UNIVERSITY OF MINNESOTA TWIN CITIES

A brief tour of the spectral problems of data mining

Yousef Saad
Department of Computer Science and Engineering

University of Minnesota

Householder XVII, Zeuthen - June 6th, 2008

Team members involved in this work – Support

Past:

- Efi Kokiopoulou [Now at the U. of Lausanne]

Current:

- Jie Chen [grad student]
- Sofia Sakellaridi [grad student]
- Haw-Ren Fang [Post-Doc]

Support:

- National Science Foundation

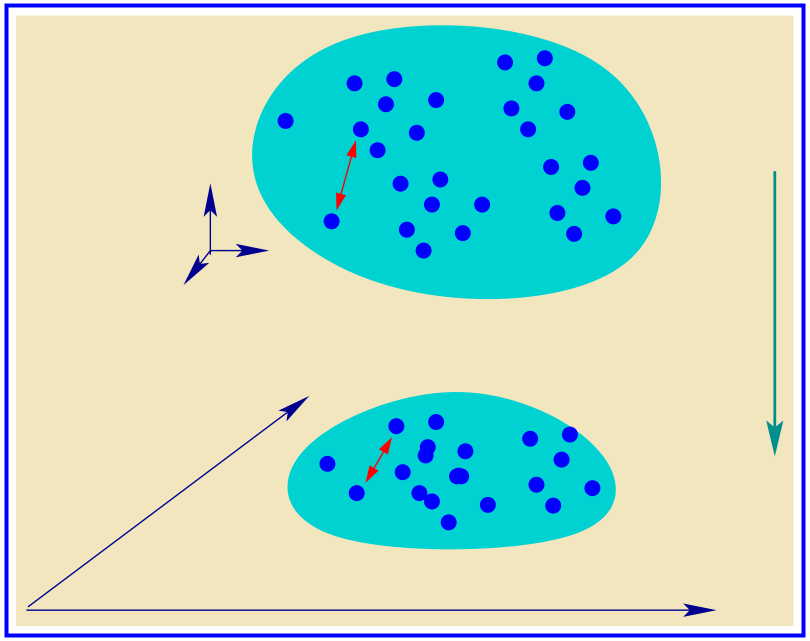
Introduction, background, and motivation

Common goal of data mining methods: to extract meaningful information or patterns from data. Very broad area – includes: data analysis, machine learning, pattern recognition, information retrieval, ...

- ➤ Main tools used: linear algebra; graph theory; approximation theory; optimization; ...
- ➤ In this talk: emphasis on dimension reduction techniques and the interrelations between techniques

The problem

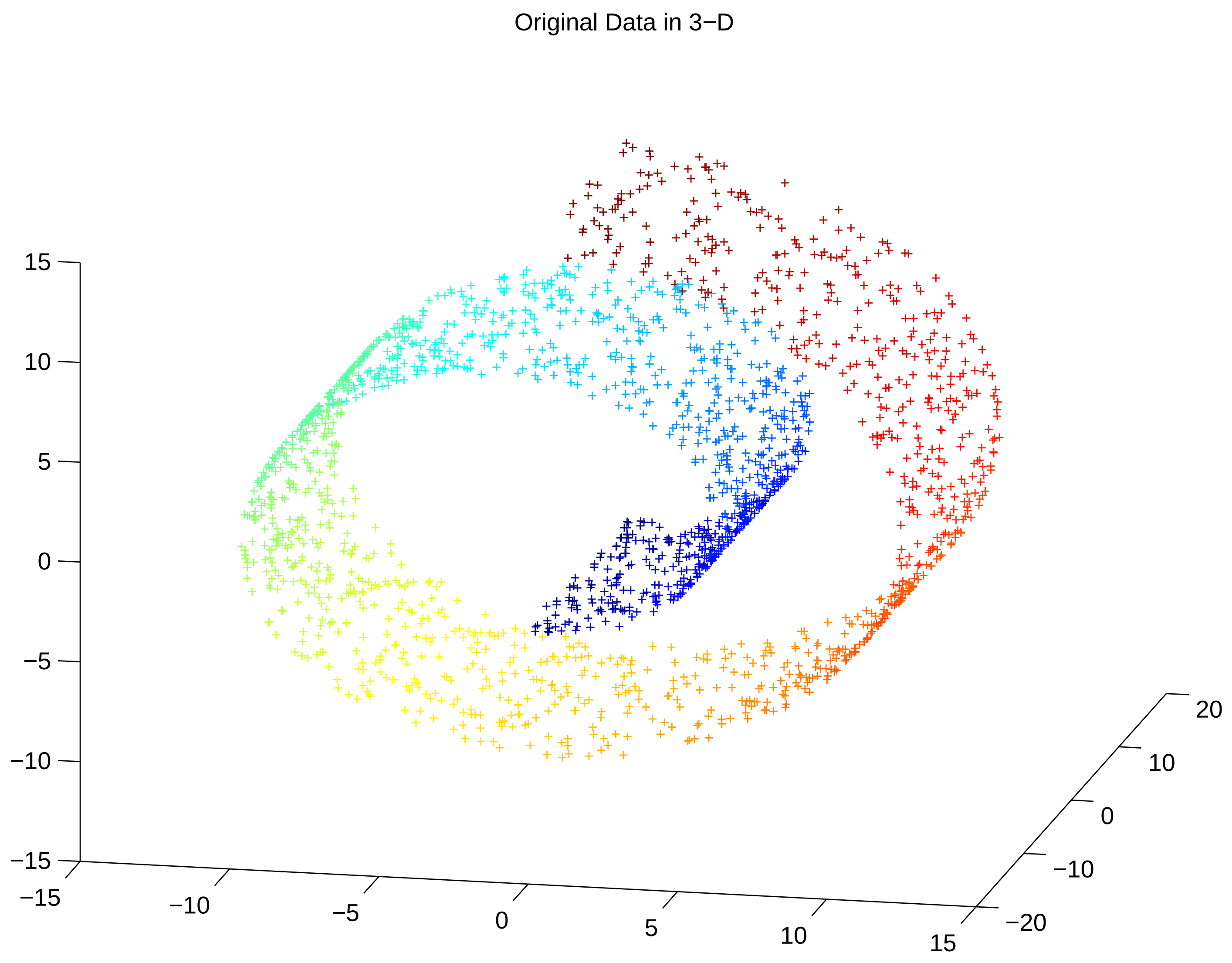
- Given $d \ll m$ find a mapping $\Phi : x \in \mathbb{R}^m \longrightarrow y \in \mathbb{R}^d$
- Mapping may be explicit (e.g., $y = V^T x$)
- Or implicit (nonlinear)



