UNIVERSITY OF MINNESOTA TWIN CITIES

Dimension reduction methods: Algorithms and Applications

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

NASCA – 2018, Kalamata

July 6, 2018

Introduction, background, and motivation

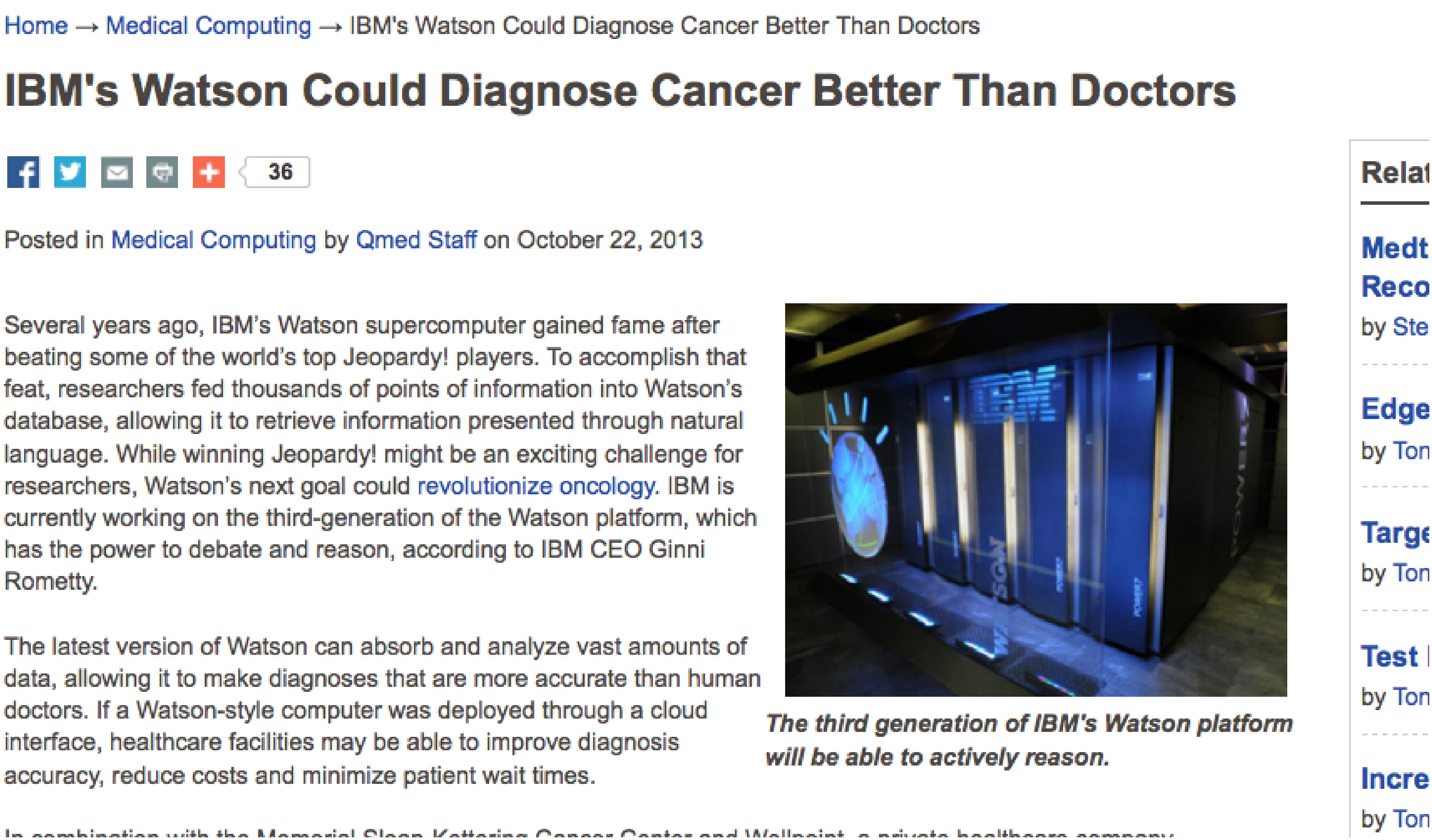
Common goal of data mining methods: to extract meaningful information or patterns from data. Very broad area – includes: data analysis, machine learning, pattern recognition, information retrieval, ...

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

Introduction: a few factoids

- ➤ We live in an era increasingly shaped by ‘DATA’
  - ● $\approx \mathbf{2.5 \times 10^{18}}$ bytes of data created in 2015
  - ● 90 % of data on internet created since 2016
  - ● 3.8 Billion internet users in 2017.
  - ● 3.6 Million Google searches worldwide / minute (5.2 B/day)
  - ● 15.2 Million text messages worldwide / minute
- ➤ Mixed blessing: Opportunities & big challenges.
- ➤ Trend is re-shaping & energizing many research areas ...
- ➤ ... including : numerical linear algebra

A huge potential: Health sciences



Home → Medical Computing → IBM's Watson Could Diagnose Cancer Better Than Doctors

IBM's Watson Could Diagnose Cancer Better Than Doctors

36

Posted in Medical Computing by Qmed Staff on October 22, 2013

Several years ago, IBM's Watson supercomputer gained fame after beating some of the world's top Jeopardy! players. To accomplish that feat, researchers fed thousands of points of information into Watson's database, allowing it to retrieve information presented through natural language. While winning Jeopardy! might be an exciting challenge for researchers, Watson's next goal could revolutionize oncology. IBM is currently working on the third-generation of the Watson platform, which has the power to debate and reason, according to IBM CEO Ginni Rometty.

The third generation of IBM's Watson platform will be able to actively reason.

The latest version of Watson can absorb and analyze vast amounts of data, allowing it to make diagnoses that are more accurate than human doctors. If a Watson-style computer was deployed through a cloud interface, healthcare facilities may be able to improve diagnosis accuracy, reduce costs and minimize patient wait times.

In combination with the Memorial Sloan Kettering Cancer Center and Wellpoint, a private healthcare company

Relat

Medt
Reco
by Ste

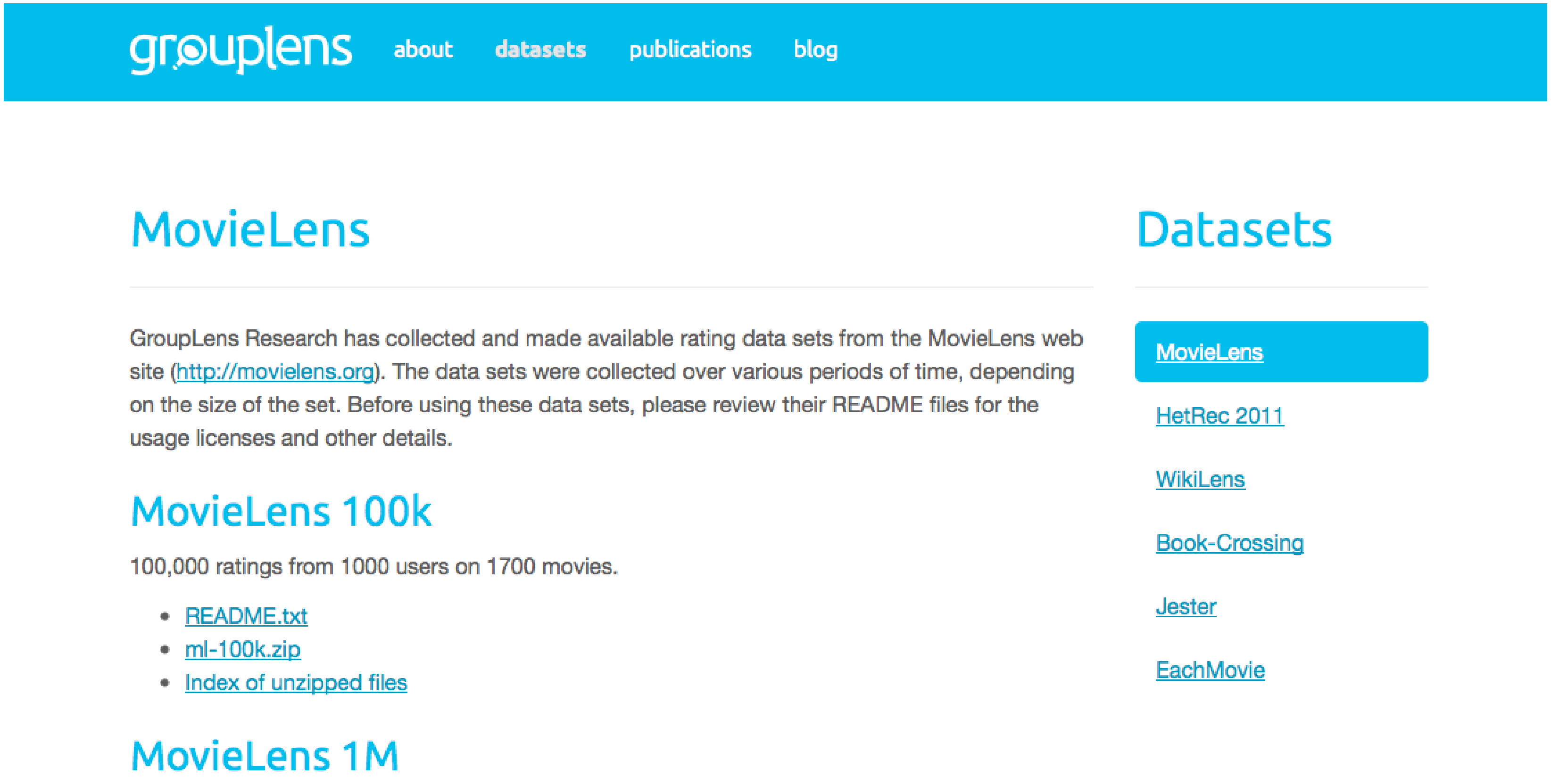
Edge
by Ton

Targe
by Ton

Test
by Ton

Incre
by Ton

Recommending books or movies: recommender systems



MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

MovieLens 100k

100,000 ratings from 1000 users on 1700 movies.

