+n^2\right) \text{ multiplication}$$

Theorem

If K is positive definite, then K can be factored uniquely in the form $K=R^TR$, where R is upper triangular with positive diagonal elements.

$$(K+\delta I)\boldsymbol{\alpha}=\mathbf{y} \Rightarrow R^TR\boldsymbol{\alpha}=\mathbf{y}$$
$$R^T\mathbf{b}=\mathbf{y} \qquad \left(\frac{1}{6}n^3+n^2\right) \text{ multiplication}$$
$$R\boldsymbol{\alpha}=\mathbf{b}$$

Traditional approach : $\frac{9}{2}n^3+O(n^2m)$ multiplication

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Incremental Computation

$$R_m^TR_m = K_m = \begin{pmatrix} K_{m-1} & \mathbf{k}_{1m} \\ \mathbf{k}_{1m}^T & k_{mm} \end{pmatrix} = \begin{pmatrix} R_{m-1}^T & \mathbf{0} \\ \mathbf{r}_{1m}^T & r_{mm} \end{pmatrix} \begin{pmatrix} R_{m-1} & \mathbf{r}_{1m} \\ \mathbf{0}^T & r_{mm} \end{pmatrix}$$

$$K_{m-1}=R_{m-1}^TR_{m-1}$$

$$\mathbf{k}_{1m}=R_{m-1}^T\mathbf{r}_{1m}$$

$$k_{mm}=r_{mm}^2$$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Computational Complexity of LGE

- Dense Eigen-problem
  - Time
    - Supervised (LDA)

      $O\left(mnt + t^3\right)$
    - Unsupervised (LPP)

      $O\left(n^2 s + n^2 \log n + mnt + t^3\right)$
  - Memory

    $O\left(mn\right)$

    $t = \min(m, n) \qquad s \le m$
- Spectral Regression
  - Time
    - Supervised

      $O\left(ns\right)$
    - Unsupervised

      $O\left(n^2 s + n^2 \log n + ns\right)$
  - Memory

    $O\left(ns\right)$

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Computational Complexity of KGE

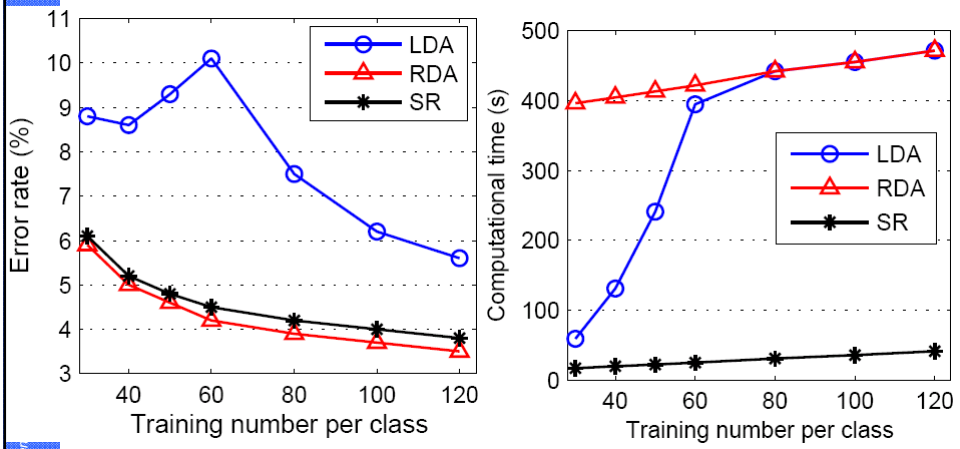
- Traditional Kernel Subspace Learning (Kernel Discriminant Analysis)

  $\frac{9}{2}n^3 + O\left(n^2 m\right)$ multiplication
- Kernel Spectral Regression

  $\frac{1}{6}n^3 + O(n^2 m)$ multiplication 27-times speedup
- Incremental Kernel Spectral Regression