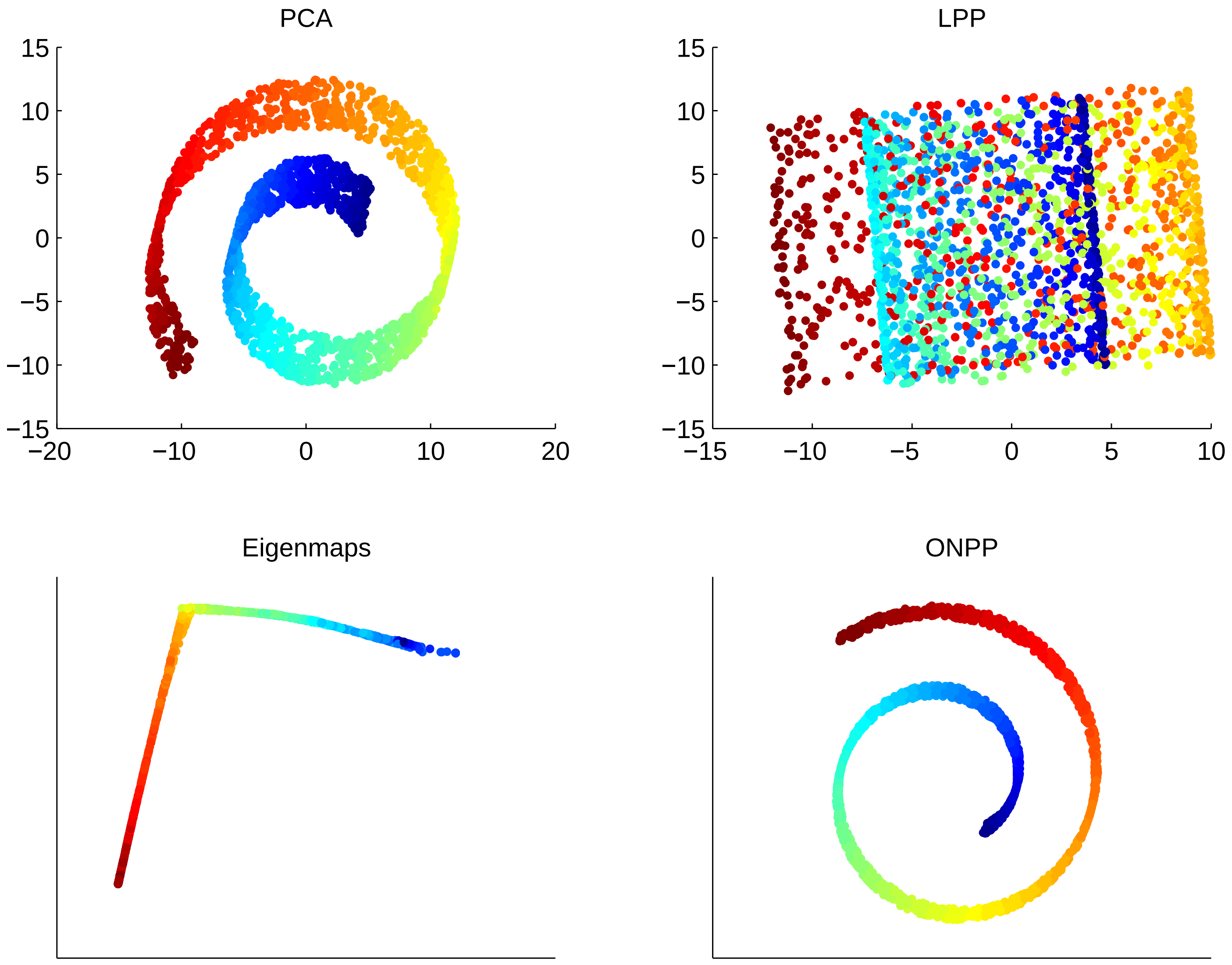
Practically: Given $X \in \mathbb{R}^{m \times n}$, we want to find a low-dimensional representation $Y \in \mathbb{R}^{d \times n}$ of X

- Two classes of methods: (1) projection techniques and (2) nonlinear implicit methods.

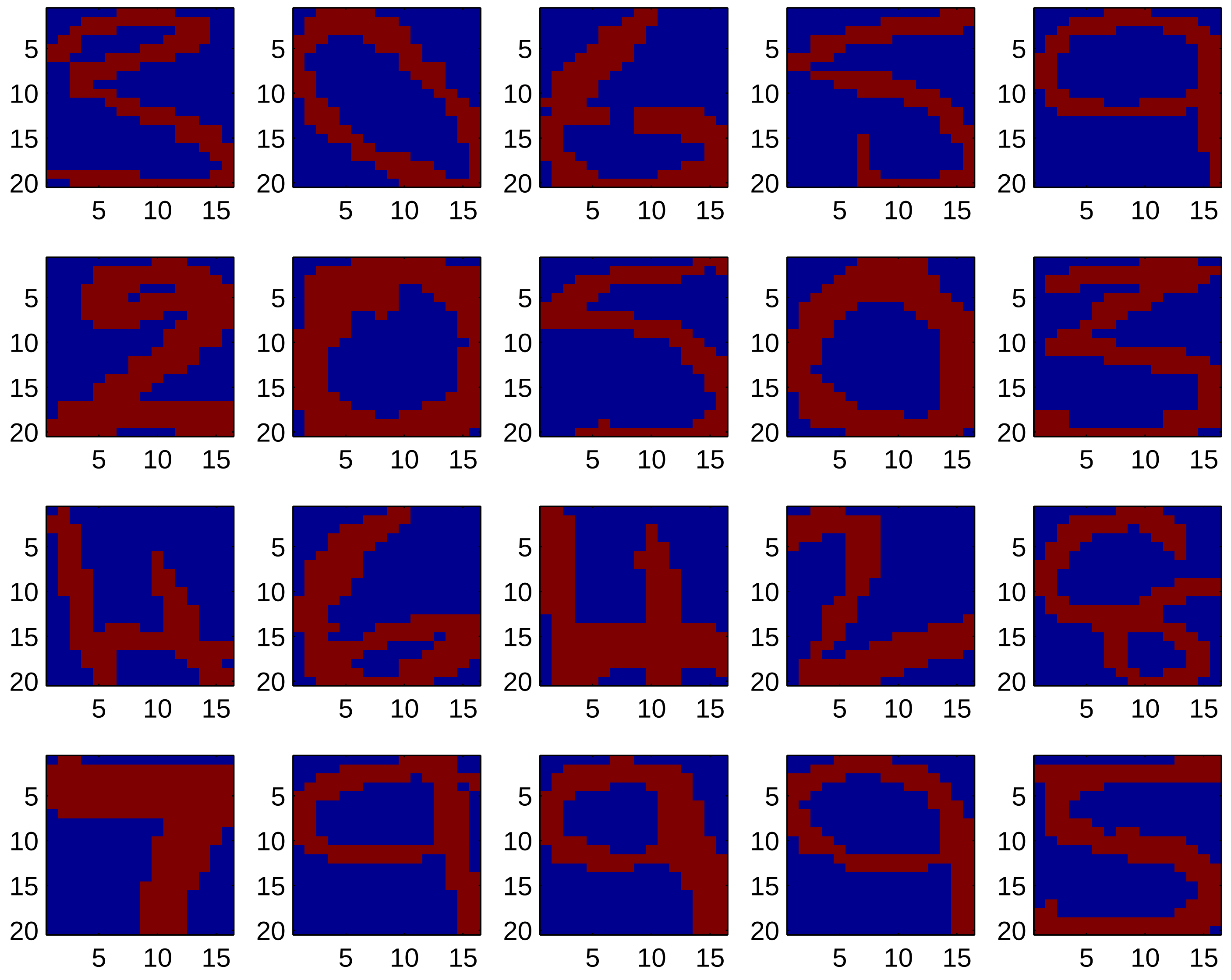
Example 1: The ‘Swill-Roll’ (2000 points in 3-D)