- README.txt
- ml-100k.zip
- Index of unzipped files

MovieLens 1M

Datasets

- MovieLens
- HetRec 2011
- WikiLens
- Book-Crossing
- Jester
- EachMovie

A few sample (classes of) problems:

- Classification: ’Benign – Malignant’, ’Dangerous-Safe’, Face recognition, digit recognition,

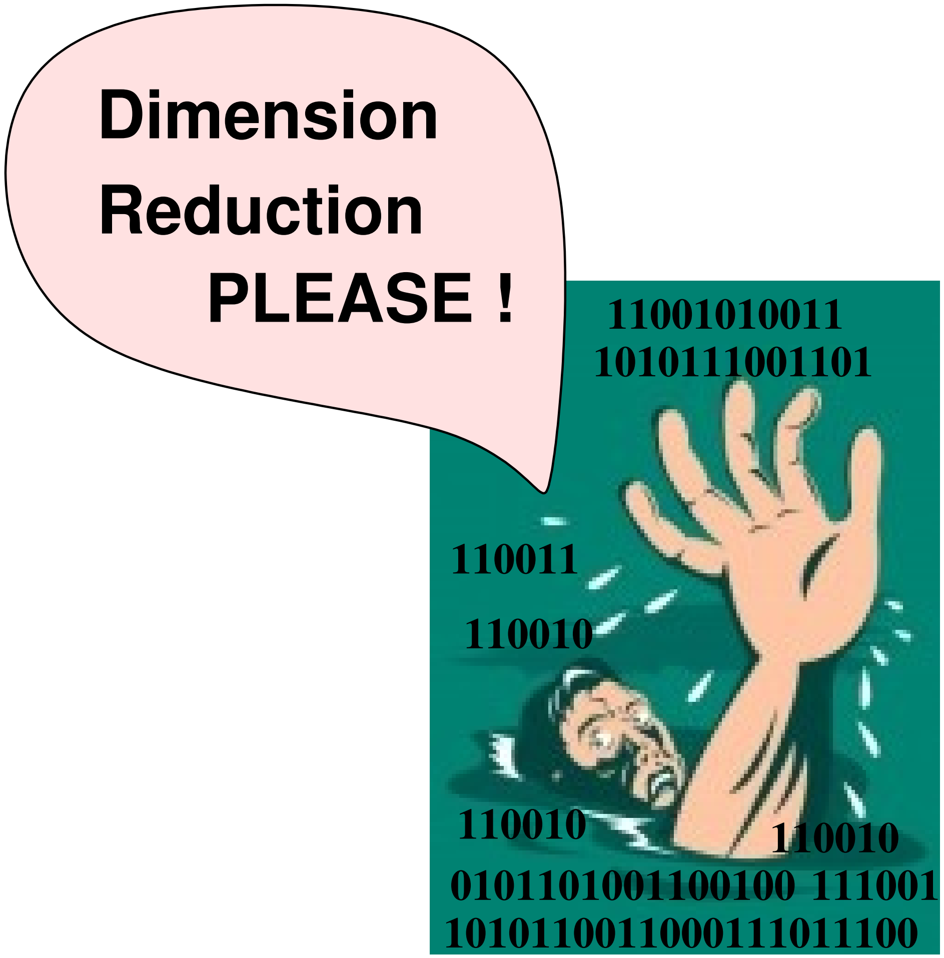
- Matrix completion: Recommender systems

- Projection type methods: PCA, LSI, Clustering, Eigenmaps, LLE, Isomap, ...

- Problems for graphs/ networks: Pagerank, analysis of graphs (node centrality, ...)

- Problems from computational statistics [Trace (inv(Cov)), Log det (A), Log Likelyhood, ...]

A common tool: Dimension reduction



Picture modified from `http://www.123rf.com/photo_7238007_man-drowning-reaching-out-for-help.html`

Major tool of Data Mining: Dimension reduction

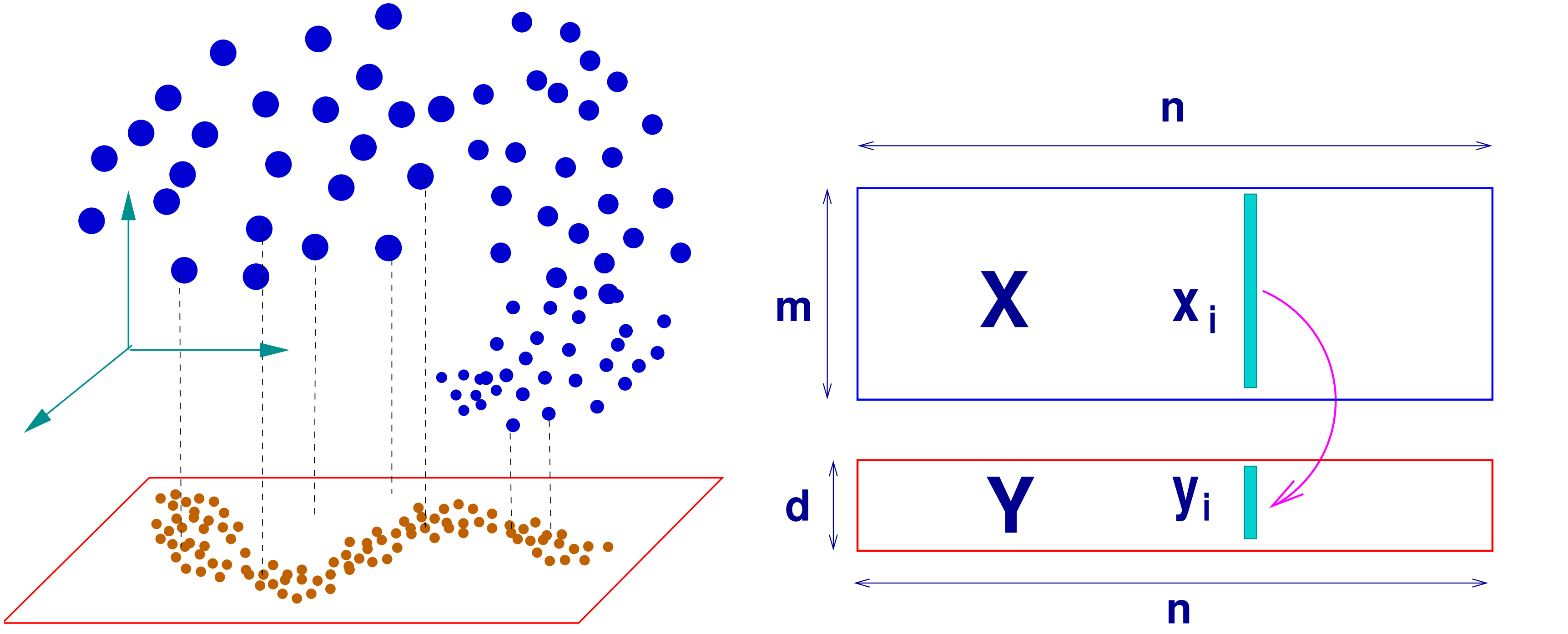
➤ Goal is not as much to reduce size (& cost) but to:

- Reduce noise and redundancy in data before performing a task [e.g., classification as in digit/face recognition]
- Discover important ‘features’ or ‘paramaters’

The problem: Given: $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$, find a low-dimens. representation $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n] \in \mathbb{R}^{d\times n}$ of $\boldsymbol{X}$

➤ Achieved by a mapping $\boldsymbol{\Phi} : \boldsymbol{x} \in \mathbb{R}^m \longrightarrow \boldsymbol{y} \in \mathbb{R}^d$ so:

$$\phi(\boldsymbol{x}_i) = \boldsymbol{y}_i, \quad \boldsymbol{i} = 1, \cdots, \boldsymbol{n}$$



- Φ may be linear : $y_j = W^\top x_j,\ \forall j$, or, $Y = W^\top X$

- ... or nonlinear (implicit).

- Mapping Φ required to: Preserve proximity? Maximize variance? Preserve a certain graph?

Basics: Principal Component Analysis (PCA)

In *Principal Component Analysis* W is computed to maximize variance of projected data:

$$\max_{W \in \mathbb{R}^{m\times d}; W^\top W = I} \quad \sum_{i=1}^{n} \left\| y_i - \frac{1}{n}\sum_{j=1}^{n} y_j \right\|_2^2, \quad y_i = W^\top x_i.$$

- Leads to maximizing

$$\text{Tr}\left[W^\top (X - \mu e^\top)(X - \mu e^\top)^\top W\right], \quad \mu = \frac{1}{n}\Sigma_{i=1}^{n} x_i$$

- Solution $W = \{$ dominant eigenvectors $\}$ of the covariance matrix $\equiv$ Set of left singular vectors of $\bar{X} = X - \mu e^\top$

SVD:

$$\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \quad \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}, \quad \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}, \quad \boldsymbol{\Sigma} = \text{Diag}$$

- Optimal $\boldsymbol{W} = \boldsymbol{U}_d \equiv$ matrix of first $\boldsymbol{d}$ columns of $\boldsymbol{U}$
- Solution $\boldsymbol{W}$ also minimizes 'reconstruction error' ..

