Dimension reduction and graph based methods in data mining

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

NASCA-09 – 05/20/2009

Introduction: What is data mining?

➤ 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, interrelations between techniques, and graph theory tools.

Major tool of Data Mining: Dimension reduction

➤ Goal is not just to reduce computational cost but to:

- Reduce noise and redundancy in data
- Discover 'features' or 'patterns' (e.g., supervised learning)

➤ Techniques depend on application: Preserve angles? Preserve distances? Maximize variance? ..

The problem of Dimension Reduction

- Given $d \ll m$ find a mapping

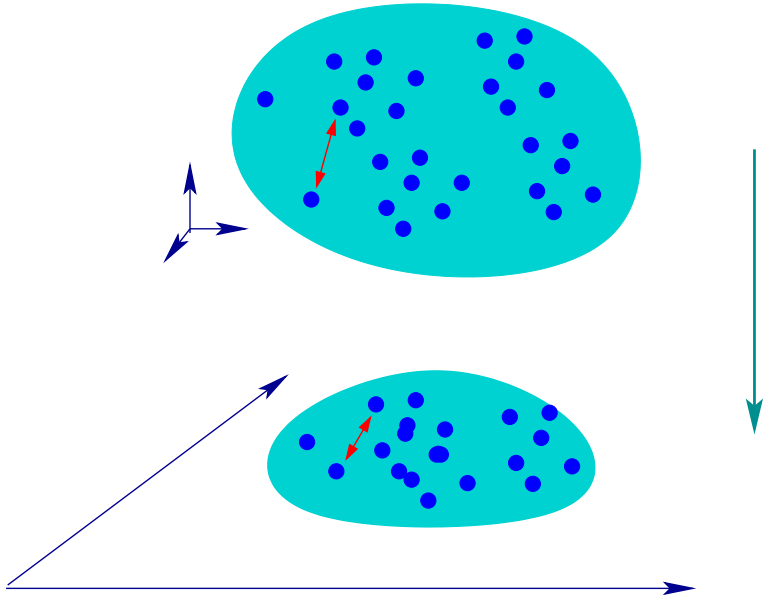
$$\Phi : x \in \mathbb{R}^m \longrightarrow y \in \mathbb{R}^d$$

- Mapping may be explicit [typically linear], e.g.:

$$y = V^T x$$

- Or implicit (nonlinear)

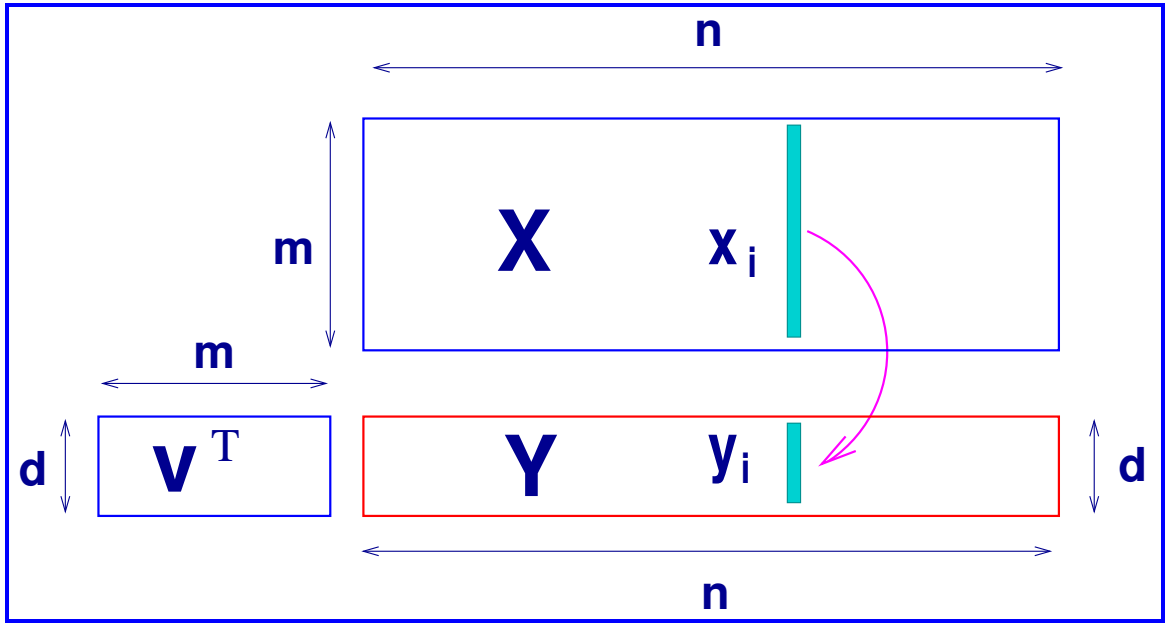
Practically: Given $X \in \mathbb{R}^{m \times n}$, we want to find a low-dimensional representation $Y \in \mathbb{R}^{d \times n}$ of X



Linear Dimensionality Reduction

Given: a data set $X = [x_1, x_2, \ldots, x_n]$, and d the dimension of the desired reduced space $Y = [y_1, y_2, \ldots, y_n]$.

Want: a linear transformation from X to Y



$$X \in \mathbb{R}^{m \times n}$$

$$V \in \mathbb{R}^{m \times d}$$

$$\boxed{Y = V^\top X}$$

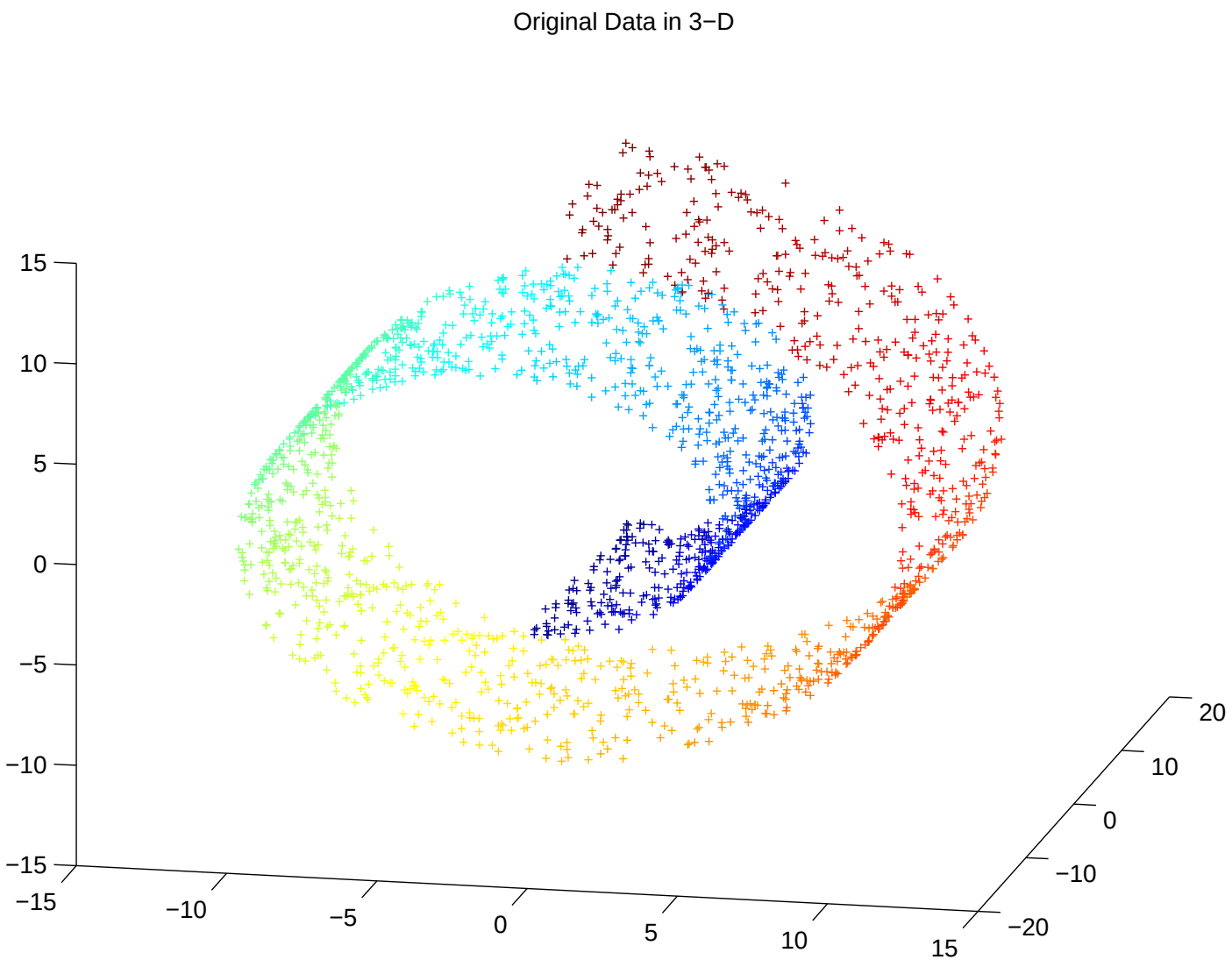
$$\rightarrow \quad Y \in \mathbb{R}^{d \times n}$$

- m-dimens. objects (x_i) 'flattened' to d-dimens. space (y_i)

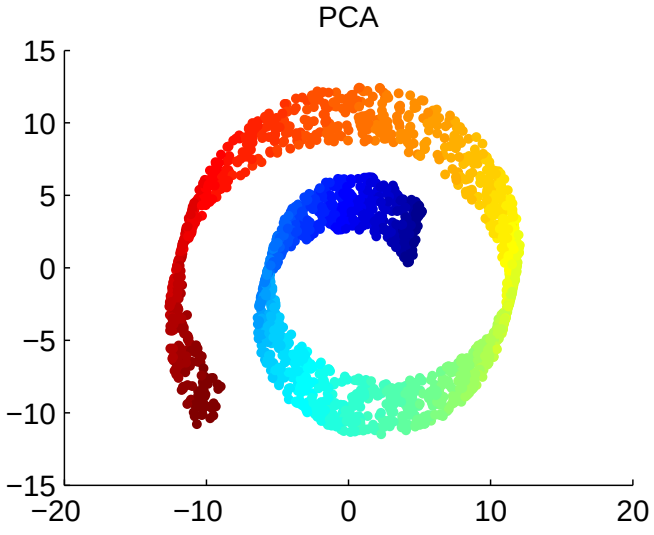
Constraint: The y_i's must satisfy certain properties