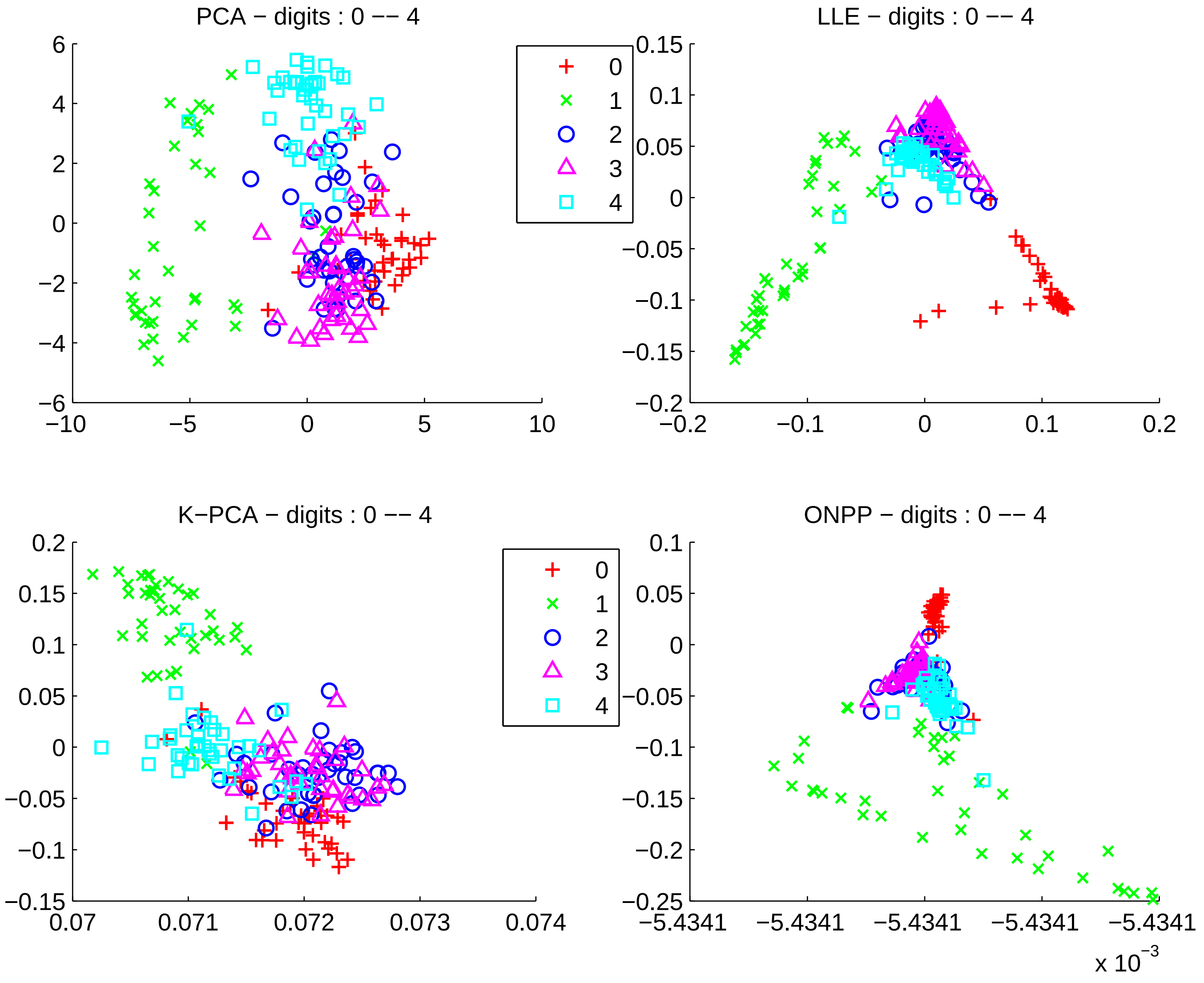
2-D ‘reductions’:



Example 2: Digit images (a sample of 20)



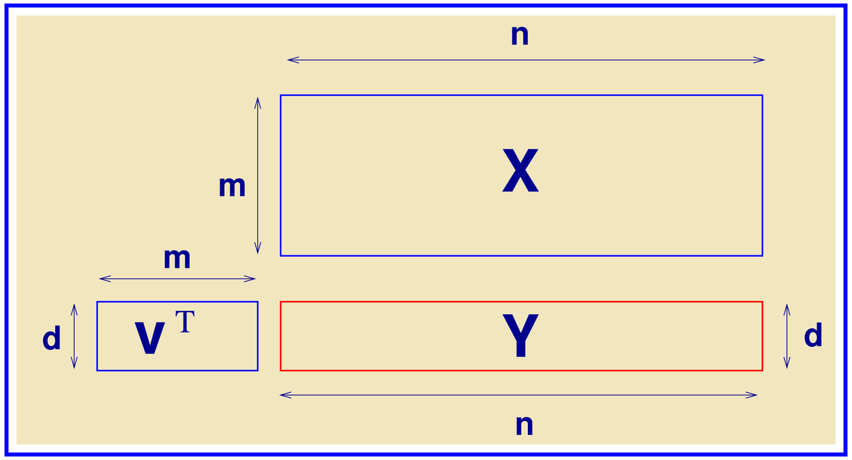
2-D 'reductions':



Projection-based Dimensionality Reduction

Given: a data set $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$, and $\boldsymbol{d}$ the dimension of the desired reduced space $\boldsymbol{Y}$.