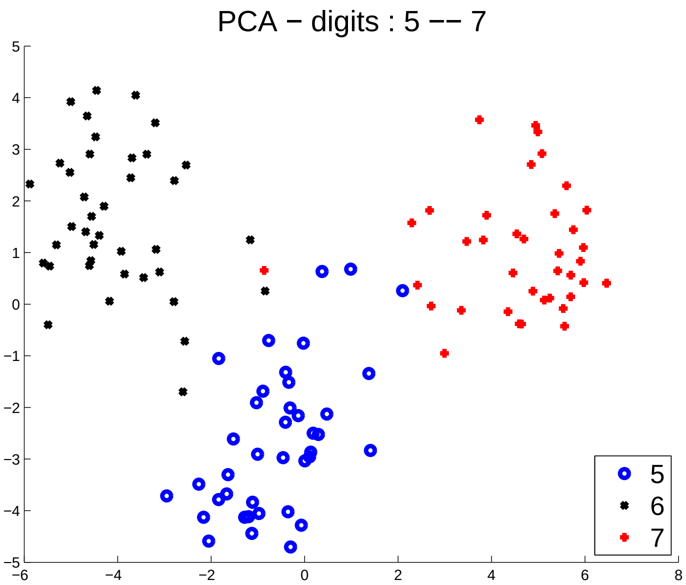
$$\sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i\|^2 = \sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{y}_i\|^2$$

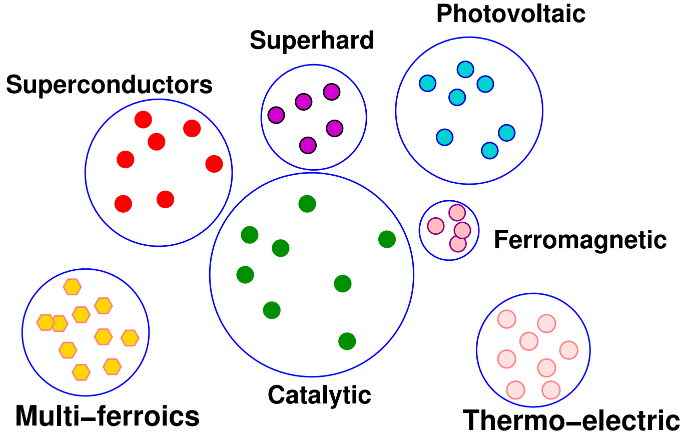
- In some methods recentering to zero is not done, i.e., $\bar{\boldsymbol{X}}$ replaced by $\boldsymbol{X}$.

Unsupervised learning

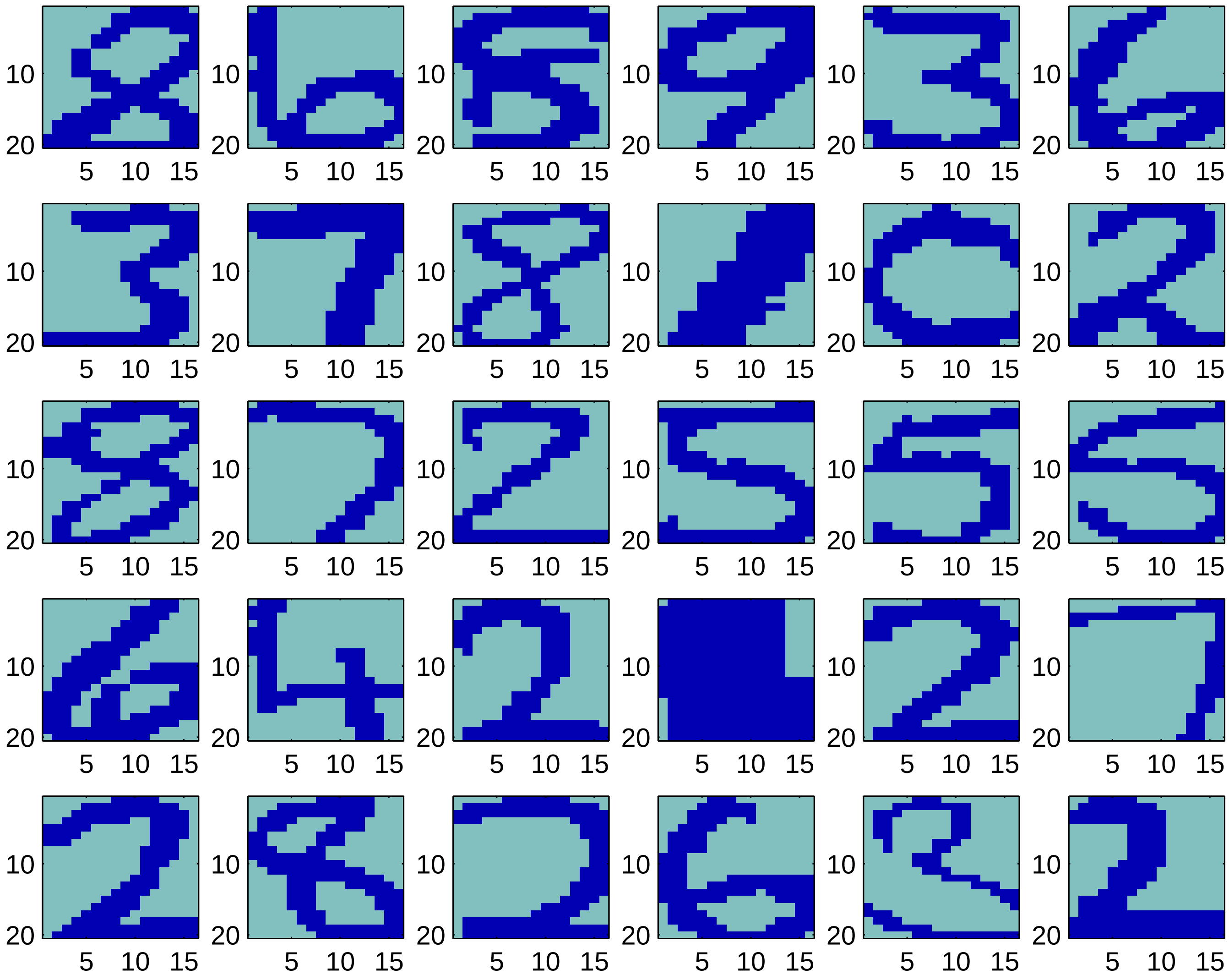
"Unsupervised learning" : methods do not exploit labeled data

- ➤ Example of digits: perform a 2-D projection
- ➤ Images of same digit tend to cluster (more or less)
- ➤ Such 2-D representations are popular for visualization
- ➤ Can also try to find natural clusters in data, e.g., in materials
- ➤ Basic clusterning technique: K-means

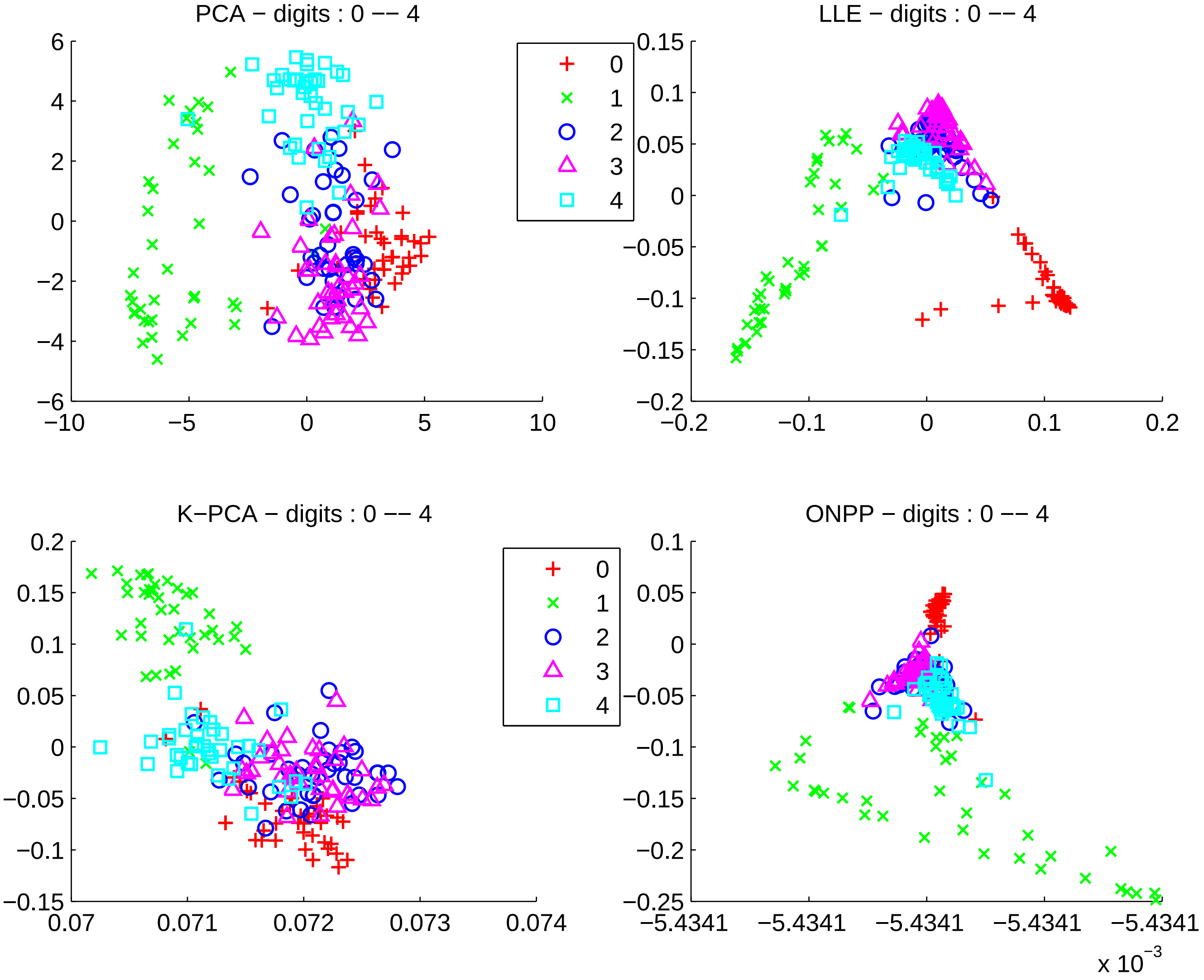




Example: Digit images (a random sample of 30)