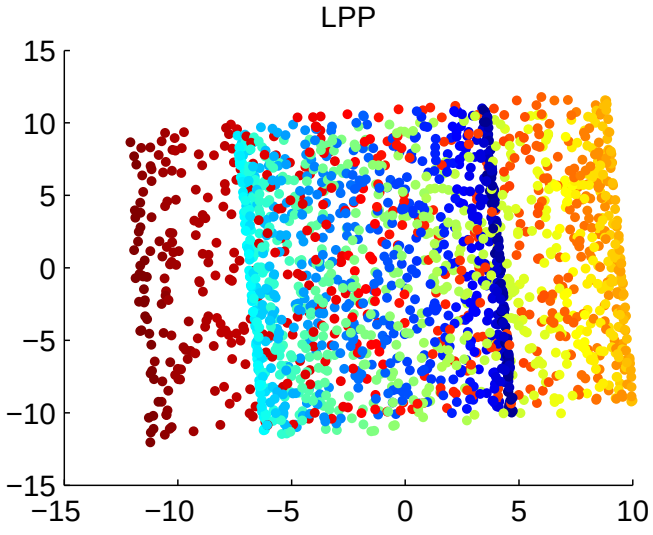
- Optimization problem

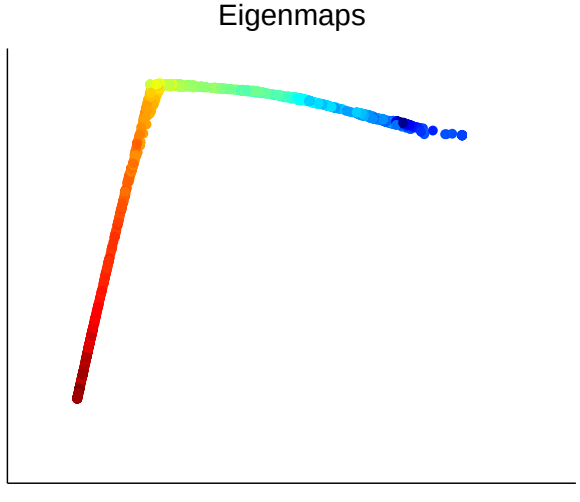
Example 1: The 'Swill-Roll' (2000 points in 3-D)

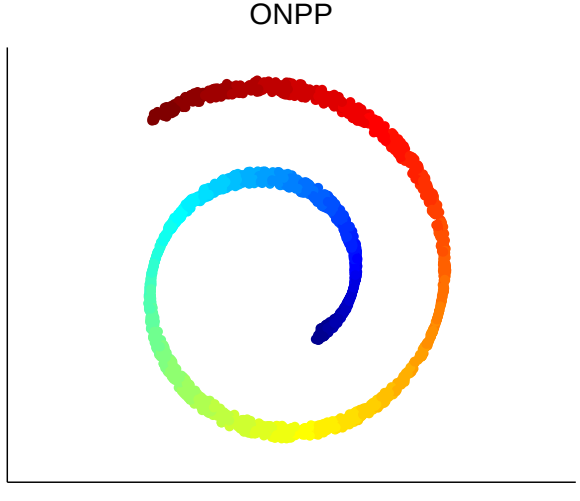


2-D ‘reductions’:

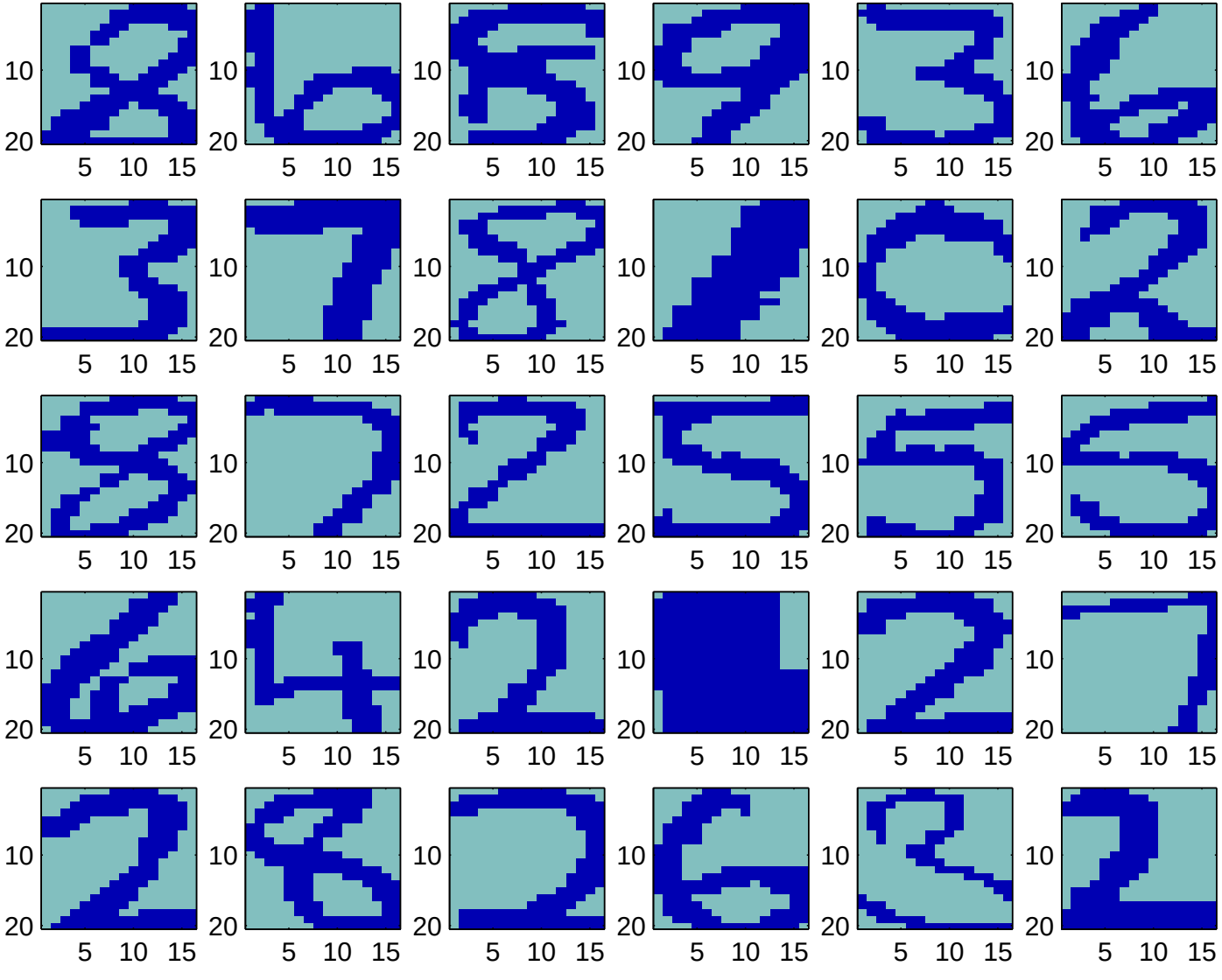




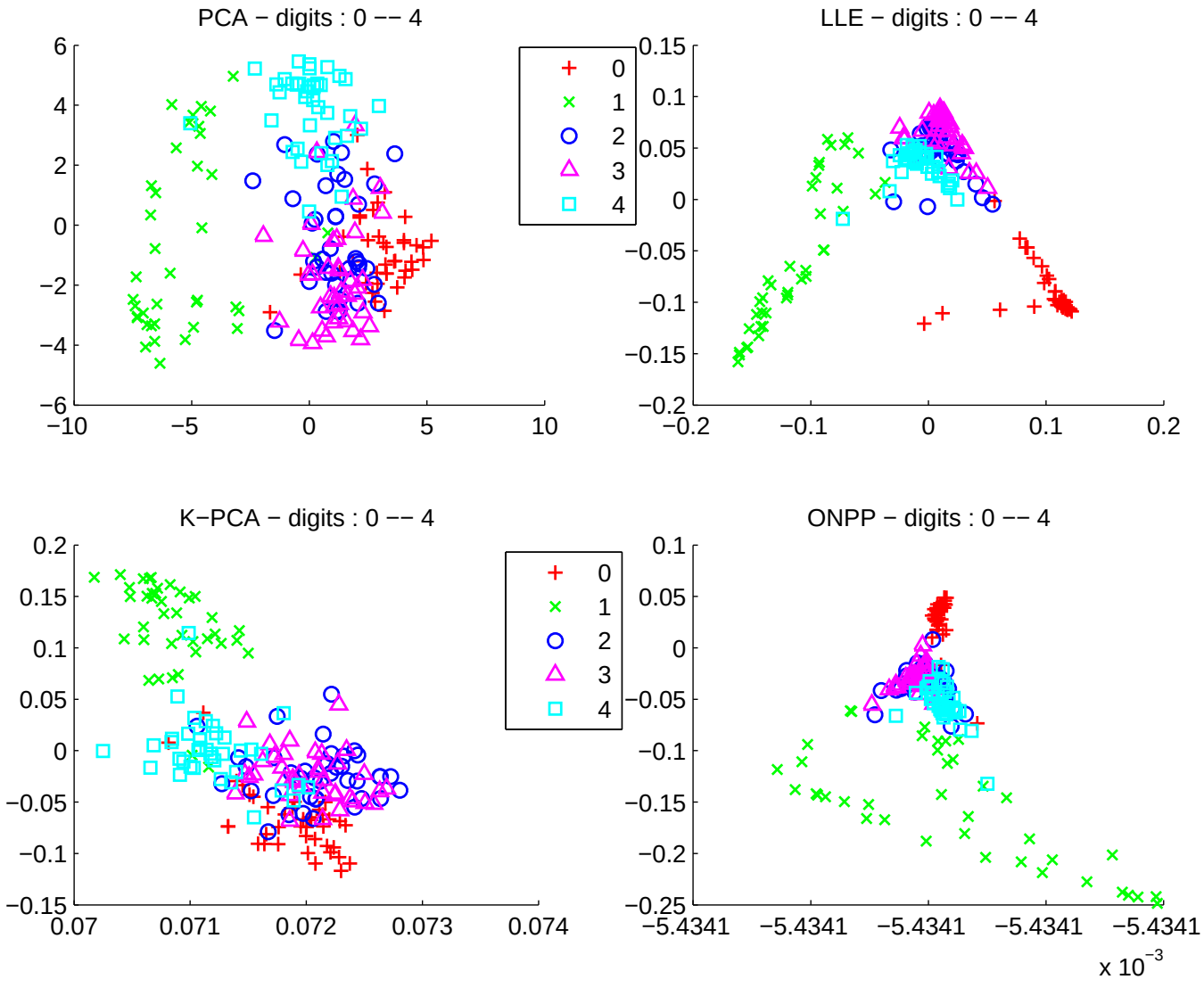




Example 2: Digit images (a random sample of 30)

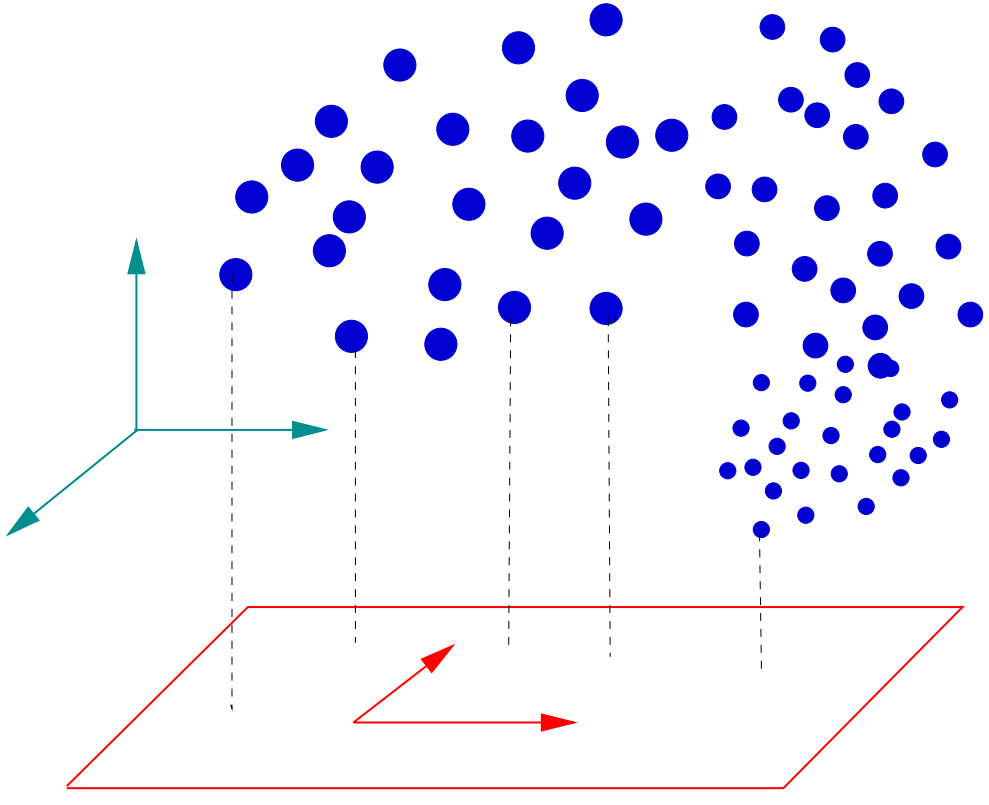


2-D 'reductions':