Want: a linear transformation from $\boldsymbol{X}$ to $\boldsymbol{Y}$



$$\boldsymbol{X} \in \mathbb{R}^{m\times n}$$
$$\boldsymbol{V} \in \mathbb{R}^{m\times d}$$
$$\boxed{\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}}$$
$$\rightarrow \quad \boldsymbol{Y} \in \mathbb{R}^{d\times n}$$

- $\boldsymbol{m}$-dimens. objects ($\boldsymbol{x}_i$) ‘flattened’ to $\boldsymbol{d}$-dimens. space ($\boldsymbol{y}_i$)

Constraint: The $\boldsymbol{y}_i$’s must satisfy certain properties

- Optimization problem

Linear Dimensionality Reduction: PCA

- In PCA projected data must have maximum variance, i.e., we need to maximize over all orthogonal $m \times d$ matrices $\boldsymbol{V}$:

$$\boxed{\sum_i \|\boldsymbol{y}_i - \tfrac{1}{n}\sum_j \boldsymbol{y}_j\|_2^2 = \cdots = \text{Tr}\ [\boldsymbol{V}^\top \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\top \boldsymbol{V}]}$$

Where: $\bar{\boldsymbol{X}} = \boldsymbol{X}(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ == origin-recentered version of $\boldsymbol{X}$

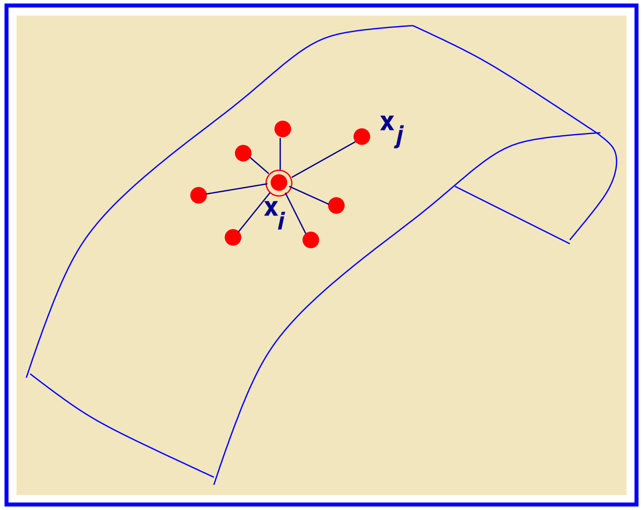
- Solution $\boldsymbol{V} =$ { dominant eigenvectors } of the covariance matrix == Set of left singular vectors of $\bar{\boldsymbol{X}}$
- Solution $\boldsymbol{V}$ also minimizes 'reconstruction error' ..

$$\sum_i \|\boldsymbol{x}_i - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_i\|^2 = \sum_i \|\boldsymbol{x}_i - \boldsymbol{V}\boldsymbol{y}_i\|^2$$

- .. and it also maximizes [Korel and Carmel 04] $\sum_{i,j} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$

Laplacean Eigenmaps (Belkin-Niyogi-02)

- Not a linear (projection) method but a Nonlinear method
- Starts with k-nearest-neighbors graph



- Defines the graph Laplacean $L = D - W$. Simplest:

$$D = \mathrm{diag}(deg(i)); \quad w_{ij} = \begin{cases} 1 & \text{if } j \in N_i \\ 0 & \text{else} \end{cases}$$

with N_i = neighborhood of i (excl. i); $deg(i) = |N_i|$

A few properties of graph Laplacean matrices

➤ Let L = any matrix s.t. $L = D - W$, with $D = diag(d_i)$ and

$$w_{ij} \geq 0, \qquad d_i = \sum_{j \neq i} w_{ij}$$

Property 1: for any $x \in \mathbb{R}^n$:

$$x^\top L x = \frac{1}{2} \sum_{i,j} w_{ij} |x_i - x_j|^2$$

Property 2: (generalization) for any $Y \in \mathbb{R}^{d \times n}$:

$$\text{Tr}\,[Y L Y^\top] = \frac{1}{2} \sum_{i,j} w_{ij} \|y_i - y_j\|^2$$

Property 3: For the particular $\boldsymbol{L} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$

$$\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top = \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\top == n \times \text{Covariance matrix}$$

[Proof: 1) $\boldsymbol{L}$ is a projector: $\boldsymbol{L}^\top\boldsymbol{L} = \boldsymbol{L}^2 = \boldsymbol{L}$, and 2) $\boldsymbol{X}\boldsymbol{L} = \bar{\boldsymbol{X}}$]

- Consequence-1: PCA equivalent to maximizing $\sum_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$
- Consequence-2: what about replacing trivial $\boldsymbol{L}$ with something else? [viewpoint in Koren-Carmel'04]

Property 4: (Graph partitioning) If x is a vector of signs (± 1) then

$$x^\top L x = 4 \times \text{('number of edge cuts')}$$

edge-cut = pair (i, j) with $x_i \neq x_j$

➤ Consequence: Can be used for partitioning graphs, or ‘clustering’

[take $p = sign(u_2)$, where u_2 = 2nd smallest eigenvector..]

Return to Laplacean eigenmaps approach

Laplacean Eigenmaps *minimizes*

$$\mathcal{F}_{EM}(Y) = \sum_{i,j=1}^{n} w_{ij} \|y_i - y_j\|^2 \quad \text{subject to} \quad YDY^\top = I \ .$$

Notes:

1. Motivation: if $\|x_i - x_j\|$ is small (orig. data), we want $\|y_i - y_j\|$ to be also small (low-D data)
2. Note Min instead of Max as in PCA [counter-intuitive]
3. Above problem uses original data indirectly through its graph

- Problem translates to:

$$\min_{\begin{cases} Y \in \mathbb{R}^{d\times n} \\ Y D Y^\top = I \end{cases}} \text{Tr}\left[Y(D-W)Y^\top\right] .$$

- Solution (sort eigenvalues increasingly):

$$(D-W)u_i = \lambda_i D u_i \; ; \quad y_i = u_i^\top; \quad i = 1, \cdots, d$$

- Note: an $n \times n$ sparse eigenvalue problem [In 'sample' space]
- Note: can assume $D = I$. Amounts to rescaling data. Problem

becomes

$$\boxed{(I-W)u_i = \lambda_i u_i \; ; \quad y_i = u_i^\top; \quad i = 1, \cdots, d}$$

Why smallest eigenvalues vs largest for PCA?