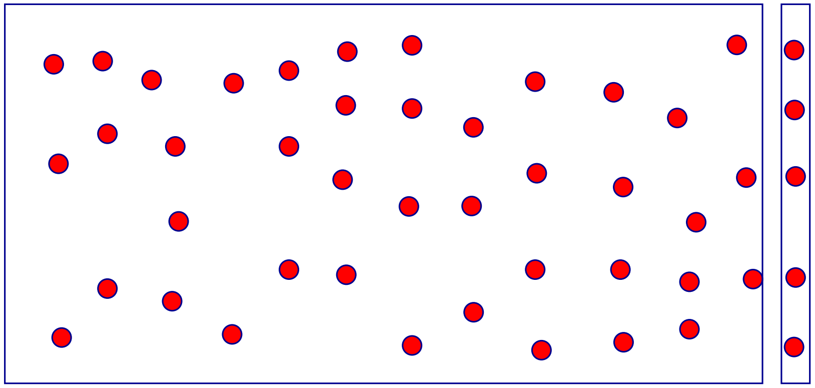
2-D 'reductions':



DIMENSION REDUCTION EXAMPLE: INFORMATION RETRIEVAL

Application: Information Retrieval

- Given: collection of documents (columns of a matrix $\boldsymbol{A}$) and a query vector $\boldsymbol{q}$.
- Representation: $\boldsymbol{m} \times \boldsymbol{n}$ term by document matrix



- A query $\boldsymbol{q}$ is a (sparse) vector in $\mathbb{R}^m$ ('pseudo-document')

Problem: find a column of $\boldsymbol{A}$ that best matches $\boldsymbol{q}$

- *Vector space model:* use $\cos\langle(\boldsymbol{A}(:,j),\boldsymbol{q}),\boldsymbol{j}=\boldsymbol{1}:\boldsymbol{n}$
- Requires the computation of $\boldsymbol{A}^T\boldsymbol{q}$
- Literal Matching → ineffective

Common approach: Dimension reduction (SVD)

- LSI: replace A by a low rank approximation [from SVD]

$$A = U\Sigma V^T \quad \rightarrow \quad A_k = U_k \Sigma_k V_k^T$$

- Replace similarity vector: $s = A^T q$ by $s_k = A_k^T q$
- Main issues: 1) computational cost 2) Updates

Idea: Replace A_k by $A\phi(A^T A)$, where ϕ == a filter function

Consider the step-function (Heaviside):

$$\phi(x) = \begin{cases} 0, & 0 \le x \le \sigma_k^2 \\ 1, & \sigma_k^2 \le x \le \sigma_1^2 \end{cases}$$

- Would yield the same result as TSVD but not practical

Use of polynomial filters

- Solution : use a polynomial approximation to ϕ
- Note: $\boxed{s^T = q^T A\phi(A^T A)}$, requires only Mat-Vec's
- Ideal for situations where data must be explored once or a small number of times only –
- Details skipped – see:

E. Kokiopoulou and YS, Polynomial Filtering in Latent Semantic Indexing for Information Retrieval, ACM-SIGIR, 2004.

IR: Use of the Lanczos algorithm (J. Chen, YS '09)

- ➤ Lanczos algorithm = Projection method on Krylov subspace Span$\{v, Av, \cdots, A^{m-1}v\}$

- ➤ Can get singular vectors with Lanczos, & use them in LSI

- ➤ Better: Use the Lanczos vectors directly for the projection

- ➤ K. Blom and A. Ruhe [SIMAX, vol. 26, 2005] perform a Lanczos run for each query [expensive].

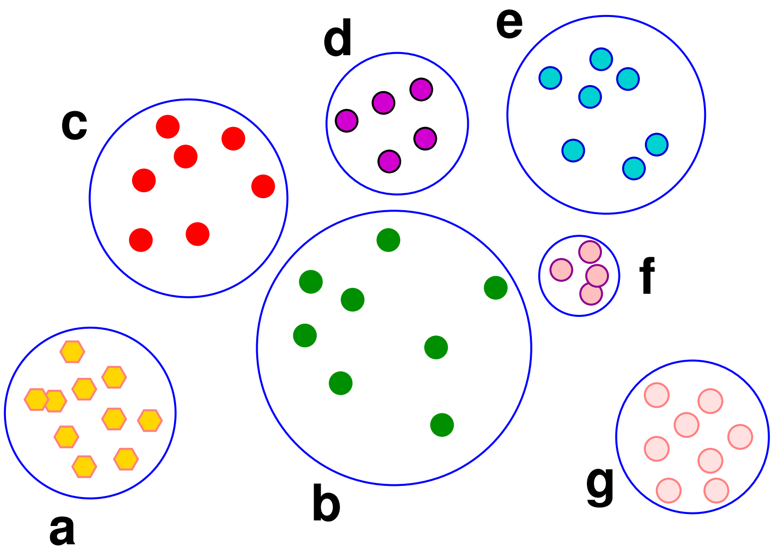
➤ Proposed: One Lanczos run- random initial vector. Then use Lanczos vectors in place of singular vectors.

In summary: Results comparable to those of SVD at a much lower cost.

Supervised learning

➤ We now have data that is ‘labeled’

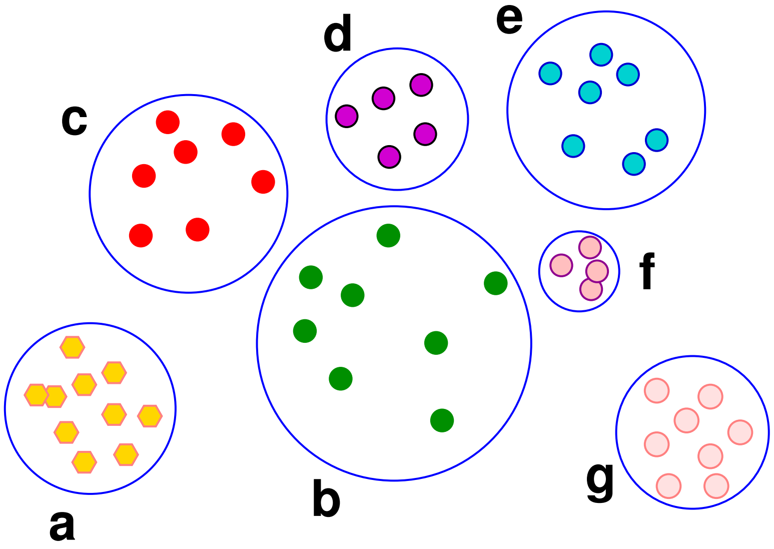
- Example: (health sciences) ‘malignant’- ’non malignant’
- Example: (materials) ’photovoltaic’, ’hard’, ’conductor’, ...
- Example: (Digit recognition) Digits ’0’, ’1’, ..., ’9’

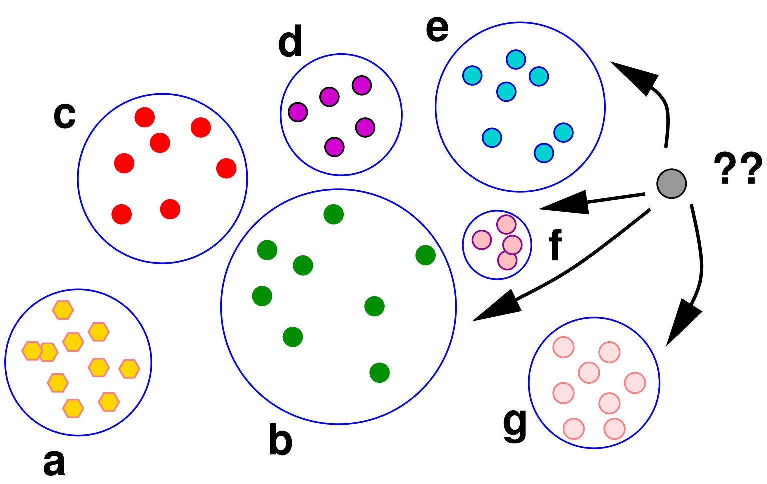


Supervised learning

We now have data that is ‘labeled’

- Example: (health sciences) ‘malignant’- ’non malignant’
- Example: (materials) ’photovoltaic’, ’hard’, ’conductor’, ...
- Example: (Digit recognition) Digits ’0’, ’1’, ..., ’9’