  $\frac{1}{2}(\Delta n + c)n^2 + O(nm\Delta n)$ multiplication

Face Recognition Results



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

SR Provides Good Text Clustering Results

c	Accuracy (mean±std-dev%)					
	Baseline	LSI	PLSI	LPP	SR	NMF
2	97.7±7.3	93.4±14.2	91.7±13.0	**99.8±0.3**	**99.9±0.2**	99.2±4.7
3	88.4±18.0	86.1±20.0	82.8±18.3	**99.6±0.4**	**99.6±0.4**	95.7±11.0
4	85.7±18.9	79.2±21.2	75.4±19.3	**99.3±0.8**	**99.4±0.8**	92.4±11.9
5	82.4±17.8	76.8±22.3	72.5±18.0	**98.7±1.8**	**98.8±1.8**	92.2±10.5
6	79.0±17.5	72.0±19.4	68.2±16.8	**98.6±1.5**	**98.8±1.3**	88.0±12.6
7	74.5±16.5	65.9±18.1	64.0±14.1	97.8±2.4	**98.2±2.1**	83.1±14.6
8	70.1±17.9	61.3±18.0	61.1±15.2	96.8±4.2	**97.3±4.2**	79.7±13.1
9	72.3±15.6	64.5±18.0	62.2±11.5	95.5±6.0	**97.5±2.4**	84.8±13.1
10	69.2±17.0	63.4±18.0	61.1±13.2	94.0±6.3	**96.0±4.5**	81.5±10.1
30	58.5	54.2	59.6	−*	**86.7**	61.0

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

SR is very Efficient

c	Processing time (s)					
	Baseline	LSI	PLSI	LPP	SR	NMF
2	6.25	0.43	3.22	11.50	1.08	6.0
3	18.23	0.67	7.49	26.81	1.95	23.1
4	29.74	0.96	11.23	33.56	2.62	63.3
5	61.82	1.50	18.67	76.37	4.50	113.7
6	66.51	1.78	20.70	65.30	4.37	238.4
7	117.63	2.97	31.57	143.86	7.65	389.5
8	171.76	4.20	40.06	179.03	9.27	766.6
9	193.85	5.00	45.57	228.12	10.93	869.7
10	261.05	6.48	56.79	266.53	13.09	1348.3
30	2720.21	132.12	511.53	$-^*$	224.71	15101.0

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Good Performance on Content-based Image Retrieval Tasks (Semi-supervised)

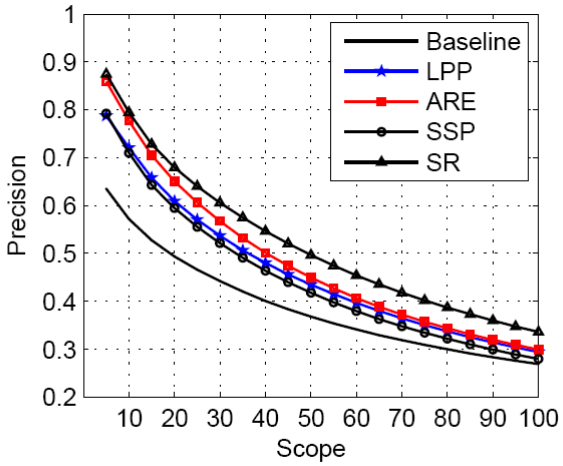


Figure: Precision-scope curve on feedback iteration 1

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Good Performance on Content-based Image Retrieval Tasks (Semi-supervised)

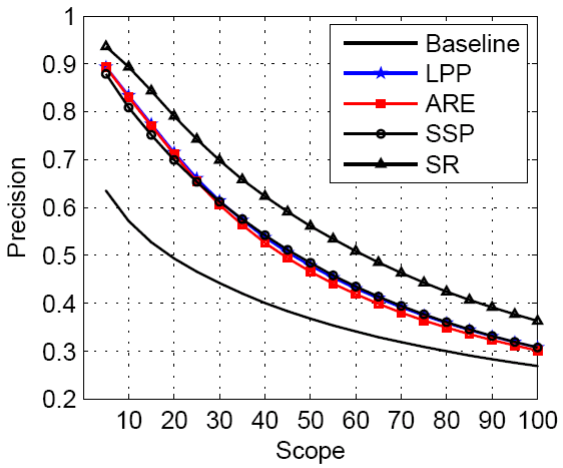


Figure: Precision-scope curve on feedback iteration 2

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Performance Comparison (Efficiency)

Table: Time on processing one query for each method (s)

	t_W	t_{SVD}	t_{GEigen}	t_{All}
LPP	0.062	0.453	0.494	1.009
ARE			0.489	1.004
SSP			0.487	1.002
		t_{SEigen}	t_{RLS}	
SR		0.024	0.041	0.127

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Performance on Isolet (Spoken Letter)

	Error (%)				
Training Set	LDA	KDA	SRKDA	KDA/QR	SVM
Isolet1	15.27	**11.74**	**12.89**	17.19	**12.51**
Isolet1+2	6.61	**3.79**	**3.85**	7.63	4.11
Isolet1+2+3	5.90	**2.99**	**3.08**	7.12	3.34
Isolet1+2+3+4	5.71	**2.82**	**2.89**	6.86	3.27

	Time (s)					Speedup
Training Set	LDA	KDA	SRKDA	KDA/QR	SVM	
Isolet1	1.93	18.86	1.21	0.93	4.75	15.6
Isolet1+2	2.14	134.6	5.51	3.60	13.79	24.4
Isolet1+2+3	2.37	451.6	14.09	7.98	23.84	32.1
Isolet1+2+3+4	2.56	991.2	27.86	14.02	34.82	35.6

*Column labeled "Speedup" shows how many times faster the SRKDA is (comparing to ordinary KDA).