Intuition:

Graph Laplacean and ‘unit’ Laplacean are very different: one involves a sparse graph (More like a discr. differential operator). The other involves a dense graph. (More like a discr. integral operator). They should be treated as the inverses of each other.

➤ Viewpoint confirmed by what we learn from Kernel approach

Locally Linear Embedding (Roweis-Saul-00)

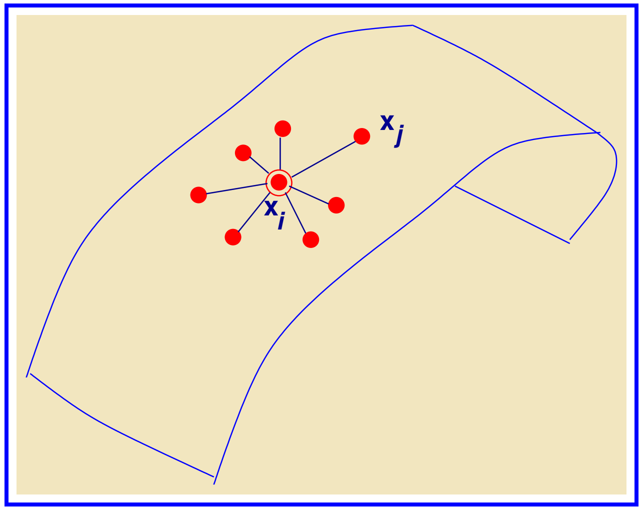
➤ LLE is very similar to Eigenmaps. Main differences:

1) Graph Laplacean matrix is replaced by an ‘affinity’ graph

2) Objective function is changed: want to preserve graph

1. Graph: Each x_i is written as a convex combination of its k nearest neighbors:

$x_i \approx \Sigma w_{ij} x_j, \quad \sum_{j \in N_i} w_{ij} = 1$



➤ Optimal weights computed (’local calculation’) by minimizing

$$\|x_i - \Sigma w_{ij} x_j\| \quad \text{for} \quad i = 1, \cdots, n$$

2. Mapping:

The y_i's should obey the same 'affinity' as x_i's $\rightsquigarrow$

Minimize:

$$\sum_i \left\| y_i - \sum_j w_{ij} y_j \right\|^2 \quad \text{subject to:} \quad Y\mathbf{1} = 0, \quad YY^\top = I$$

Solution:

$$\boxed{(I - W^\top)(I - W)u_i = \lambda_i u_i; \qquad y_i = u_i^\top .}$$

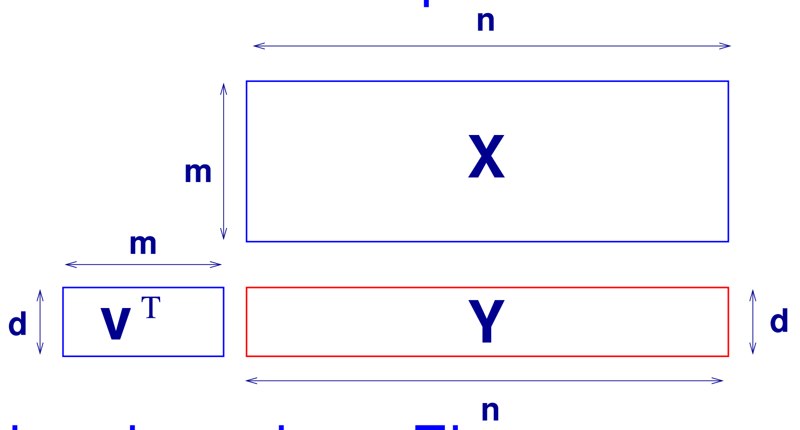
- $(I - W^\top)(I - W)$ replaces the graph Laplacean of eigenmaps

Locally Preserving Projections (He-Niyogi-03)

- LPP is a linear dimensionality reduction technique

- Recall the setting:

Want $\boldsymbol{V} \in \mathbb{R}^{m \times d}$; $\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}$



- Starts with the same neighborhood graph as Eigenmaps: $\boldsymbol{L} \equiv \boldsymbol{D} - \boldsymbol{W}$ = graph 'Laplacean'; with $\boldsymbol{D} \equiv diag(\{\Sigma_i w_{ij}\})$.

- Optimization problem is to solve

$$\min_{Y \in \mathbb{R}^{d\times n},\ YDY^\top = I} \quad \Sigma_{i,j} w_{ij} \left\| y_i - y_j \right\|^2, \quad Y = V^\top X.$$

- Difference with eigenmaps: Y is a projection of X data
- Solution (sort eigenvalues increasingly)

$$XLX^\top v_i = \lambda_i XDX^\top v_i \qquad y_{i,:} = v_i^\top X$$

- Note: essentially same method in [Koren-Carmel'04] called 'weighted PCA' [viewed from the angle of improving PCA]

ONPP (Kokiopoulou and YS '05)

- Orthogonal Neighborhood Preserving Projections
- Can be viewed as a linear version of LLE
- Uses the same graph as LLE. Objective: preserve the affinity graph (as in LEE) *but* by means of an orthogonal projection
- Objective function

$$\boldsymbol{\Phi}(\boldsymbol{Y}) = \Sigma_i \left\| \boldsymbol{y}_i - \Sigma_j w_{ij} \boldsymbol{y}_j \right\|^2 \quad \text{Constraint: } \boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}, \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$$

- Notice that

$$\boldsymbol{\Phi}(\boldsymbol{Y}) = \|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{W}^\top\|_F^2 = \cdots = \text{Tr}\left[\boldsymbol{V}^\top \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{W}^\top)(\boldsymbol{I} - \boldsymbol{W})\boldsymbol{X}^\top \boldsymbol{V}\right]$$

Resulting problem:

$$\min_{\substack{V \in \mathbb{R}^{m\times d};\\ V^\top V = I}} \quad \mathbf{Tr}\left[V^\top \underbrace{X(I - W^\top)(I - W)X^\top}_{M} V \right]$$

Solution: Columns of V = eigenvectors of M associated with smallest d eigenvalues