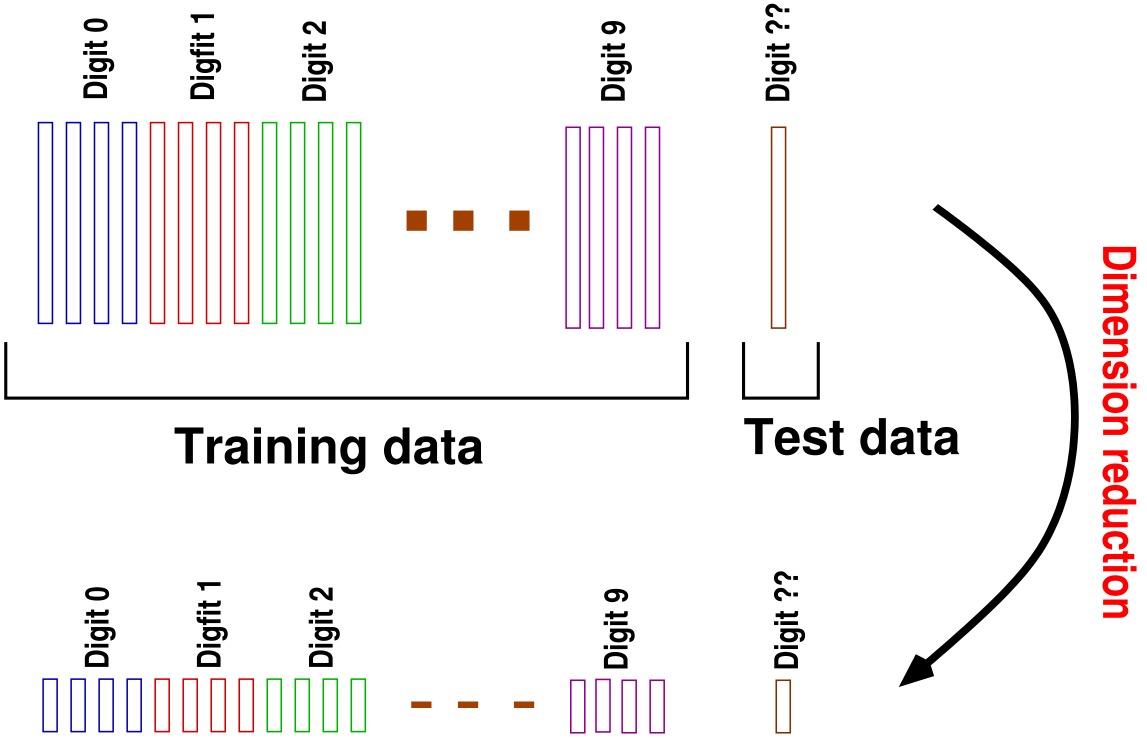




Supervised learning: classification

- Best illustration: written digits recognition example

Given: a set of labeled samples (training set), and an (unlabeled) test image.
Problem: find label of test image



Digit 0
Digfit 1
Digit 2
Digit 9
Digit ??
Training data
Test data
Dimension reduction
Digit 0
Digfit 1
Digit 2
Digit 9
Digit ??

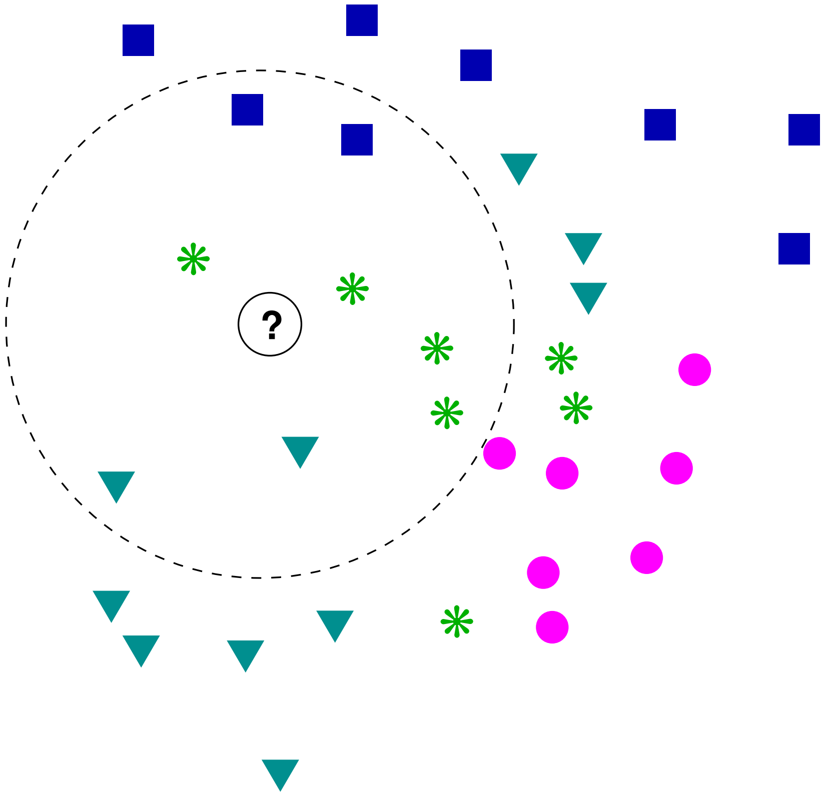
- Roughly speaking: we seek dimension reduction so that recognition is ‘more effective’ in low-dim. space

Basic method: K-nearest neighbors (KNN) classification

- ➤ Idea of a voting system: get distances between test sample and training samples

- ➤ Get the k nearest neighbors (here $k = 8$)

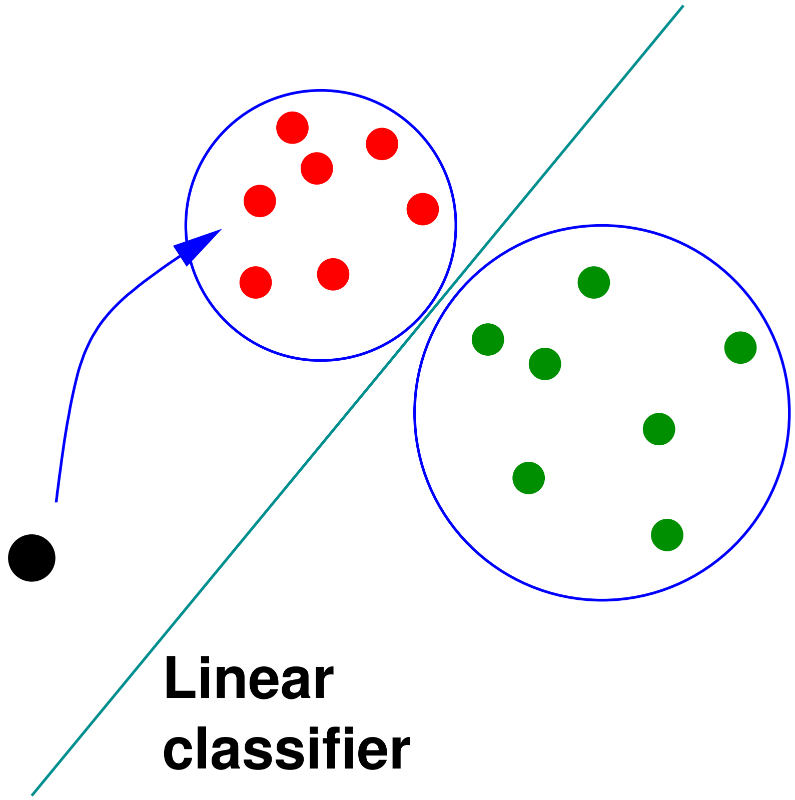
- ➤ Predominant class among these k items is assigned to the test sample ("$*$" here)



Supervised learning: Linear classification

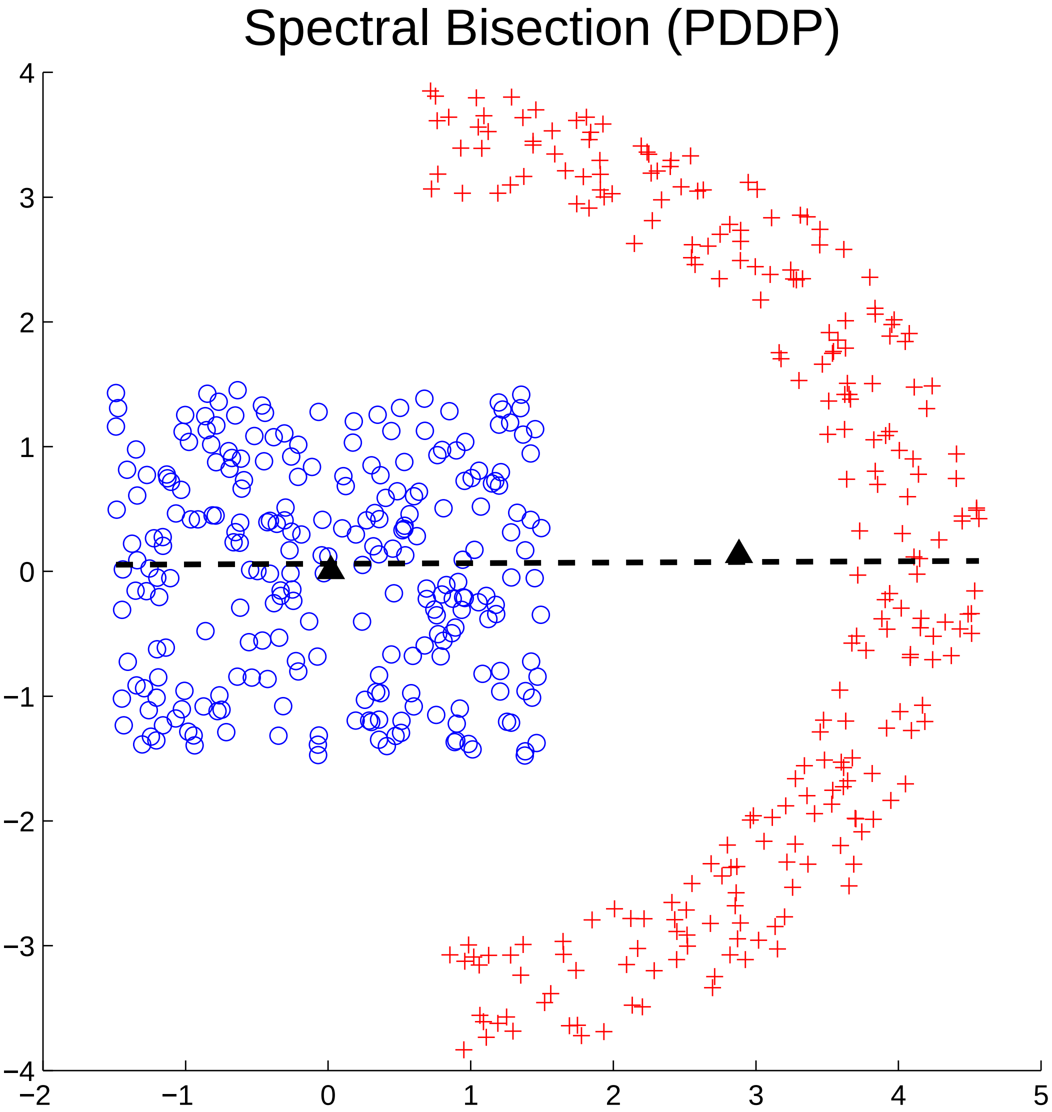
Linear classifiers: Find a hyperplane which best separates the data in classes A and B.