Basic linear dimensionality reduction method: PCA

- We are given points in $\mathbb{R}^n$ and we want to project them in $\mathbb{R}^d$. Best way to do this?
- i.e.: find the best axes for projecting the data
- Q: "best in what sense"?
- A: maximize variance of new data



- Principal Component Analysis [PCA]

- Recall $y_i = V^T x_i$, where V is $m \times d$ orthogonal
- We need to maximize over all orthogonal $m \times d$ matrices V:

$$\boxed{\sum_i \|y_i - \frac{1}{n}\sum_j y_j\|_2^2 = \cdots = \text{Tr}\left[V^\top \bar{X}\bar{X}^\top V\right]}$$

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

- Solution $V = \{$ dominant eigenvectors $\}$ of the covariance matrix == Set of left singular vectors of $\bar{X}$

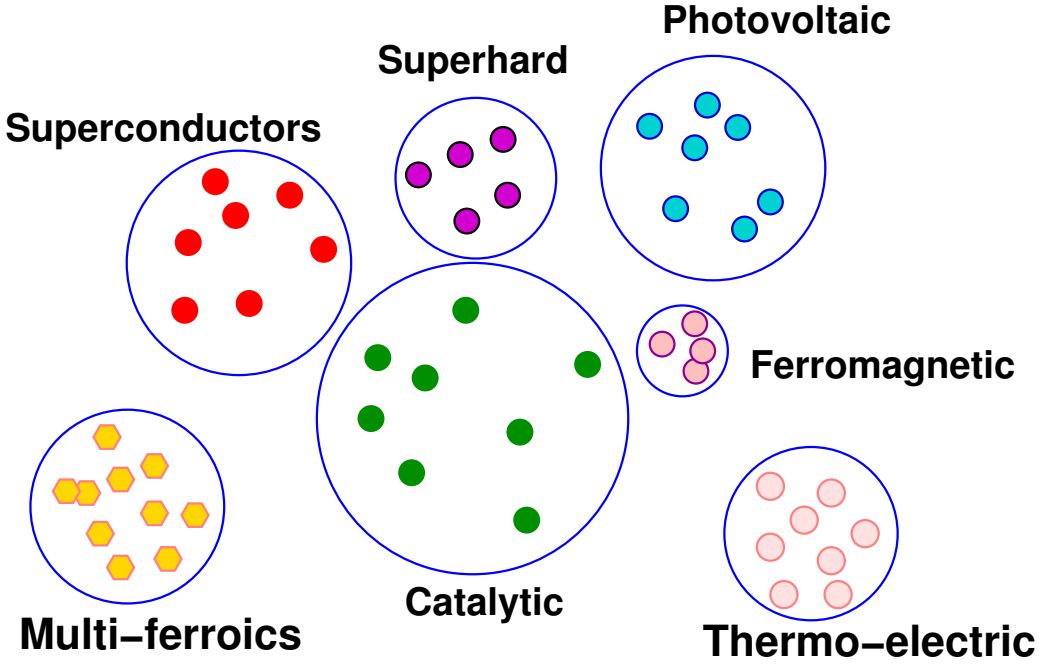
- Solution V also minimizes 'reconstruction error' ..

$$\sum_i \|x_i - VV^T x_i\|^2 = \sum_i \|x_i - Vy_i\|^2$$

- .. and it also maximizes [Korel and Carmel 04] $\sum_{i,j} \|y_i - y_j\|^2$

Unsupervised learning: Clustering

Problem: partition a given set into subsets such that items of the same subset are most similar and those of two different subsets most dissimilar.

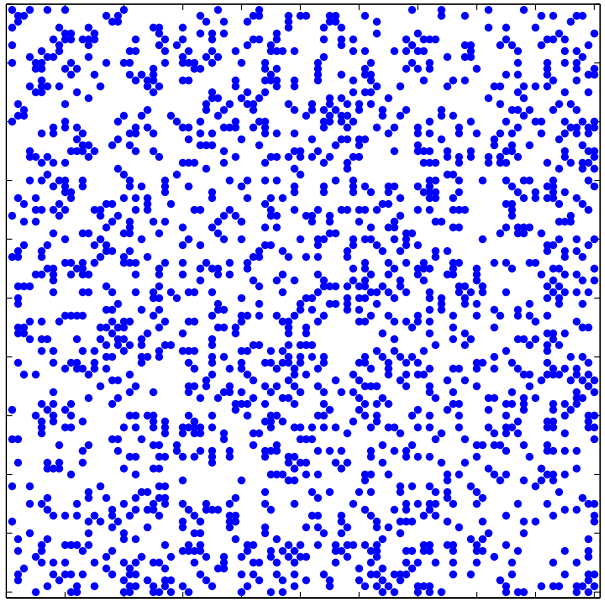


- Basic technique: K-means algorithm [slow but effective.]
- Example of application : cluster bloggers by 'social groups' (anarchists, ecologists, sports-fans, liberals-conservative, ...)

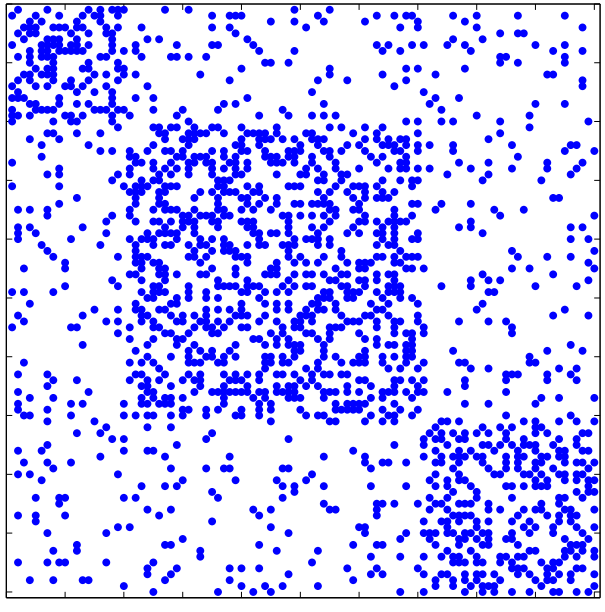
Sparse Matrices viewpoint

➤ Communities modeled by an 'affinity' graph [e.g., 'user A sends frequent e-mails to user B']

➤ Adjacency Graph represented by a sparse matrix



➤ Goal: find reordering such that blocks are as dense as possible:



➤ Advantage of this viewpoint: no need to know number of clusters.

➤ Use 'blocking' techniques for sparse matrices [O'Neil Szyld '90, YS 03, ...]

Example of application in knowledge discovery. Data set from :

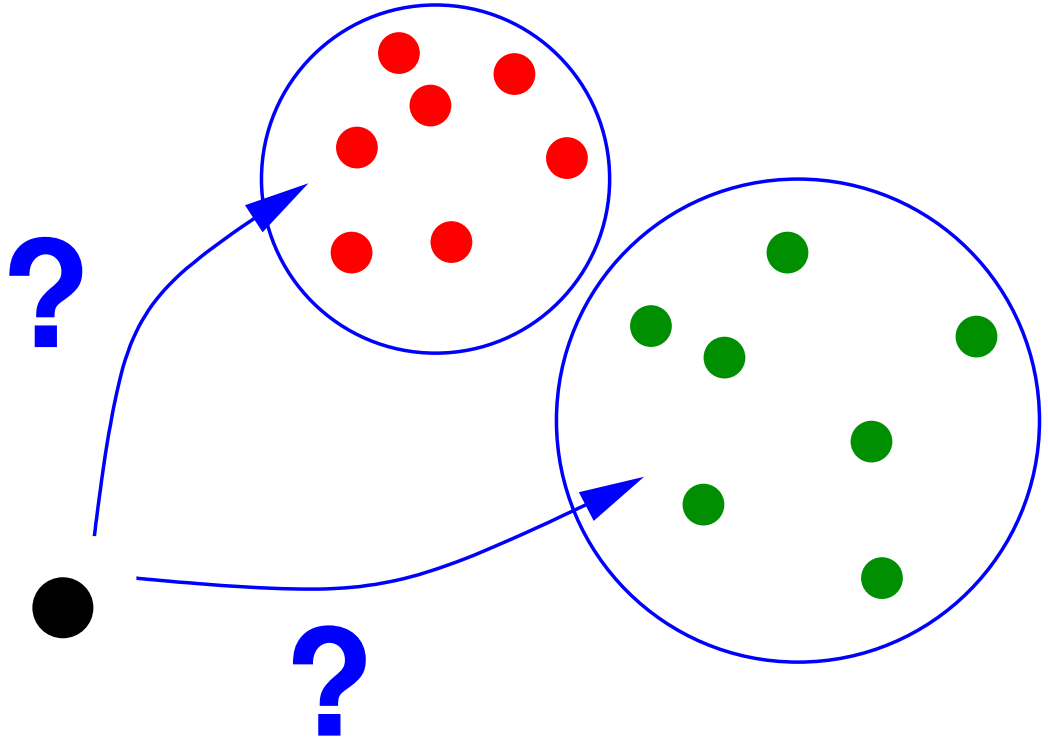
`http://www-personal.umich.edu/~mejn/netdata/`

- Network connecting bloggers of different political orientations [2004 US presidentual election]
- 'Communities': liberal vs. conservative
- Graph: $1,490$ vertices (blogs) : first 758: liberal, rest: conservative.
- Edge: $i \longrightarrow j$ means there is a citation between blogs i and j
- Blocking algorithm (Density theshold=0.4) found 2 subgraphs
- Note: density = $|E|/|V|^2$.
- Smaller subgraph: conservative blogs, larger one: liberals
- One observation: smaller subgraph is denser than that of the larger one. Concl. agrees with [L. A. Adamic and N. Glance 05] :