➤ Can be computed as d lowest left singular vectors of

$$X(I - W^\top)$$

A unified view

Method	Object. (min)	Constraint
PCA/MDS	$\text{Tr}\,[V^\top X(-I + ee^\top)X^\top V]$	$V^\top V = I$
LLE	$\text{Tr}\,[Y(I - W^\top)(I - W)Y^\top]$	$YY^\top = I$
Eigenmaps	$\text{Tr}\,[Y(I - W)Y^\top]$	$YY^\top = I$
LPP	$\text{Tr}\,[V^\top X(I - W)X^\top V]$	$V^\top XX^\top V = I$
ONPP	$\text{Tr}\,[V^\top X(I - W^\top)(I - W)X^\top V]$	$V^\top V = I$
LDA	$\text{Tr}\,[V^\top X(I - H)X^\top V]$	$V^\top XX^\top V = I$

- Let $M = I - W$ = a Laplacean matrix ($-I + ee^\top$ for PCA/MDS); or the LLE matrix $(I - W)(I - W^\top)$, or geodesic distance matrix (ISOMAP).
- All techniques lead to one of two types of problems
- First type is:

$$\boxed{\min_{\begin{cases} Y \in \mathbb{R}^{d\times n} \\ YY^\top = I \end{cases}} \textbf{Tr}\left[YMY^\top\right]}$$

- Y obtained from solving an eigenvalue problem
- LLE, Eigenmaps (normalized), ..

➤ And the second type is:

$$\boxed{\min_{\begin{cases} V \in \mathbb{R}^{m\times d} \\ V^\top G\, V = I \end{cases}} \textbf{Tr}\left[V^\top X M X^\top V\right].}$$

➤ G is either the identity matrix or $XDX^\top$ or $XX^\top$.

➤ Low-Dim. data : $Y = V^\top X$

Important observation: 2nd is just a projected version of the 1st, i.e., approximate eigenvectors are sought in Span $\{X\}$ [Rayleigh-Ritz procedure]

➤ Problem is of dim. m (dim. of data) not n (# of samples).

➤ This difference can be mitigated by resorting to Kernels..

TIME FOR A MATLAB DEMO

A brief tour of Kernels

➤ Kernels emply an implicit nonlinear map of original data into a higher dimensional feature space $\mathbb{H}$.

$$\boldsymbol{\Phi} \quad : \mathbb{R}^m \longrightarrow \mathbb{H}$$

➤ Mapping $\boldsymbol{\Phi}$ only known through its Kernel on data:

$$< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) > \equiv \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

➤ Can do PCA, eigenmaps, ..., on this data without using $\boldsymbol{\Phi}$

Kernel PCA (Ham et. al. 2004)

- Classical PCA on the set $\{\boldsymbol{\Phi}\}$

$$\min \mathrm{Tr}\,[\boldsymbol{V}^\top \bar{\boldsymbol{\Phi}} \bar{\boldsymbol{\Phi}}^\top \boldsymbol{V}] \quad \text{subject to} \quad \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$$

- Projected data $\boldsymbol{Y} = \boldsymbol{V}^\top \bar{\boldsymbol{\Phi}}$
- Problem to solve $\bar{\boldsymbol{\Phi}} \bar{\boldsymbol{\Phi}}^\top \boldsymbol{u}_i = \boldsymbol{\lambda} \boldsymbol{u}_i$

- Right singular vector approach. Multiply both sides by $\phi^\top$:

$$\underbrace{[\bar{\boldsymbol{\Phi}}^\top \bar{\boldsymbol{\Phi}}]}_{\bar{\boldsymbol{K}}} \bar{\boldsymbol{\Phi}}^\top \boldsymbol{u}_i = \boldsymbol{\lambda}_i \bar{\boldsymbol{\Phi}}^\top \boldsymbol{u}_i$$

- Note

  1. $\bar{\boldsymbol{\Phi}}^\top \bar{\boldsymbol{\Phi}} = (\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^\top)\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^\top)$ Denoted by $\bar{\boldsymbol{K}}$
  2. $\bar{\boldsymbol{\Phi}}^\top \boldsymbol{u}_i = \boldsymbol{y}_i^\top$ (recall $\boldsymbol{Y} = \boldsymbol{V}^\top \bar{\boldsymbol{\Phi}}$)

- Result: columns of $\boldsymbol{Y}^\top$ are largest eigenvectors of $\bar{\boldsymbol{K}}$

$$\bar{\boldsymbol{K}} \boldsymbol{y}_i^\top = \boldsymbol{\lambda}_i \boldsymbol{y}_i^\top \qquad \text{or} \qquad \boldsymbol{y}_i \bar{\boldsymbol{K}} = \boldsymbol{\lambda}_i \boldsymbol{y}_i$$

- Compare with Eigenmaps: the columns of $\boldsymbol{Y}^\top$ ($\boldsymbol{n}$-vectors) are smallest eigenvectors of $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{W}$
- Interpretation [see Ham, Mika, and Scölkopf, 2004]: Eigenmaps can be interpreted as Kernel PCA with Kernel $\boldsymbol{K} = \boldsymbol{L}^\dagger$.

Kernel LPP & ONPP

- Proceed similarly to PCA.
- Assumption & notation: $\Phi = \Phi(X)$, $K = \Phi^\top \Phi$ is invertible

LPP: Problem in feature space:

$$\min \text{Tr}\left[V^\top \Phi(X) L \Phi(X)^\top V\right] \quad \text{Subj. to} \quad V^\top \Phi D \Phi^\top V = I$$

- Leads to the eigenvalue problem:

$$\boxed{\Phi L \Phi^\top u_i = \lambda_i \Phi D \Phi^\top u_i}$$

- Left multiply by $\Phi^\top$, then by K^{-1}, + recall that $y_i^\top = \Phi^\top u_i$:

$$\boxed{L y_i^\top = \lambda_i D y_i^\top}$$

- Note: K disappeared from picture; What's the catch??.