➤ Example of application: Distinguish between SPAM and non-SPAM e-mails

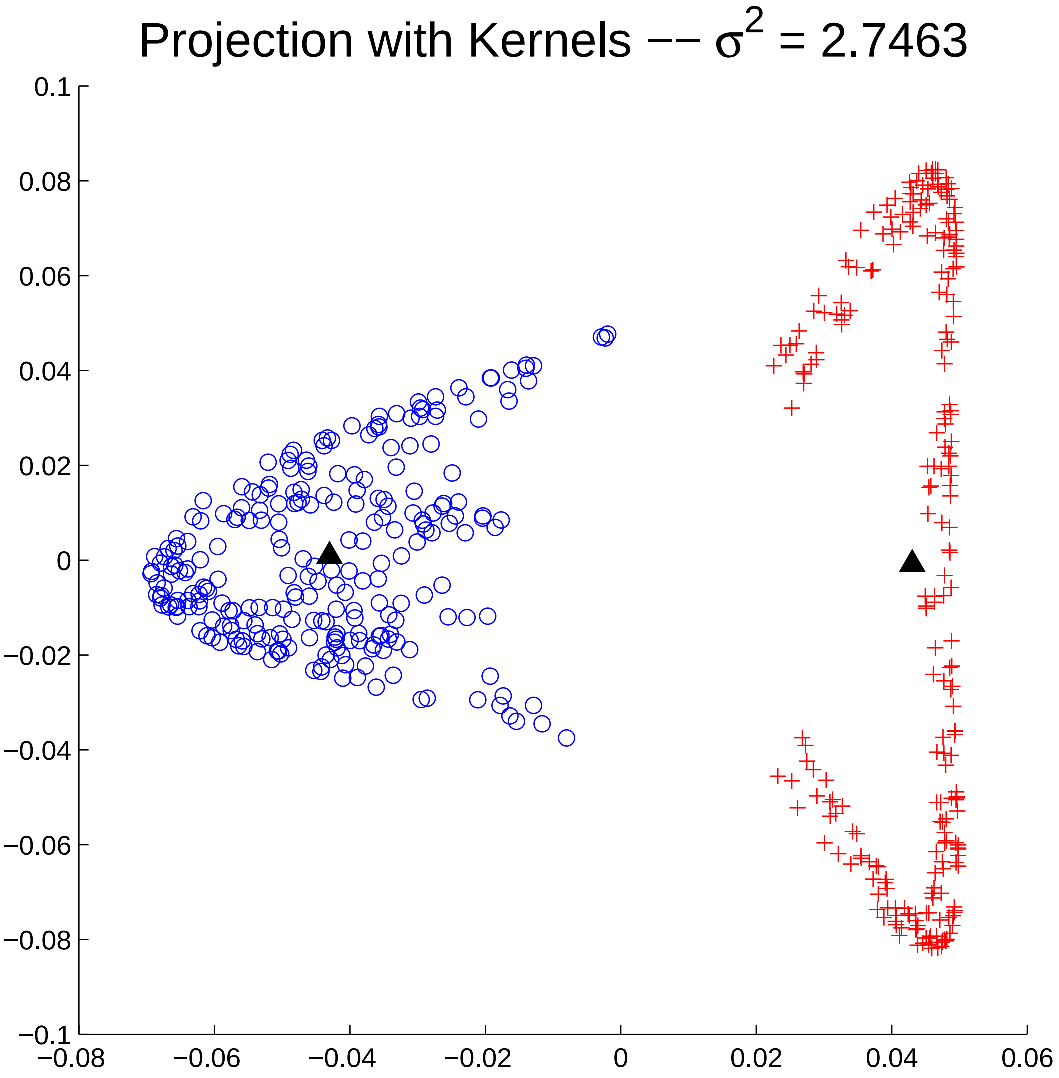


➤ Note: The world in non-linear. Often this is combined with Kernels – amounts to changing the inner product

A harder case:



- Use kernels to transform



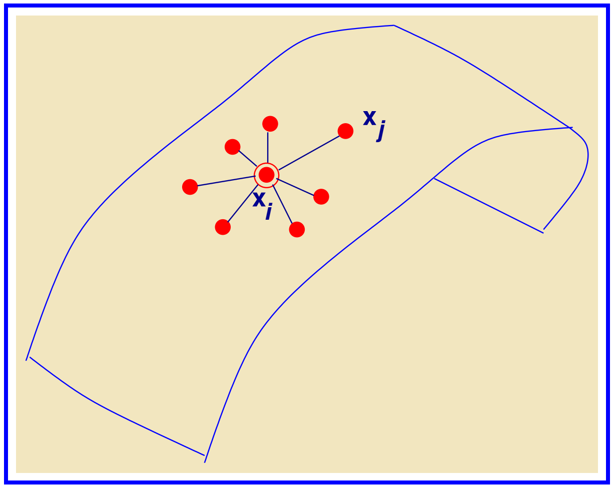
Transformed data with a Gaussian Kernel

GRAPH-BASED TECHNIQUES

Graph-based methods

- Start with a graph of data. e.g.: graph of k nearest neighbors (k-NN graph)

Want: Perform a projection which preserves the graph in some sense



- Define a *graph Laplacean:*

$$L = D - W$$

e.g.,: $w_{ij} = \begin{cases} 1 \text{ if } j \in Adj(i) \\ 0 \quad \text{else} \end{cases} \qquad D = \text{diag}\left[d_{ii} = \sum_{j \neq i} w_{ij} \right]$

with $Adj(i)$ = neighborhood of i (excluding i)

A side note: Graph partitioning

If x is a vector of signs (± 1) then

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

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..]

A few properties of graph Laplacean matrices

- Let L = any matrix s.t. $L = D - W$, with:

$$D = diag(d_i), \quad w_{ij} \geq 0, \quad d_i \ = \ \sum_{j \neq i} w_{ij}$$

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

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

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

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

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

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

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

→ PCA equivalent to maximizing $\sum_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$

Property 4: (Graph partitioning) If $\boldsymbol{x}$ is a vector of signs (± 1) and an edge-cut = pair $(\boldsymbol{i}, \boldsymbol{j})$ with $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ then

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

➤ Can be used for partitioning graphs, or for 'clustering' [take $\boldsymbol{p} = \boldsymbol{sign}(\boldsymbol{u}_2)$, where $\boldsymbol{u}_2$ = 2nd smallest eigenvector..]

Example: The Laplacean eigenmaps approach

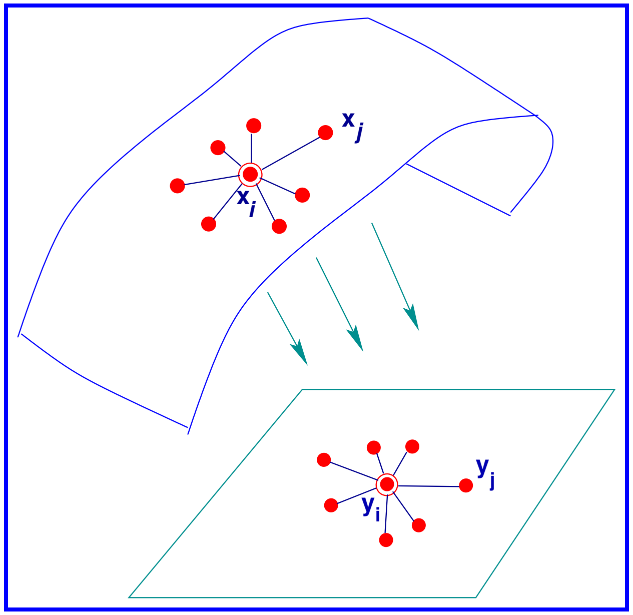
Laplacean Eigenmaps [Belkin-Niyogi '01] *minimizes*

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

Motivation: if $\|x_i - x_j\|$ is small (orig. data), we want $\|y_i - y_j\|$ to be also small (low-Dim. data)

➤ Original data used indirectly through its graph

➤ Leads to $n \times n$ sparse eigenvalue problem [In 'sample' space]



- Problem translates to:

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

- Solution (sort eigenvalues increasingly):

$$(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{u}_i = \boldsymbol{\lambda}_i \boldsymbol{D}\boldsymbol{u}_i \; ; \quad \boldsymbol{y}_i = \boldsymbol{u}_i^\top ; \quad \boldsymbol{i} = 1, \cdots, \boldsymbol{d}$$

- Note: can assume $\boldsymbol{D} = \boldsymbol{I}$. Amounts to rescaling data. Problem becomes

$$\boxed{(\boldsymbol{I}-\boldsymbol{W})\boldsymbol{u}_i = \boldsymbol{\lambda}_i \boldsymbol{u}_i \; ; \quad \boldsymbol{y}_i = \boldsymbol{u}_i^\top ; \quad \boldsymbol{i} = 1, \cdots, \boldsymbol{d}}$$

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

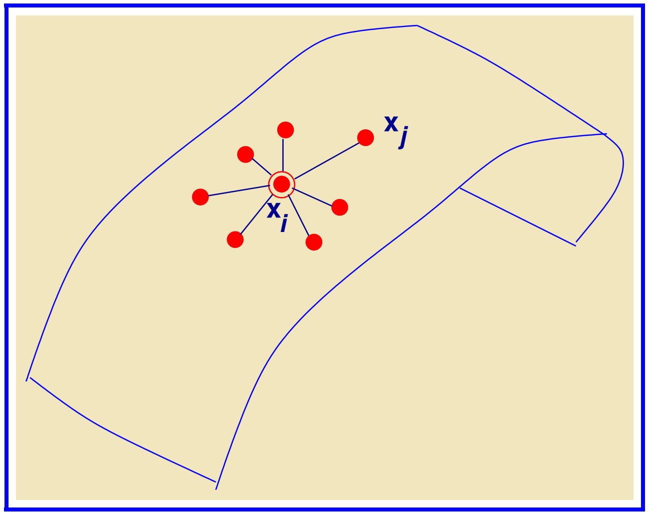
2) Objective function is changed.