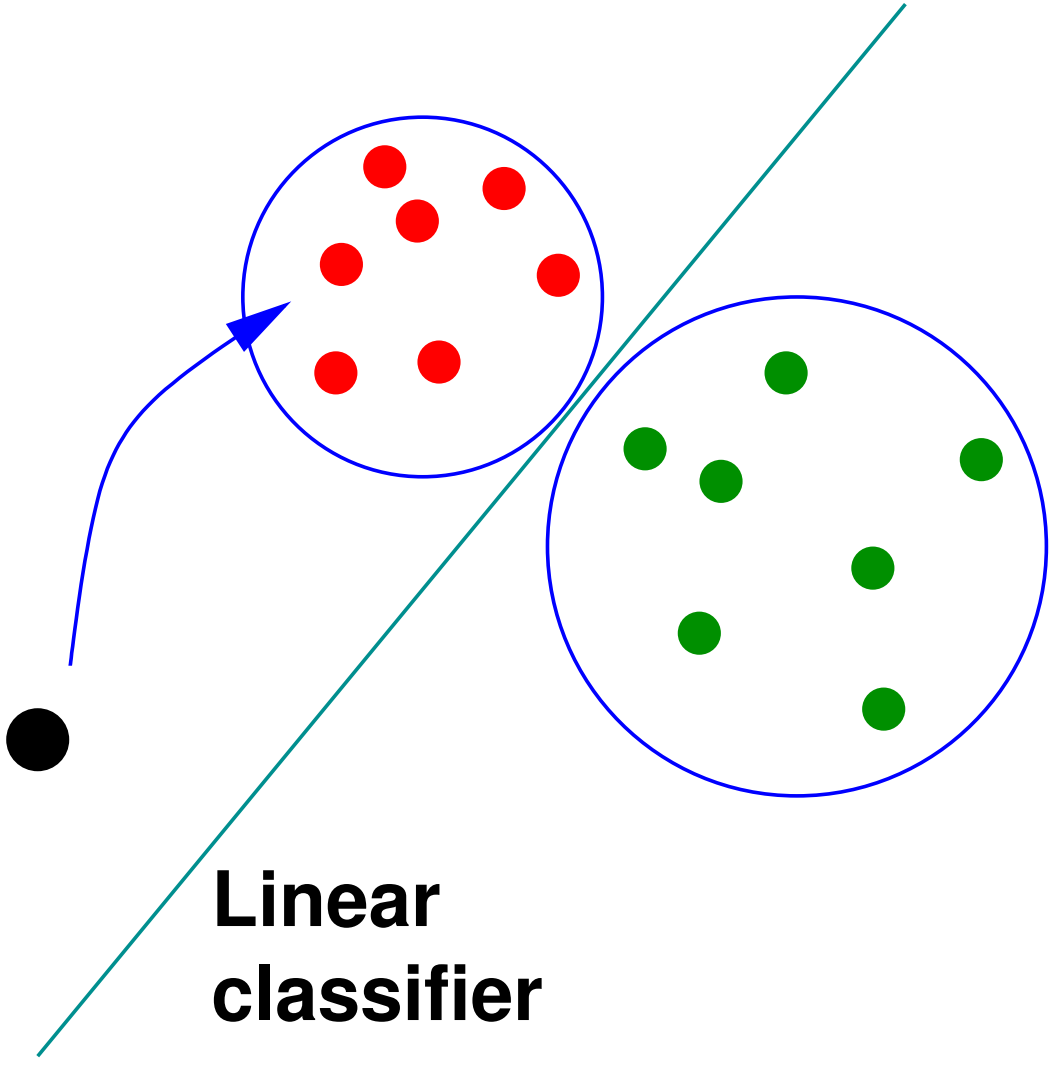
'right-leaning (conservative) blogs have a denser structure of strong connections than the left (liberal)"

Supervised learning: classification

Problem: Given labels (say “A” and “B”) for each item of a given set, find a way to classify an unlabelled item into either the “A” or the “B” class.

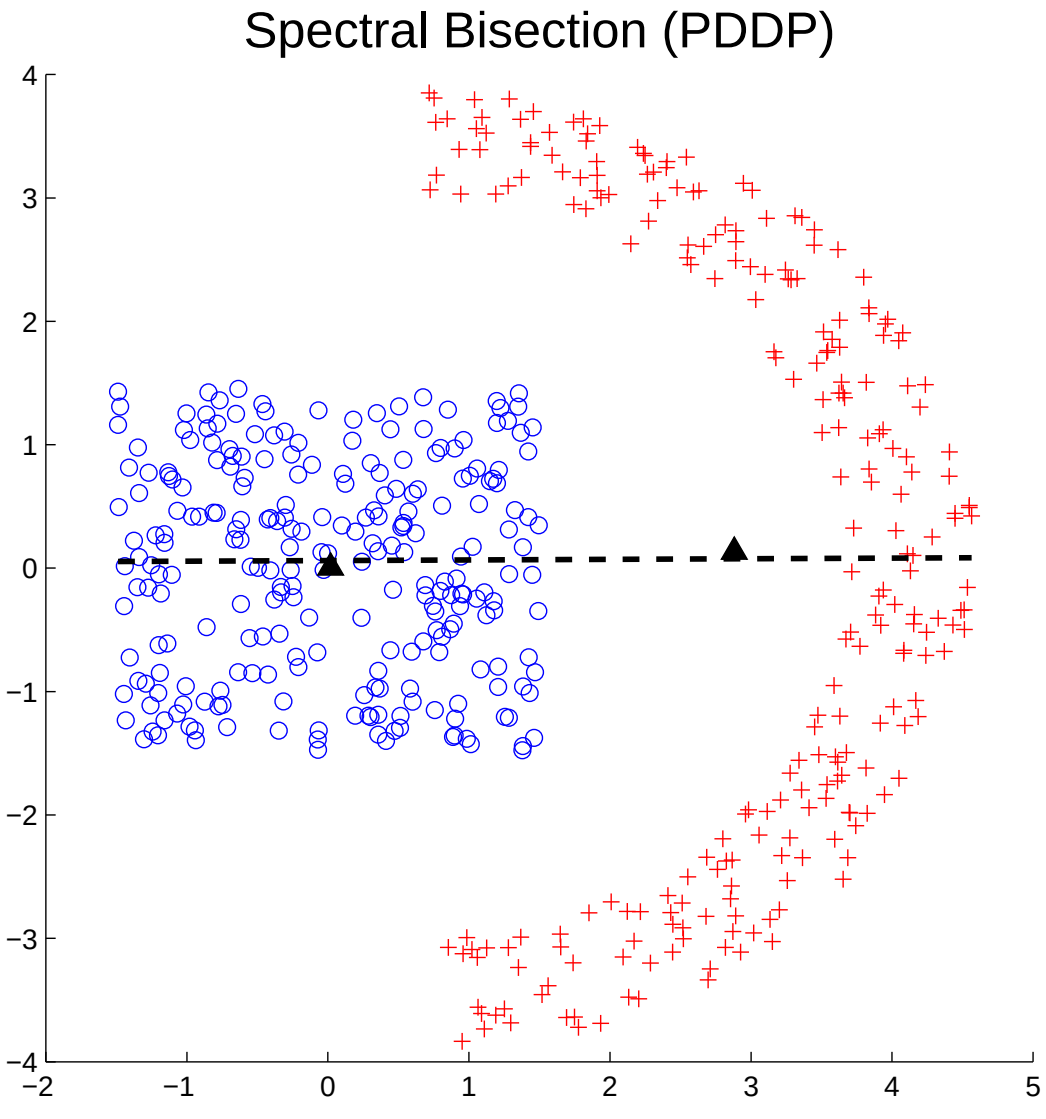


- Many applications.
- Example: distinguish between SPAM and non-SPAM messages
- Can be extended to more than 2 classes.



Linear classifiers: Find a hyperplane which best separates the data in classes A and B.

A harder case:



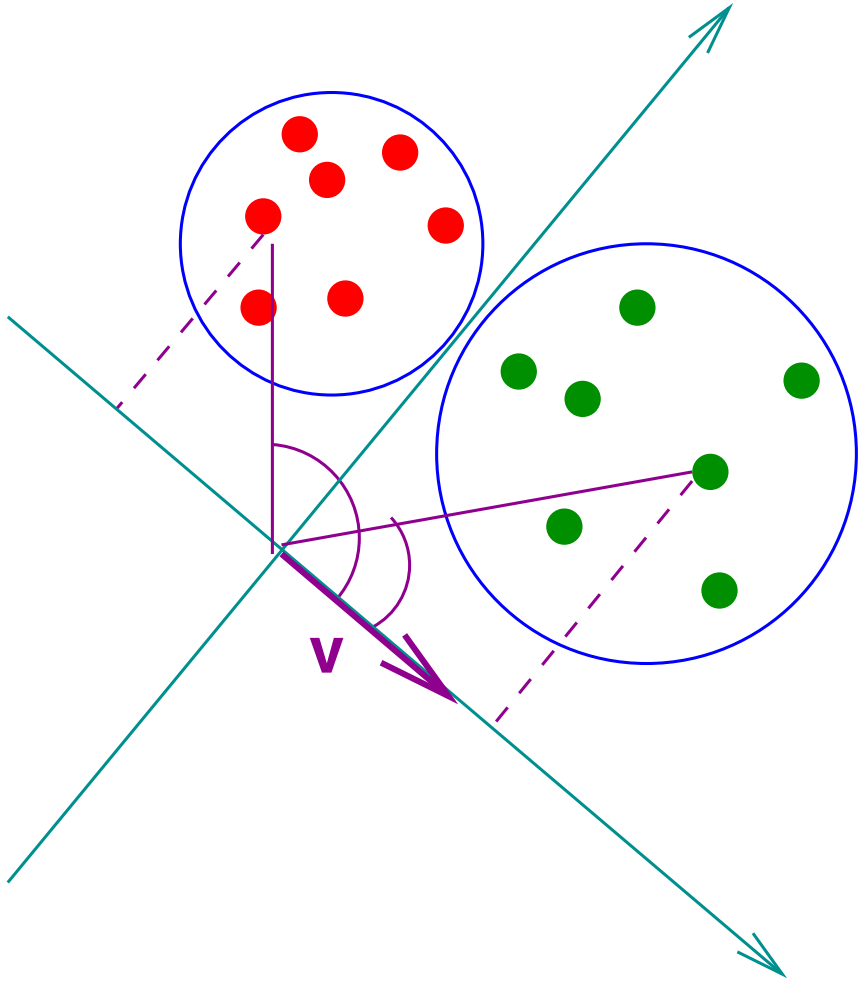
- Use kernels to transorm

Linear classifiers

- Let $X = [x_1, \cdots, x_n]$ be the data matrix.
- and $L = [l_1, \cdots, l_n]$ the labels either +1 or -1.