Kernel-ONPP

$$\min_{V \in \mathbb{R}^{L\times d}\; V^\top V = I} \quad \text{Tr}\left[V^\top \Phi(X) M \Phi(X)^\top V\right]$$

➤ Leads to the eigenvalue problem:

$$\boxed{\Phi M \Phi^\top u_i = \lambda_i u_i}$$

➤ Multiply by $\Phi^\top$ and note as before $K = \Phi^\top \Phi$, $y_i^\top = \Phi^\top u_i$:

$$\boxed{K M y_i^\top = \lambda_i y_i^\top \quad \text{or} \quad M y_i^\top = K^{-1} y_i^\top}$$

➤ Solution is set of eigenvectors of Matrix M – but constraint: K^{-1} - orthogonality

Conclusion

- So how is this related to intitial title of “efficient algorithms in data mining”?
- Answer: All these eigenvalue problems are not cheap to solve..
- .. and cost issue does not seem to bother practitioners too much for now..
- Ingredients that will become mandatory:

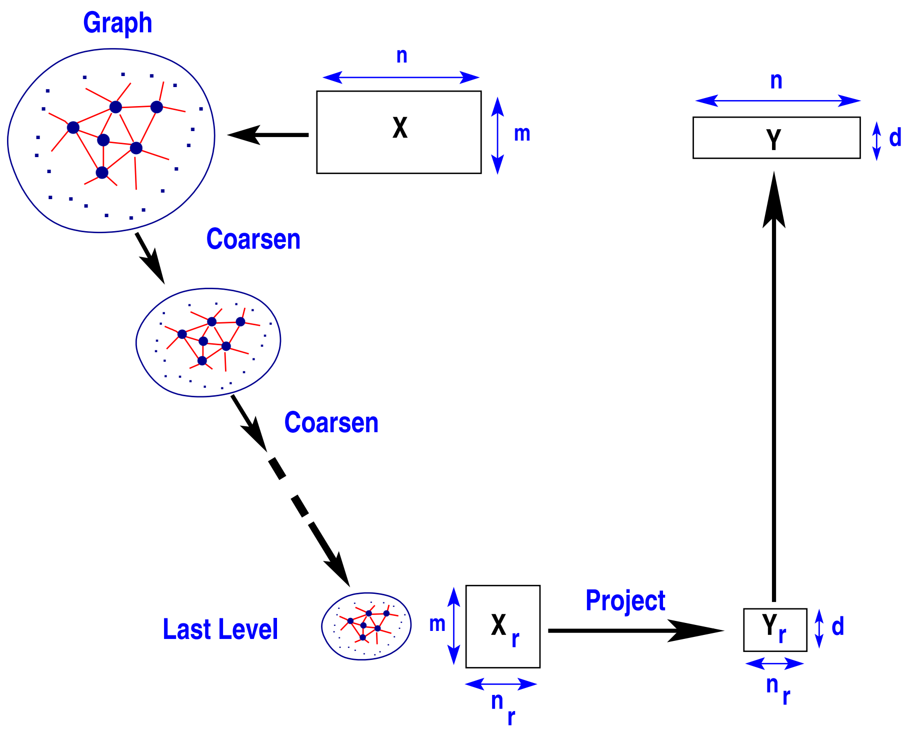
1. Avoid the SVD
2. Fast algorithms that do not sacrifice quality.
3. In particullar: Multilevel approaches
4. Multilinear algebra [tensors]

Multilevel techniques in brief

- Divide and conquer paradigms as well as multilevel methods in the sense of ‘domain decomposition’
- Main principle: very costly to do an SVD [or Lanczos] on the whole set. Why not find a smaller set on which to do the analysis – without too much loss?
- Tools used: graph coarsening, divide and conquer –
- For information retrieval we use hypergraphs

Multilevel Dimension Reduction

Main Idea: coarsen for a few levels. Use the resulting data set $\hat{X}$ to find a projector P from $\mathbb{R}^m$ to $\mathbb{R}^d$. P can be used to project original data or new data



➤ Gain: Dimension reduction is done with a much smaller set. Hope: not much loss compared to using whole data

Application to Information Retrieval

➤ Recall common approach:

1. Scale data [e.g. TF-IDF scaling:
2. Perform a (partial) SVD on resulting matrix $\boldsymbol{X} \approx \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^T$
3. Process query by same scaling (e.g. TF-IDF)
4. Compute similarities in d-dimensional space: $s_i = \langle \hat{\boldsymbol{q}}, \hat{\boldsymbol{x}}_i \rangle / \|\hat{\boldsymbol{q}}\| \|\hat{\boldsymbol{x}}_i\|$

where $[\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \dots, \hat{\boldsymbol{x}}_n] = \boldsymbol{V}_d^T \in \mathbb{R}^{d \times n}$; $\quad \hat{\boldsymbol{q}} = \boldsymbol{\Sigma}_d^{-1} \boldsymbol{U}_d^T \bar{\boldsymbol{q}} \in \mathbb{R}^d$

➤ Multilevel approach: replace SVD (or any other dim. reduction) by dimension reduction on coarse set. Only difference: TF-IDF done on the coarse set not original set.

Tests

Three public data sets used for experiments: `Medline`, `Cran` and `NPL` (cs.cornell.edu)

➤ Coarsening to a max. of 4 levels.