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

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



2. Mapping:

The $\boldsymbol{y_i}$'s should obey the same 'affinity' as $\boldsymbol{x_i}$'s $\leadsto$

Minimize:

$$\sum_i \left\| \boldsymbol{y_i} - \sum_j \boldsymbol{w_{ij} y_j} \right\|^2 \quad \text{subject to:} \quad \boldsymbol{Y1 = 0}, \quad \boldsymbol{YY^\top = I}$$

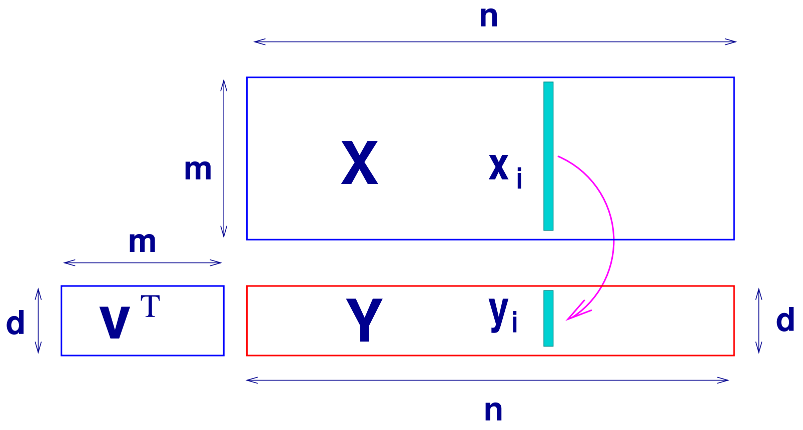
Solution:

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

➤ $\boldsymbol{(I - W^\top)(I - W)}$ 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 $\boldsymbol{D} \equiv \boldsymbol{diag(\{\Sigma_i w_{ij}\})}$.

- Optimization problem is to solve

$$\min_{\boldsymbol{Y} \in \mathbb{R}^{d\times n},\ \boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^\top = \boldsymbol{I}} \quad \Sigma_{i,j} w_{ij} \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|^2, \quad \boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}.$$

- Difference with eigenmaps: $\boldsymbol{Y}$ is a projection of $\boldsymbol{X}$ data

- Solution (sort eigenvalues increasingly)

$$\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top \boldsymbol{v}_i \quad \boldsymbol{y}_{i,:} = \boldsymbol{v}_i^\top \boldsymbol{X}$$

- 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

- A linear (orthogonoal) version of LLE obtained by writing $\boldsymbol{Y}$ in the form $\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}$

- Same graph as LLE. Objective: preserve the affinity graph (as in LEE) *but* with the constraint $\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}$

- Problem solved to obtain mapping:

$$\min_{\boldsymbol{V}} \text{Tr} \left[\boldsymbol{V}^\top \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{W}^\top)(\boldsymbol{I} - \boldsymbol{W})\boldsymbol{X}^\top \boldsymbol{V} \right]$$

s.t. $\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{I}$

- In LLE replace $\boldsymbol{V}^\top \boldsymbol{X}$ by $\boldsymbol{Y}$

Implicit vs explicit mappings

- In PCA the mapping Φ from high-dimensional space ($\mathbb{R}^m$) to low-dimensional space ($\mathbb{R}^d$) is explicitly known:

$$\boldsymbol{y} = \Phi(\boldsymbol{x}) \equiv \boldsymbol{V}^T \boldsymbol{x}$$

- In Eigenmaps and LLE we only know

$$\boldsymbol{y_i} = \phi(\boldsymbol{x_i}), \boldsymbol{i} = 1, \cdots, \boldsymbol{n}$$

- Mapping ϕ is complex, i.e.,
- Difficult to get $\phi(x)$ for an arbitrary x not in the sample.
- Inconvenient for classification
- “The out-of-sample extension” problem

K-nearest neighbor graphs

➤ Nearest Neighbor graphs needed in: data mining, manifold learning, robot motion planning, computer graphics, ...

➤ Given: a set of n data points $X = \{x_1, \dots, x_n\} \rightarrow$ vertices

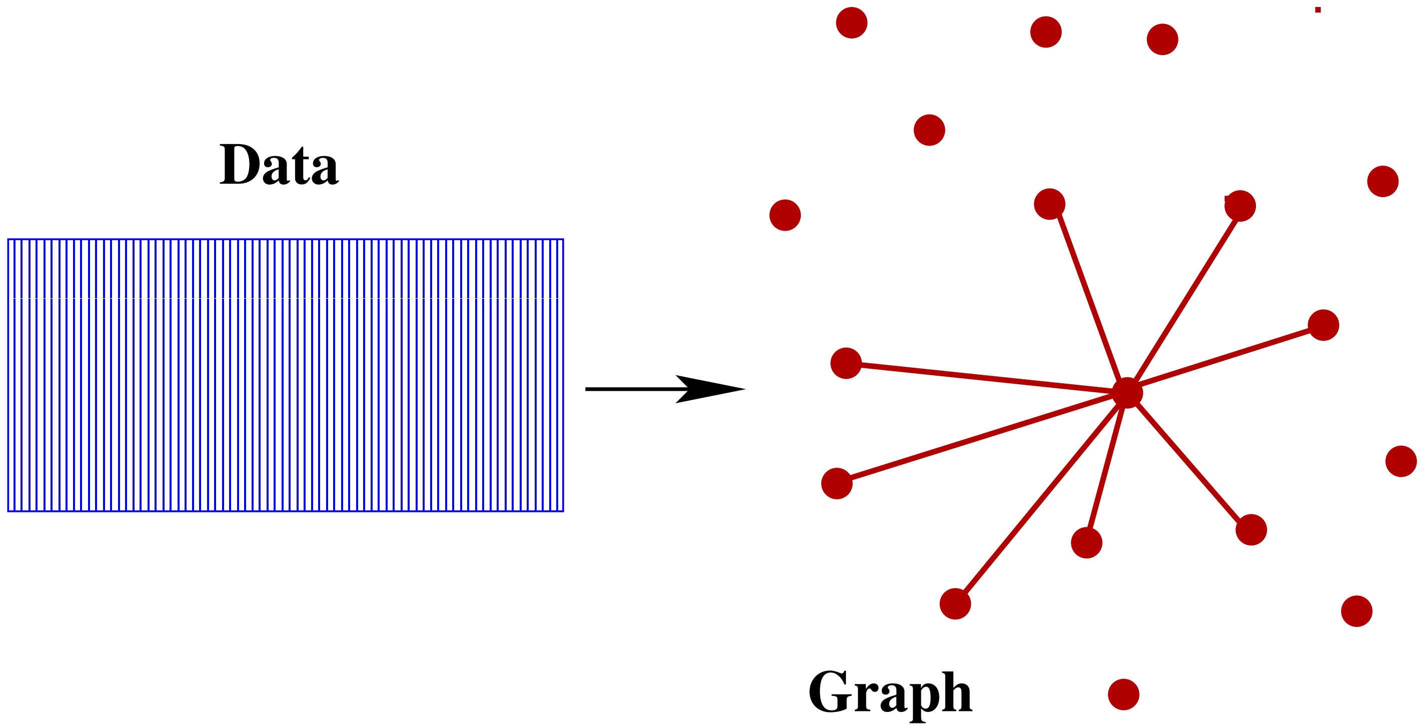
➤ Given: a proximity measure between two data points x_i and x_j – as measured by a quantity $\rho(x_i, x_j)$

➤ Want: For each point x_i a list of the 'nearest neighbors' of x_i (edges between x_i and these nodes).

(wll make the definition less vague shortly)

Building a nearest neighbor graph

- Problem: Build a nearest-neighbor graph from given data



- Will demonstrate the power of the Lanczos algorithm combined with a divide a conquer approach.

Two types of nearest neighbor graph often used:

ϵ-graph: Edges consist of pairs (x_i, x_j) such that $\rho(x_i, x_j) \leq \epsilon$

kNN graph: Nodes adjacent to x_i are those nodes x_ℓ with the k with smallest distances $\rho(x_i, x_\ell)$.

- ➤ ϵ-graph is undirected and is geometrically motivated. Issues: 1) may result in disconnected components 2) what ϵ?
- ➤ kNN graphs are directed in general (can be trivially fixed).
- ➤ kNN graphs especially useful in practice.