- First Solution: Find a vector v such that $v^T x_i$ close to l_i for all i.
- More common solution: Use SVD to reduce dimension of data [e.g. 2-D] then do comparison in this space. e.g. A: $v^T x_i \geq 0$, B: $v^T x_i < 0$

[Note: for clarity v principal axis drawn below where it should be]



Better solution: Linear Discriminant Analysis (LDA)

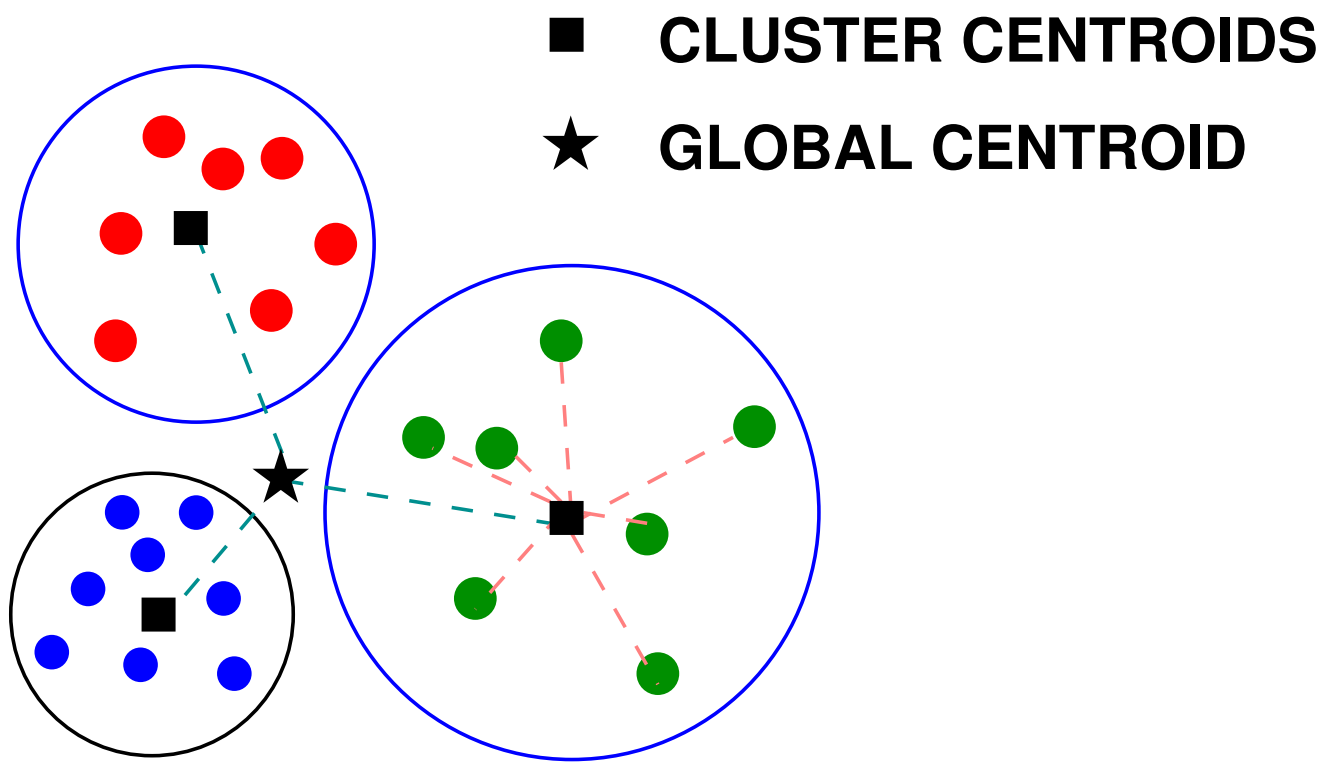
Define "between scatter": a measure of how well separated two distinct classes are.

Define "within scatter": a measure of how well clustered items of the same class are.

➤ Goal: to make "between scatter" measure large, while making "within scatter" small.

Idea: Project the data in low-dimensional space so as to maximize the ratio of the "between scatter" measure over "within scatter" measure of the classes.

- Let $\boldsymbol{\mu} =$ mean of $\boldsymbol{X}$,
- and $\boldsymbol{\mu}^{(k)}$ = mean of the $\boldsymbol{k}$-th class (of size $\boldsymbol{n_k}$).



Define

$$S_B = \sum_{k=1}^{c} n_k (\mu^{(k)} - \mu)(\mu^{(k)} - \mu)^T,$$

$$S_W = \sum_{k=1}^{c} \sum_{x_i \in X_k} (x_i - \mu^{(k)})(x_i - \mu^{(k)})^T.$$

- Project set on a one-dimensional space spanned by a vector a. Then:

$$a^T S_B a = \sum_{i=1}^{c} n_k |a^T(\mu^{(k)} - \mu)|^2, \quad a^T S_W a = \sum_{k=1}^{c} \sum_{x_i \in X_k} |a^T(x_i - \mu^{(k)})|^2$$

- LDA projects the data so as to maximize the ratio of these two numbers:

$$\max_a \frac{a^T S_B a}{a^T S_W a}$$

- Optimal a = eigenvector associated with the largest eigenvalue of:

$$S_B u_i = \lambda_i S_W u_i \ .$$

LDA – Extension to arbitrary dimension

- Would like to project in d dimensions –
- Normally we wish to maximize the ratio of two traces

$$\frac{\mathrm{Tr}\,[U^T S_B U]}{\mathrm{Tr}\,[U^T S_W U]}$$

- U subject to being unitary $U^T U = I$ (orthogonal projector).
- Reduced dimension data: $Y = U^T X$.

Difficulty: Hard to maximize. See Bellalij & YS (in progress).

- Common alternative: Solve instead the (easier) problem:

$$\max_{U^T S_W U = I} \mathrm{Tr}\,[U^T S_B U]$$

- Solution: largest eigenvectors of

$$S_B u_i = \lambda_i S_W u_i \,.$$

Motivating example: Collaborative filtering, Netflix Prize

- Important application in commerce ..
- .. but potential for other areas too
- When you order a book in Amazon you will get recommendations

'Recommender system'

- A very hot topic at the height of the dot-com era ...
- Some of the tools used: statistics, clustering, kNN graphs, LSI (a form of PCA), ...
- Visit the page GoupLens page *http://www.grouplens.org/*
- Netflix: *"qui veut gagner un million?"* [who wants to win a $ million?]

Applications Places System Tue May 19, 7:31 AM

Netflix Prize: Home - Firefox

File Edit View Go Bookmarks Tools Help

http://www.netflixprize.com//index

Go

NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Welcome!

The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love.

Read the Rules to see what is required to win the Prizes. If you are interested in joining the quest, you should register a team.

You should also read the frequently-asked questions about the Prize. And check out how various teams are doing on the Leaderboard.

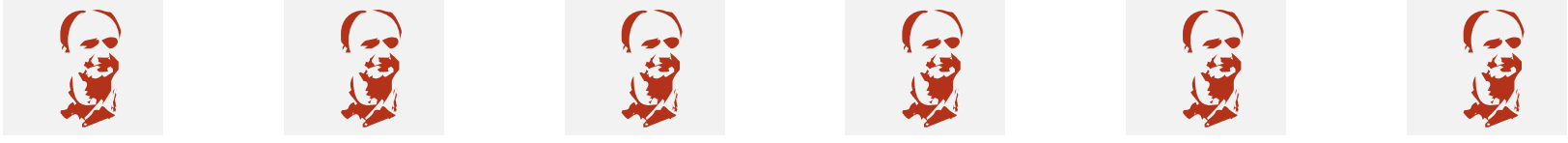
http://www.netflix.com/

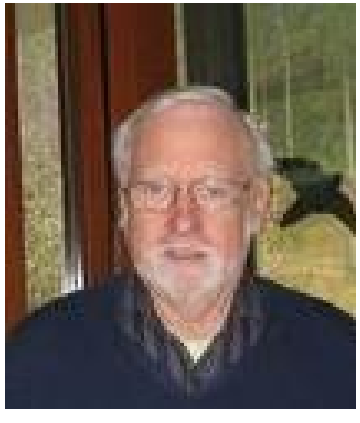
Terminal | Network settings | Netflix Prize: Ho... | Starting Take S...

Face Recognition – background

Problem: We are given a database of images: [arrays of pixel values].

And a test (new) image.





Question: Does this new image correspond to one of those in the database?

Difficulty

- Different positions, expressions, lighting, ..., situations :

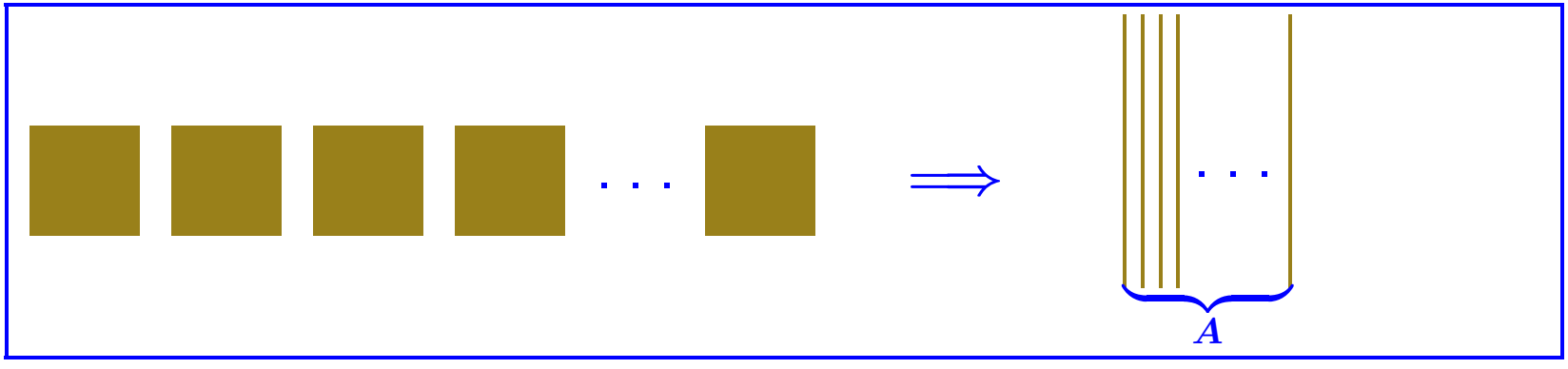
Common approach: eigenfaces – Principal Component Analysis technique

- See real-life examples – [international man-hunt]
- Poor images or deliberately altered images ['occlusion']
- See recent paper by John Wright et al.

Eigenfaces

– Consider each picture as a one-dimensional colum of all pixels

– Put together into an array A of size $\#\_pixels \times \#\_images$.



– Do an SVD of A and perform comparison with any test image in low-dim. space

– Similar to LSI in spirit – but data is not sparse.