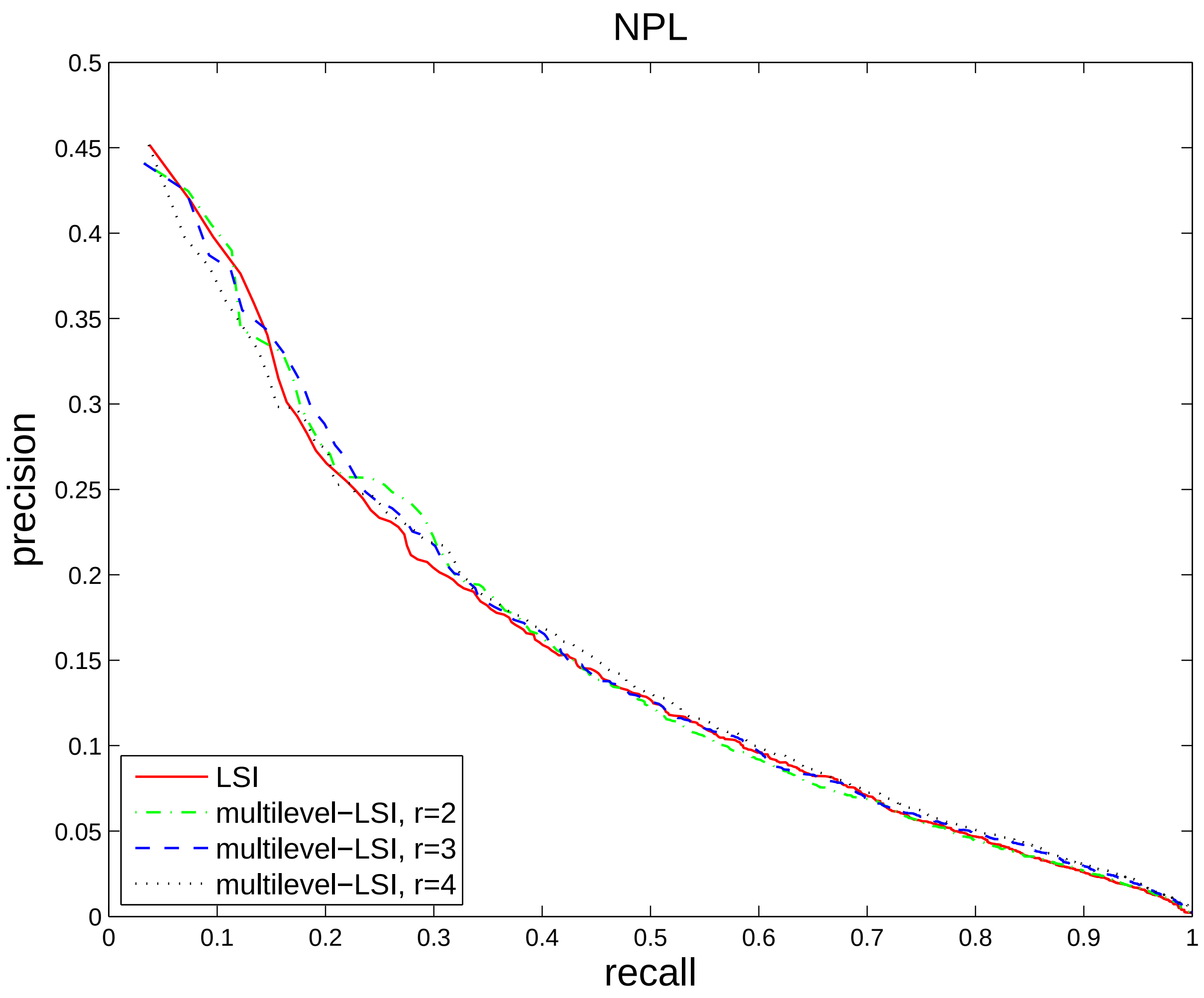
Data set	`Medline`	`Cran`	`NPL`
# documents	1033	1398	11429
# terms	7014	3763	7491
sparsity (%)	0.74%	1.41%	0.27%
# queries	30	225	93
avg. # rel./query	23.2	8.2	22.4

Results with NPL

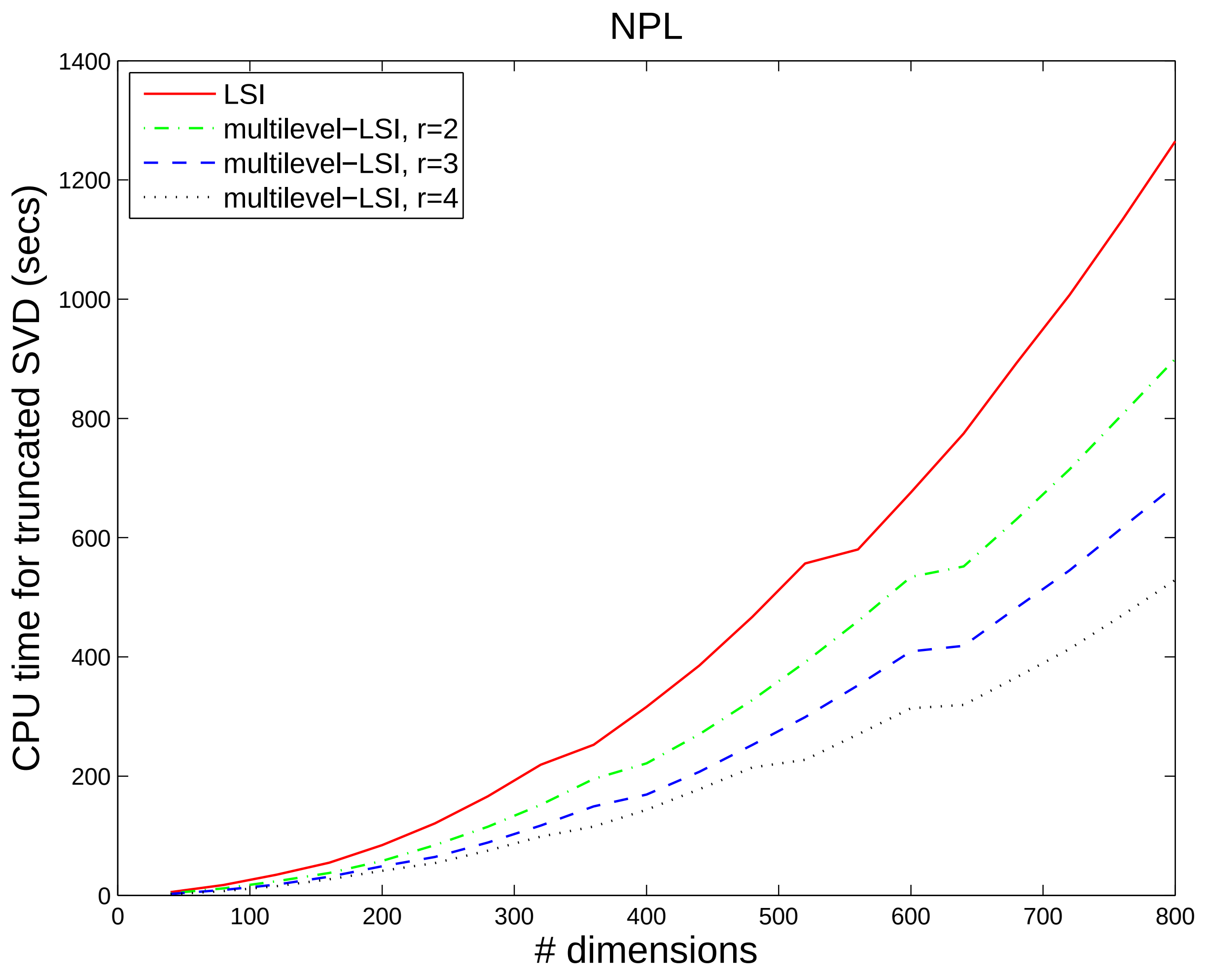
Statistics

Level	coarsen. time	# doc.	optimal # dim.	optimal avg. precision
#1	N/A	11429	736	23.5%
#2	3.68	5717	592	23.8%
#3	2.19	2861	516	23.9%
#4	1.50	1434	533	23.3%

Precision-Recall curves



CPU times for preprocessing (Dim. reduction part)


NPL
LSI
multilevel−LSI, r=2
multilevel−LSI, r=3
multilevel−LSI, r=4
CPU time for truncated SVD (secs)
# dimensions