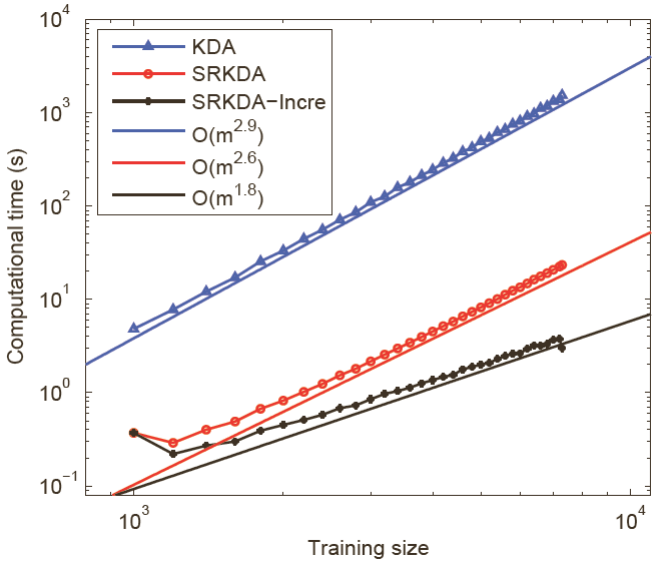
© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Performance on USPS (Handwritten Digit)

	Error (%)				
Training Set	LDA	KDA	SRKDA	KDA/QR	SVM
1500	10.61	6.58	**5.88**	10.86	6.85
3000	9.77	5.53	**5.38**	10.66	5.58
4500	9.52	5.53	**4.88**	9.67	5.13
6000	9.92	5.03	**4.43**	9.37	5.08
7291	10.26	4.83	**4.04**	9.02	4.83

	Time (s)					Speedup
Training Set	LDA	KDA	SRKDA	KDA/QR	SVM	
1500	0.21	14.97	0.92	0.66	0.78	16.3
3000	0.27	111.9	4.35	2.61	2.20	25.7
4500	0.34	354.3	11.29	5.85	4.06	31.4
6000	0.40	825.3	22.74	10.41	6.22	36.3
7291	0.47	1553.6	37.59	15.60	8.18	41.3

Computational Complexity of Incremental KSR



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Outline

- Part 1 – Traditional linear methods
- Part 2 – Graph-based methods
- Part 3 – Linear and kernel extension
- Part 4 – Spectral regression for efficient computation
- Part 5 – Sparse subspace learning for feature selection

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

The Projective Function (Projective Vector) is Dense

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \qquad \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$$

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T AA^T \mathbf{x}_j = \mathbf{z}_i^T \mathbf{z}_j \qquad \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T AA^T (\mathbf{x}_i - \mathbf{x}_j)}$$

$$\sqrt{(A^T\mathbf{x}_i - A^T\mathbf{x}_j)^T (A^T\mathbf{x}_i - A^T\mathbf{x}_j)}$$

$$\sqrt{(\mathbf{z}_i - \mathbf{z}_j)^T (\mathbf{z}_i - \mathbf{z}_j)}$$

$$\mathbf{z} = A^T \mathbf{x}$$

$$y_i = \mathbf{a}^T \mathbf{x} = \sum_{j=1}^{m} a_j x_j$$

Every new feature is a linear combination of all the original features

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Sparse Projective Function for Better Interpretation

$$y = \mathbf{a}^T \mathbf{x} = \sum_{j=1}^{m} a_j x_j$$

- Sparse a
  - Better Interpretation (Feature Selection)
  - Avoid Over-fitting

$$y = f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

  - "Support Vector"

Sparse Subspace Learning

$$\max \frac{\mathbf{a}^T XWX^T \mathbf{a}}{\mathbf{a}^T XDX^T \mathbf{a}} \qquad \max \frac{\boldsymbol{\alpha}^T KWK\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T KDK\boldsymbol{\alpha}}$$
$$\text{card}(\mathbf{a}) = k \qquad \text{card}(\boldsymbol{\alpha}) = k$$