Idea: replace SVD by Lanczos vectors (same as for IR)

Tests: Face Recognition

Tests with 2 well-known data sets:

ORL 40 subjects, 10 sample images each – example:

# of pixels : 112×92 TOT. # images : 400

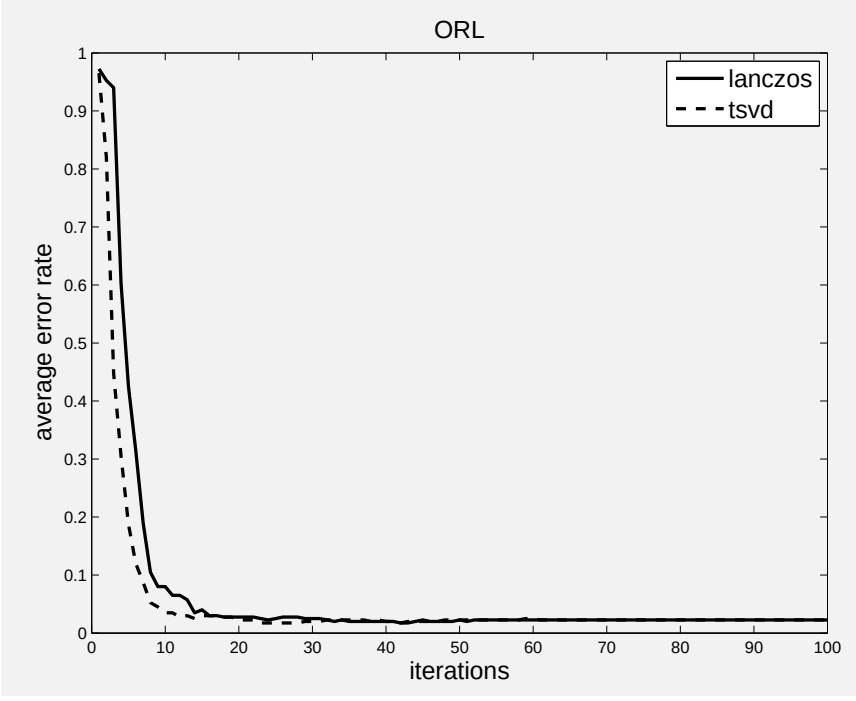
AR set 126 subjects – 4 facial expressions selected for each [natural, smiling, angry, screaming] – example:

# of pixels : 112×92 # TOT. # images : 504

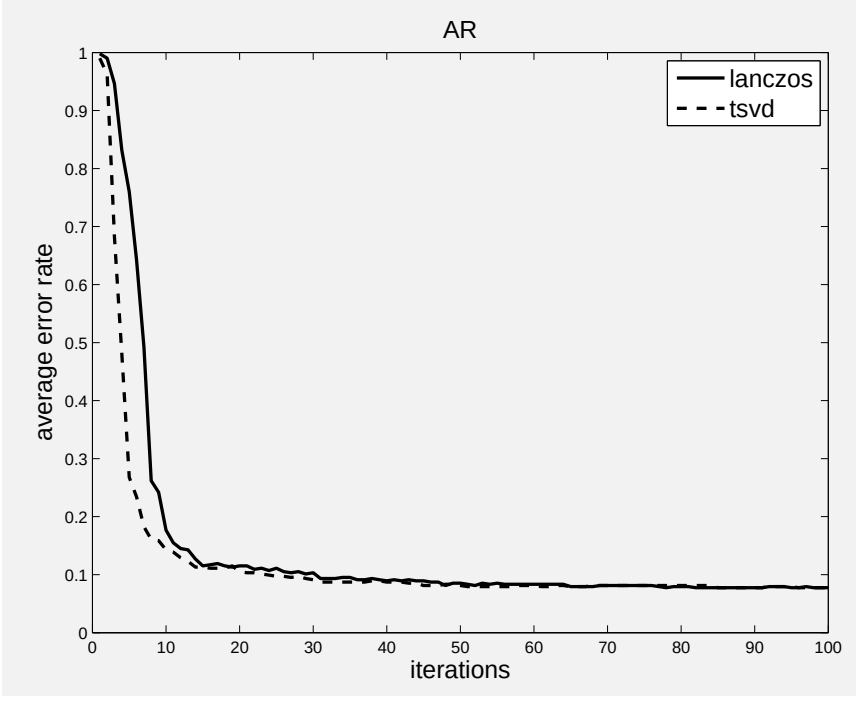
Tests: Face Recognition

Recognition accuracy of Lanczos approximation vs SVD

ORL dataset



AR dataset



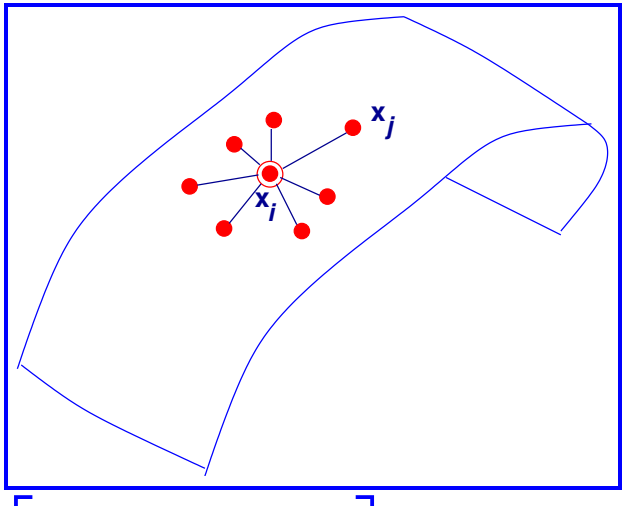
Vertical axis shows average error rate. Horizontal = Subspace dimension

GRAPH-BASED TECHNIQUES

Laplacean Eigenmaps (Belkin-Niyogi-02)

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

k-NN graph = graph of k nearest neighbors



- Then define a *graph Laplacean:*

$$L = D - W$$

- Simplest:

$$w_{ij} = \begin{cases} 1 \text{ if } j \in N_i \\ 0 \quad \text{else} \end{cases} \qquad D = \text{diag}\left[d_{ii} = \sum_{j \neq i} w_{ij} \right]$$

with N_i = neighborhood of i (excl. 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 $\boldsymbol{L = I - \frac{1}{n}11^\top}$

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

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

- Consequence-1: PCA equivalent to maximizing $\sum_{ij} \|\boldsymbol{y_i - y_j}\|^2$
- Consequence-2: what about replacing trivial $\boldsymbol{L}$ with something else?

[viewpoint in Koren-Carmel'04]

Property 4: (Graph partitioning) If $\boldsymbol{x}$ is a vector of signs (± 1) then

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

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

- Consequence: Can be used for partitioning graphs, or 'clustering' [take $\boldsymbol{p = sign(u_2)}$, where $\boldsymbol{u_2}$ = 2nd smallest eigenvector..]

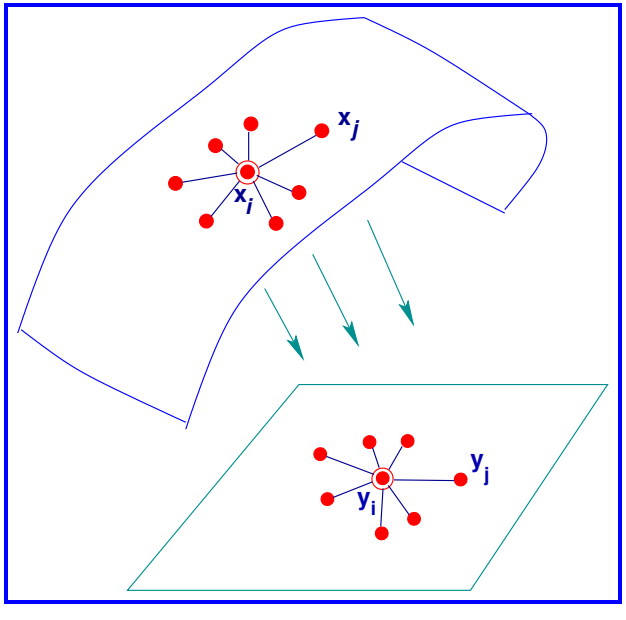
Return to Laplacean eigenmaps approach

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:

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

- Solution (sort eigenvalues increasingly):

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

- 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

A unified view

➤ Most techniques lead to one of two types of problems

First :

➤ $\boldsymbol{Y}$ obtained from solving an eigenvalue problem

➤ LLE, Eigenmaps (normalized), ..

$$\begin{cases} \min \\ \boldsymbol{Y} \in \mathbb{R}^{d \times n} \\ \boldsymbol{Y}\boldsymbol{Y}^\top = \boldsymbol{I} \end{cases} \textbf{Tr}\left[\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top\right]$$

Second:

➤ Low-Dimensional data: $\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}$

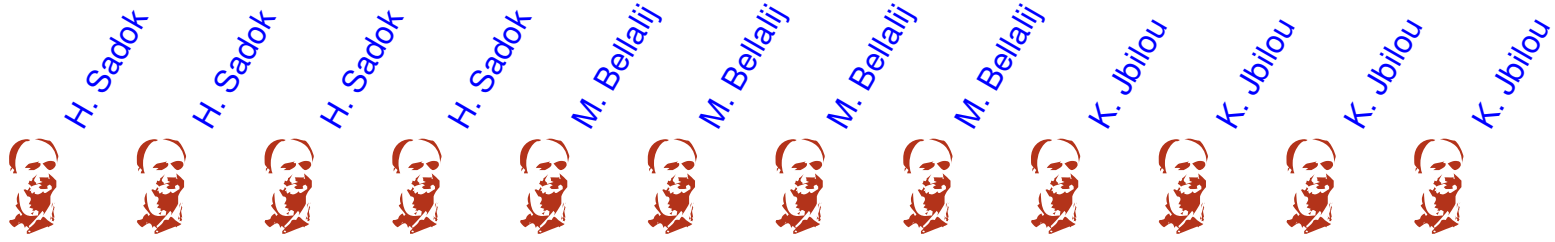
➤ $\boldsymbol{G}$ is either the identity matrix or $\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top$ or $\boldsymbol{X}\boldsymbol{X}^\top$.

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

Important observation: 2nd is just a projected version of the 1st.

Graph-based methods in a supervised setting

➤ Subjects of training set are known (labeled). Q: given a test image (say) find its label.



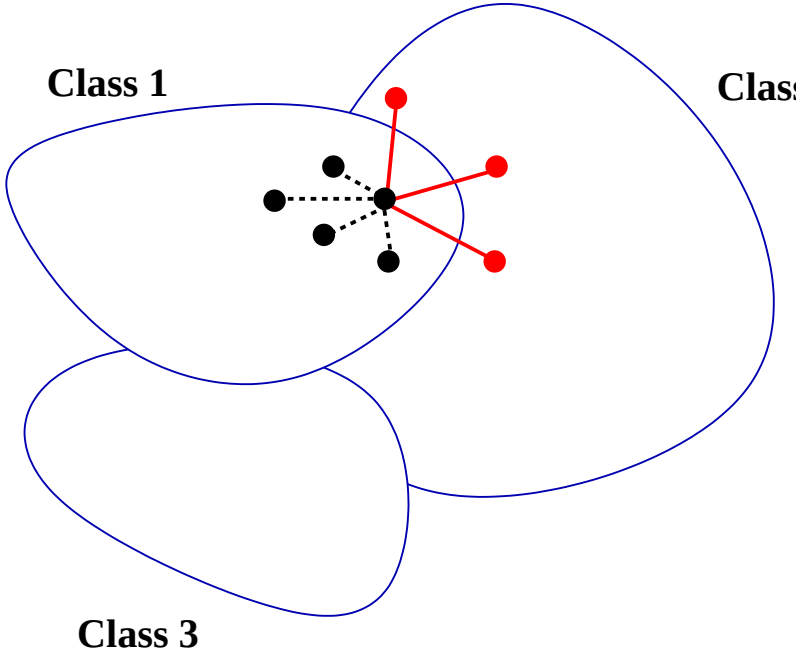
Question: Find label (best match) for test image.

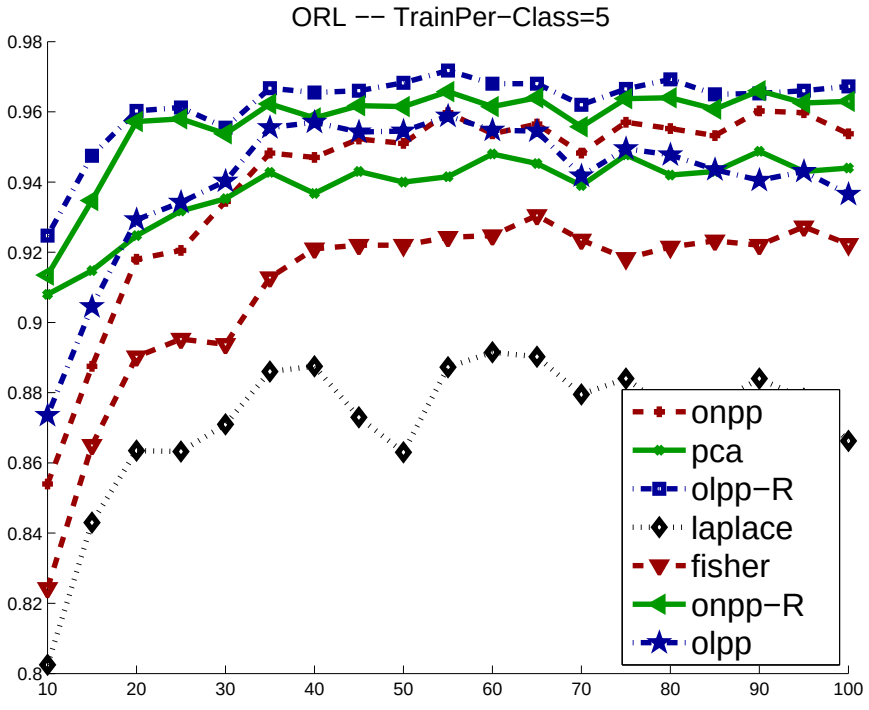
Methods can be adapted to supervised mode by building the graph to take into account class labels. Idea is simple: Build G so that nodes in the same class are neighbors. If c = # classes, G will consist of c cliques.

- Matrix W will be block-diagonal

$$W = \begin{pmatrix} W_1 & & & & \\ & W_2 & & & \\ & & W_3 & & \\ & & & W_4 & \\ & & & & W_5 \end{pmatrix}$$

- Easy to see that $\mathbf{rank}(W) = n - c$.
- Can be used for LPP, ONPP, etc..
- Recent improvement: add repulsion Laplacean [Kokiopoulou, YS 09]

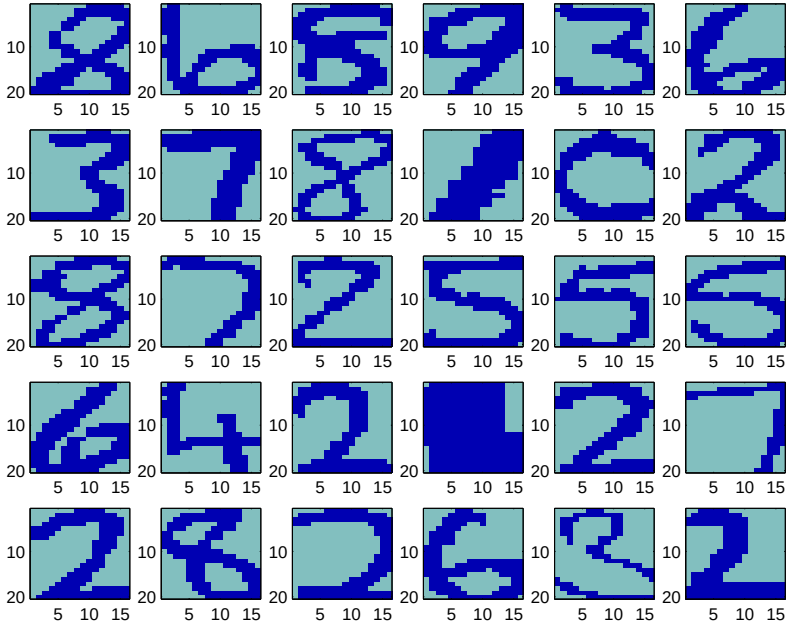

Class 1
Class 2
Class 3


ORL −− TrainPer−Class=5
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
10
20
30
40
50
60
70
80
90
100
onpp
pca
olpp−R
laplace
fisher
onpp−R
olpp

TIME FOR A MATLAB DEMO

Supervised learning experiments: digit recognition

- Set of 390 images of digits (39 of each digit)
- Each picture has $20 \times 16 = 320$ pixels.
- Select any one of the digits and try to recognize it among the 389 remaining images
- Methods: PCA, LPP, ONPP

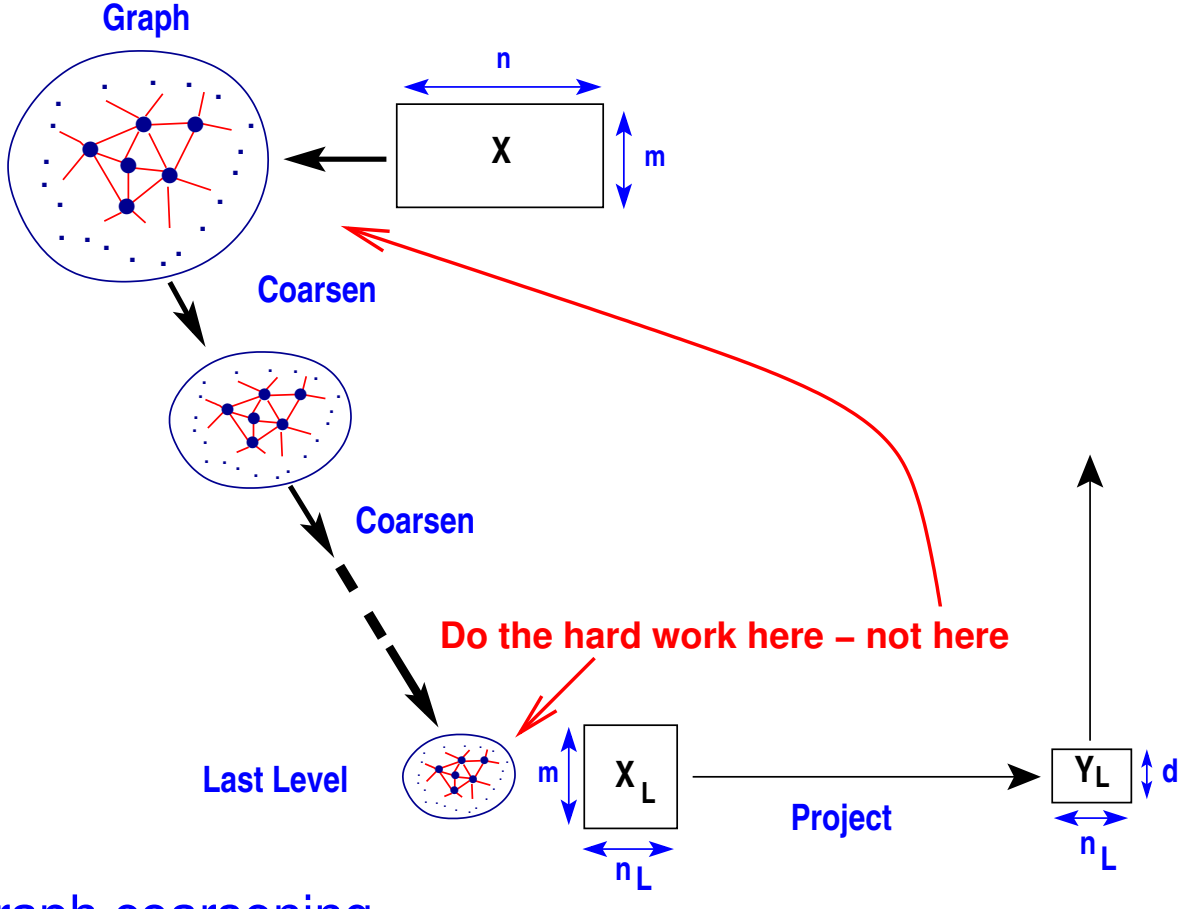


MULTILEVEL METHODS

Multilevel techniques

➤ 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?



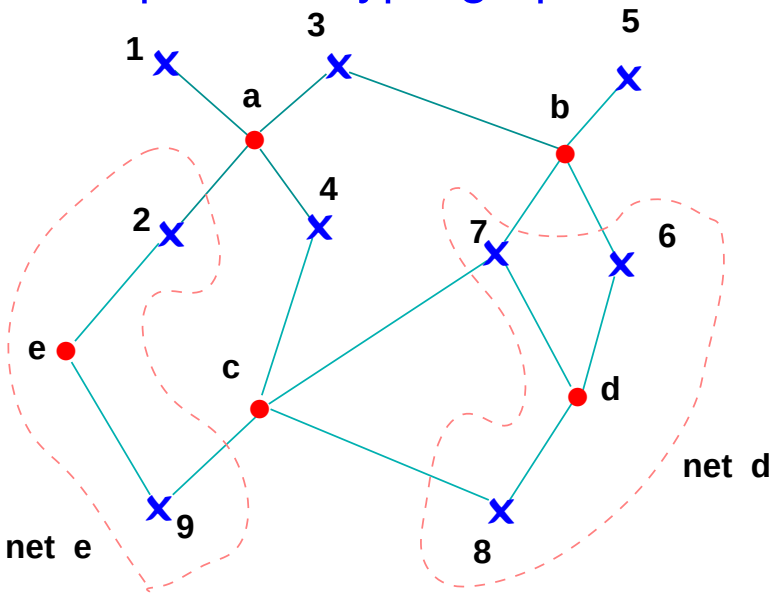
➤ Main tool used: graph coarsening

Hypergraphs and Hypergraph coarsening

A hypergraph $H = (V, E)$ is a generalizaition of a graph

- V = set of vertices V
- E = set of hyperedges. Each $e \in E$ is a nonempty subset of V
- Standard undirected graphs: each e consists of two vertices.
- *degree* of e = $|e|$
- *degree* of a vertex v = number of hyperedges e s.t. $x \in e$.
- Two vertices are *neighbors* if there is a hyperedge connecting them

Example of a hypergraph



Boolean matrix representation

$$A = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline a & 1 & 1 & 1 & 1 & & & & & \\ b & & & 1 & & 1 & 1 & 1 & & \\ c & & & & 1 & & & 1 & 1 & 1 \\ d & & & & & & 1 & 1 & 1 & \\ e & & 1 & & & & & & & 1 \end{array}$$

- Canonical hypergraph representation for sparse data (e.g. text)

Hypergraph Coarsening

➤ Coarsening a hypergraph $H = (V, E)$ means finding a 'coarse' approximation $\hat{H} = (\hat{V}, \hat{E})$ to H with $|\hat{V}| < |V|$, which tries to retain as much as possible of the structure of the original hypergraph

➤ Idea: repeat coarsening recursively.

➤ Result: succession of smaller hypergraphs which approximate the original graph.