- General sparse subspace learning problem:

$$\max \frac{\mathbf{a}^T A\mathbf{a}}{\mathbf{a}^T B\mathbf{a}}, \quad \text{card}(\mathbf{a}) = k$$
$$= \max \frac{\mathbf{b}^T A_k \mathbf{b}}{\mathbf{b}^T B_k \mathbf{b}}$$

$\mathbf{b} \in \mathrm{R}^k$ contains the k non-zero elements in $\mathbf{a}$

$k \times k$ sub-matrices A_k, B_k

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Possible Solution with O(m⁴)

$$\max \frac{\mathbf{a}^T A\mathbf{a}}{\mathbf{a}^T B\mathbf{a}} = \max \frac{\mathbf{b}^T A_k \mathbf{b}}{\mathbf{b}^T B_k \mathbf{b}} \qquad \text{card}(\mathbf{a}) = k$$

- Greedy algorithm which combines backward elimination and forward selection. (Moghaddam et al. NIPS 2005, ICML 2006)
  - The cost of backward elimination is with complexity

$$O(m^4)$$

  - The algorithm does not give suggestions on how to find more than ONE sparse projections ("eigenvector"s).

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Sparse Subspace Learning in Spectral Regression

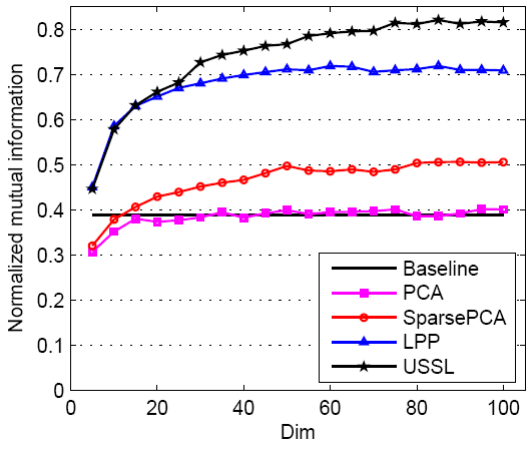
$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \alpha \|\mathbf{a}\|^2 \right)$$

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{n} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \alpha \sum_{j=1}^{m} |a_j| \right)$$

- Lasso regression
  - Sparse solution
  - Can be efficiently solved by LARS algorithm (Efron et. al 2004) $O\left(m^3\right)$
  - More than one projective function

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

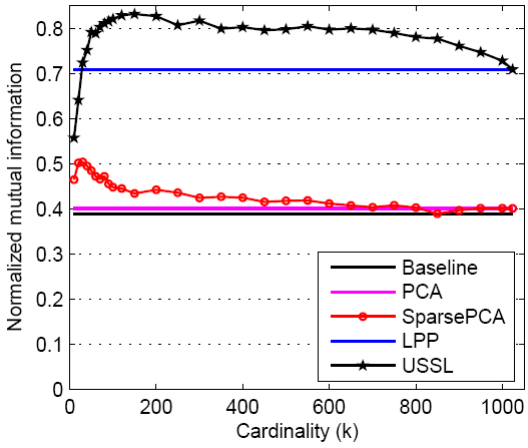
Face Clustering on PIE, the performance varies with the number of reduced dimensions



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Face Clustering on PIE, the performance varies with the cardinality



© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Performance of Sparse Kernel Discriminant Analysis

TABLE VI
CLASSIFICATION ERROR ON ISOLET DATASET

Training Set	Error (%)			Sparsity
	KDA	SRKDA	SRKDA(Sparse)	
Isolet1	11.74	12.89	11.74	60%
Isolet1+2	3.79	3.85	3.59	60%
Isolet1+2+3	2.99	3.08	2.82	60%
Isolet1+2+3+4	2.82	2.89	2.82	60%

TABLE VII
CLASSIFICATION ERROR ON USPS DATASET

Training Set	Error (%)			Sparsity
	KDA	SRKDA	SRKDA(Sparse)	
1500	6.58	5.88	5.83	60%
3000	5.53	5.38	5.13	60%
4500	5.53	4.88	4.73	60%
6000	5.03	4.43	4.04	60%
7291	4.83	4.04	3.94	60%

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Conclusion

- Big ideas
  - Manifolds are everywhere.
  - Graph-based methods can learn them.
  - The projective function can be efficiently computed through spectral regression.
- Ongoing work
  - Theoretical guarantees & extrapolation
  - Spherical & toroidal geometries
  - Applications (vision, graphics, speech)

© Deng Cai, Xiaofei He, Jiawei Han SDM'09 Tutorial, April, 2009

Thank You!