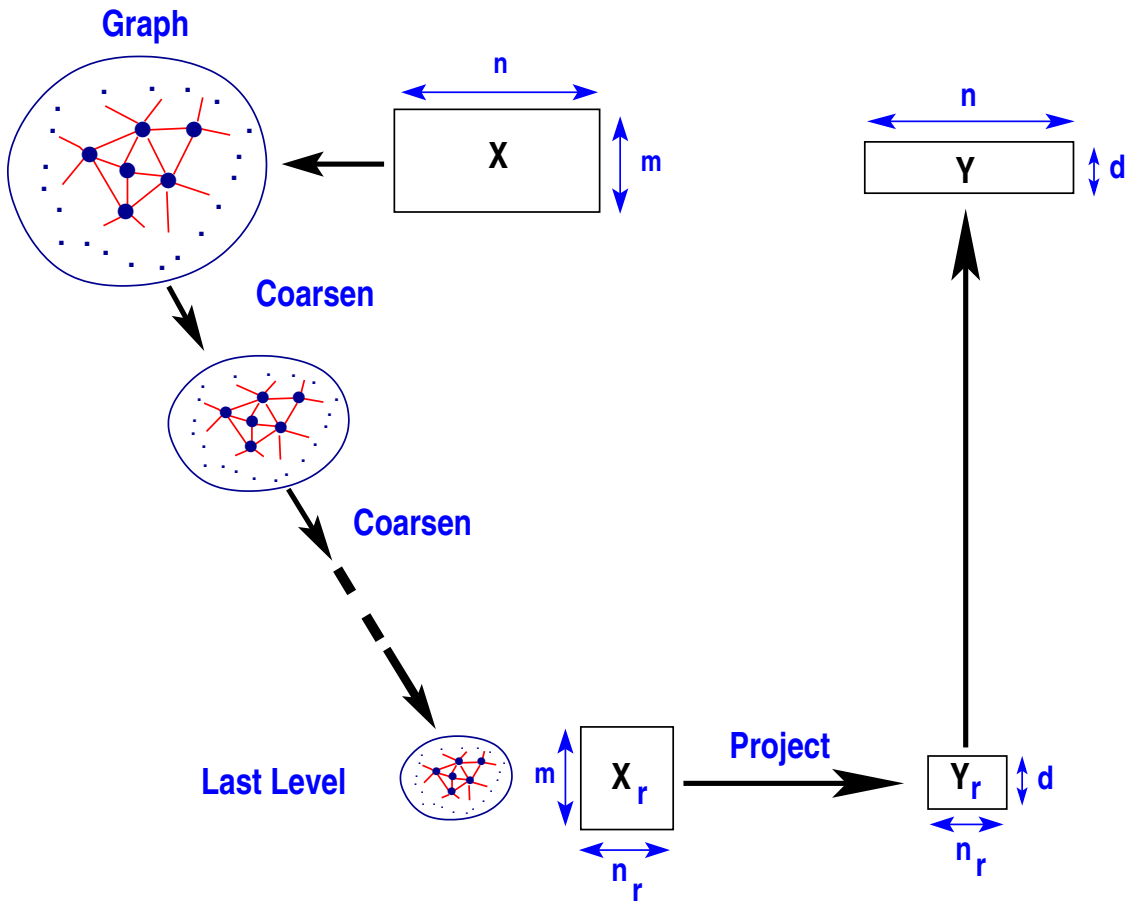
➤ Several methods exist. We use 'matching', which successively merges pairs of vertices

➤ Used in most graph partitioning methods: hMETIS, Patoh, zoltan, ..

➤ Algorithm successively selects pairs of vertices to merge – based on measure of similarity of the vertices.

Application: Multilevel Dimension Reduction

Main Idea: coarsen to a certain level. Then use the resulting data set $\hat{X}$ to find projector from $\mathbb{R}^m$ to $\mathbb{R}^d$. This projector can be used to project the original data or any new data.



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

Application to text mining

➤ Recall common approach:

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

where $[\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_n] = \boldsymbol{V_d^T} \in \mathbb{R}^{\boldsymbol{d \times n}}$; $\hat{\boldsymbol{q}} = \boldsymbol{\Sigma_d^{-1} U_d^T \bar{q}} \in \mathbb{R}^{\boldsymbol{d}}$

➤ 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)

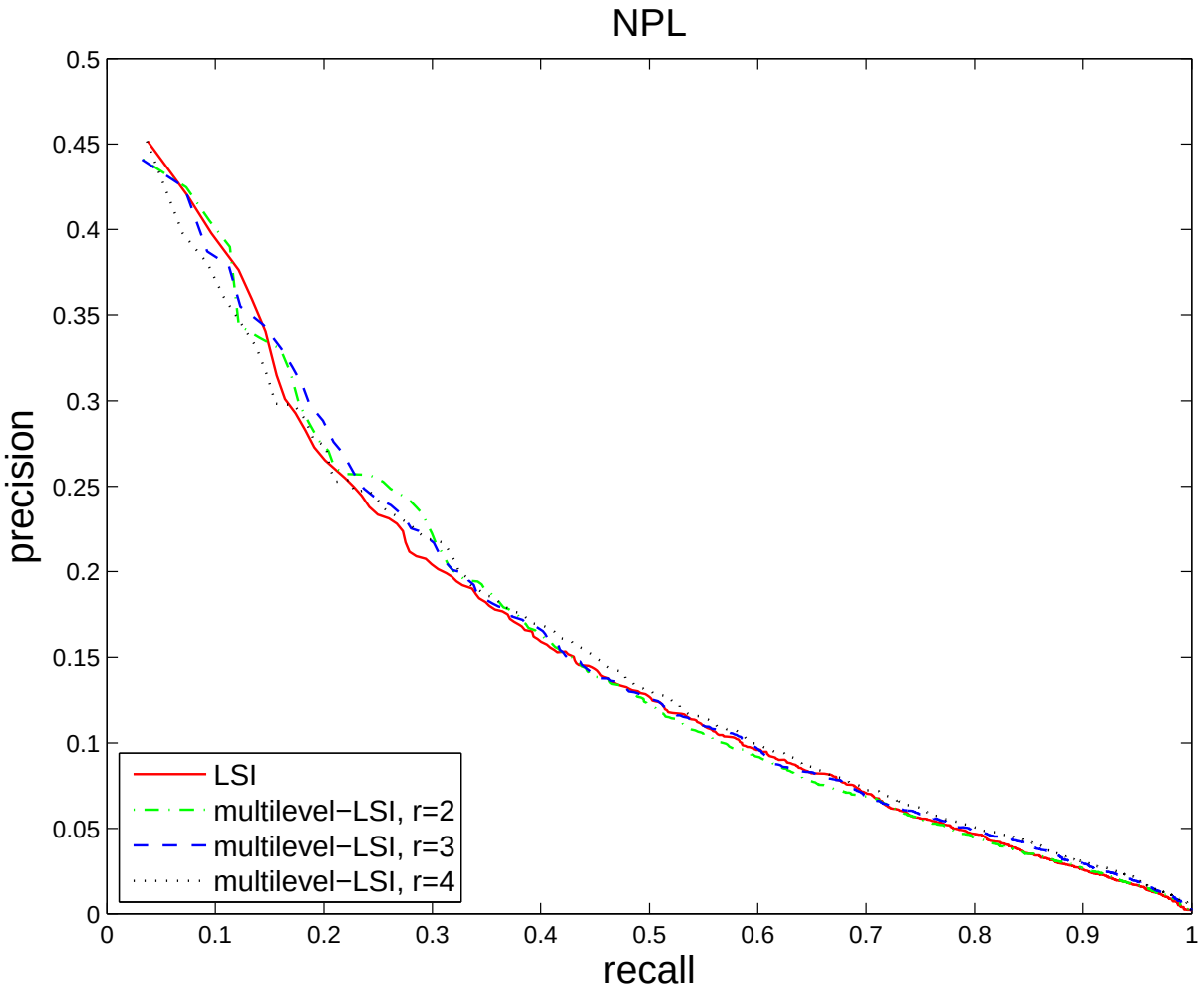
- Coarsening to a max. of 4 levels.

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

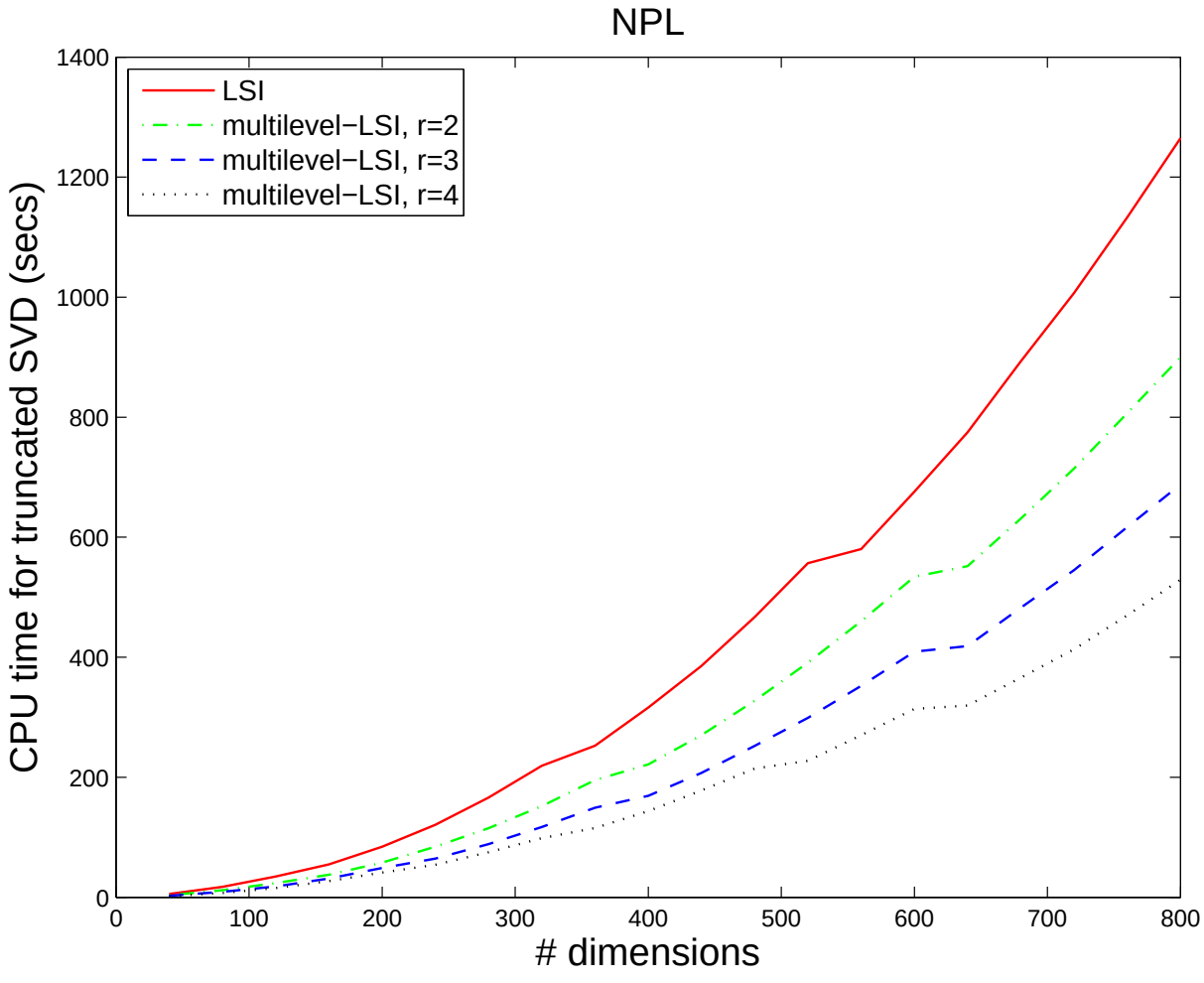
Statistics

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)



Conclusion

- Eigenvalue problems of data-mining 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]