Divide and conquer KNN: key ingredient

- Key ingredient is *Spectral bisection*
- Let the data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$
- Each column == a data point.
- Center the data: $\hat{X} = [\hat{x}_1, \ldots, \hat{x}_n] = X - ce^T$ where c == centroid; $e = ones(d, 1)$ (matlab)

Goal: Split $\hat{X}$ into halves using a hyperplane.

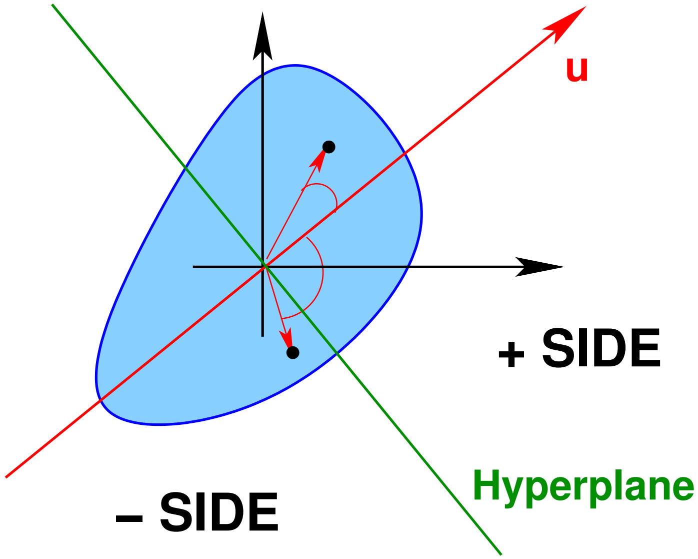
Method: Principal Direction Divisive Partitioning D. Boley, ’98.

Idea: Use the (σ, u, v) = largest singular triplet of $\hat{X}$ with:

$$u^T\hat{X} = \sigma v^T.$$

- Hyperplane is defined as $\langle u, x \rangle = 0$, i.e., it splits the set of data points into two subsets:

$$X_+ = \{x_i \mid u^T \hat{x}_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid u^T \hat{x}_i < 0\}.$$



- Note that $u^T \hat{x}_i = u^T \hat{X} e_i = \sigma v^T e_i \rightarrow$

$$X_+ = \{x_i \mid v_i \geq 0\} \quad \text{and} \quad X_- = \{x_i \mid v_i < 0\},$$

where v_i is the i-th entry of v.

- In practice: replace above criterion by

$$\boxed{X_+ = \{x_i \mid v_i \geq \text{med}(v)\} \ \& \ X_- = \{x_i \mid v_i < \text{med}(v)\}}$$

where *med*(v) *== median of the entries of* v.

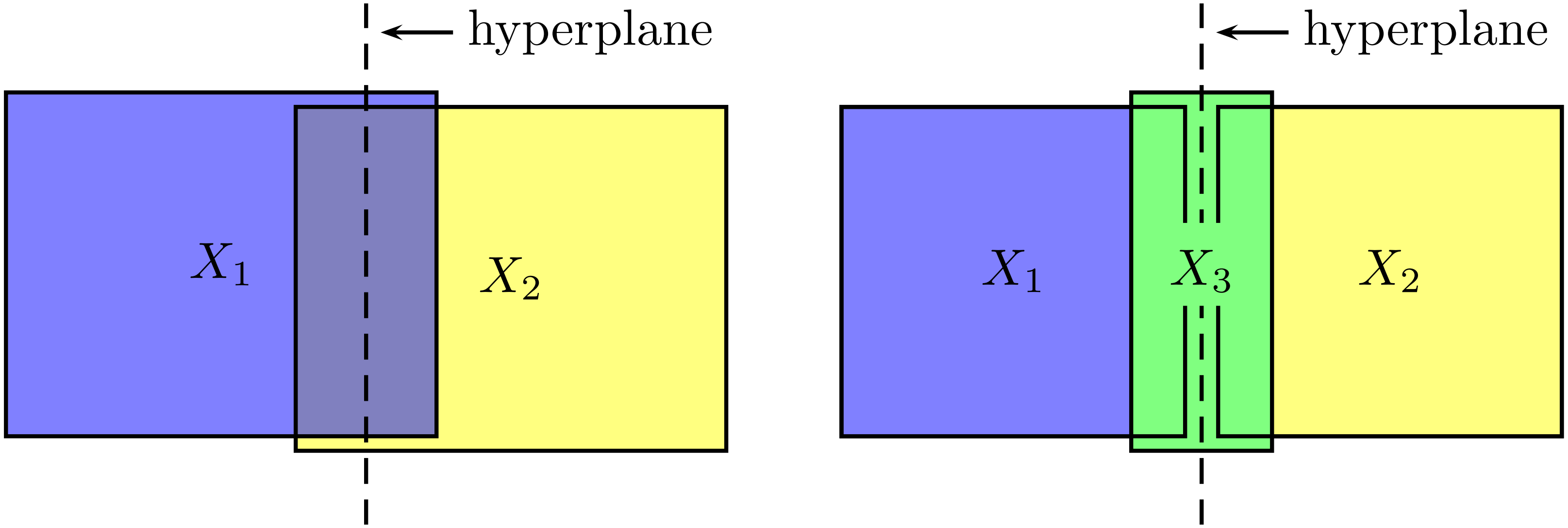
- For largest singular triplet (σ, u, v) of $\hat{X}$: use Golub-Kahan-Lanczos algorithm or Lanczos applied to $\hat{X}\hat{X}^T$ or $\hat{X}^T\hat{X}$

- Cost (assuming s Lanczos steps) : $O(n \times d \times s)$; Usually: d very small

Two divide and conquer algorithms

Overlap method: divide current set into two overlapping subsets $\boldsymbol{X_1, X_2}$

Glue method: divide current set into two disjoint subsets $\boldsymbol{X_1, X_2}$ plus a third set $\boldsymbol{X_3}$ called gluing set.



The Overlap Method

➤ Divide current set $\boldsymbol{X}$ into two overlapping subsets:

$\boldsymbol{X_1 = \{x_i \mid v_i \geq -h_\alpha(S_v)\}}$ and $\boldsymbol{X_2 = \{x_i \mid v_i < h_\alpha(S_v)\}}$,

- where $\boldsymbol{S_v = \{|v_i| \mid i = 1, 2, \ldots, n\}}$.
- and $\boldsymbol{h_\alpha(\cdot)}$ is a function that returns an element larger than $\boldsymbol{(100\alpha)\%}$ of those in $\boldsymbol{S_v}$.

➤ Rationale: to ensure that the two subsets overlap $\boldsymbol{(100\alpha)\%}$ of the data, i.e.,

$$|\boldsymbol{X_1 \cap X_2}| = \lceil \boldsymbol{\alpha} |\boldsymbol{X}| \rceil .$$

The Glue Method

Divide the set X into two disjoint subsets X_1 and X_2 with a gluing subset X_3:

$$X_1 \cup X_2 = X, \quad X_1 \cap X_2 = \emptyset, \quad X_1 \cap X_3 \neq \emptyset, \quad X_2 \cap X_3 \neq \emptyset.$$

Criterion used for splitting:

$$X_1 = \{x_i \mid v_i \geq 0\}, \quad X_2 = \{x_i \mid v_i < 0\},$$
$$X_3 = \{x_i \mid -h_\alpha(S_v) \leq v_i < h_\alpha(S_v)\}.$$

Note: gluing subset X_3 here is just the intersection of the sets X_1, X_2 of the overlap method.

Approximate kNN Graph Construction: The Overlap Method

```
function $G = k$NN-Overlap($X$, $k$, $\alpha$)
    if $|X| < n_k$ then
        $G \leftarrow k$NN-BruteForce($X$, $k$)
    else
        $(X_1, X_2) \leftarrow$ Divide-Overlap($X$, $\alpha$)
        $G_1 \leftarrow k$NN-Overlap($X_1$, $k$, $\alpha$)
        $G_2 \leftarrow k$NN-Overlap($X_2$, $k$, $\alpha$)
        $G \leftarrow$ Conquer($G_1$, $G_2$)
        Refine($G$)
    end if
end function
```

Approximate kNN Graph Construction: The Glue Method

```
function $G = k$NN-Glue$(X, k, \alpha)$
    if $|X| < n_k$ then
        $G \leftarrow k$NN-BruteForce$(X, k)$
    else
        $(X_1, X_2, X_3) \leftarrow$ Divide-Glue$(X, \alpha)$
        $G_1 \leftarrow k$NN-Glue$(X_1, k, \alpha)$
        $G_2 \leftarrow k$NN-Glue$(X_2, k, \alpha)$
        $G_3 \leftarrow k$NN-Glue$(X_3, k, \alpha)$
        $G \leftarrow$ Conquer$(G_1, G_2, G_3)$
        Refine$(G)$
    end if
end function
```

Theorem The time complexity for the overlap method is

$$T_{\mathrm{o}}(n) = \Theta(dn^{t_{\mathrm{o}}}), \tag{1}$$

where

$$t_{\mathrm{o}} = \log_{2/(1+\alpha)} 2 = \frac{1}{1 - \log_2(1+\alpha)}. \tag{2}$$

Theorem The time complexity for the glue method is

$$T_{\mathrm{g}}(n) = \Theta(dn^{t_{\mathrm{g}}}/\alpha), \tag{3}$$

where t_{g} is the solution to the equation: $\frac{2}{2^t} + \alpha^t = 1.$

Example: When $\alpha = 0.1$, then $t_{\mathrm{o}} = 1.16$ while $t_{\mathrm{g}} = 1.12$.

Multilevel techniques in brief

- Divide and conquer paradigms as well as multilevel methods in the sense of ‘domain decomposition’

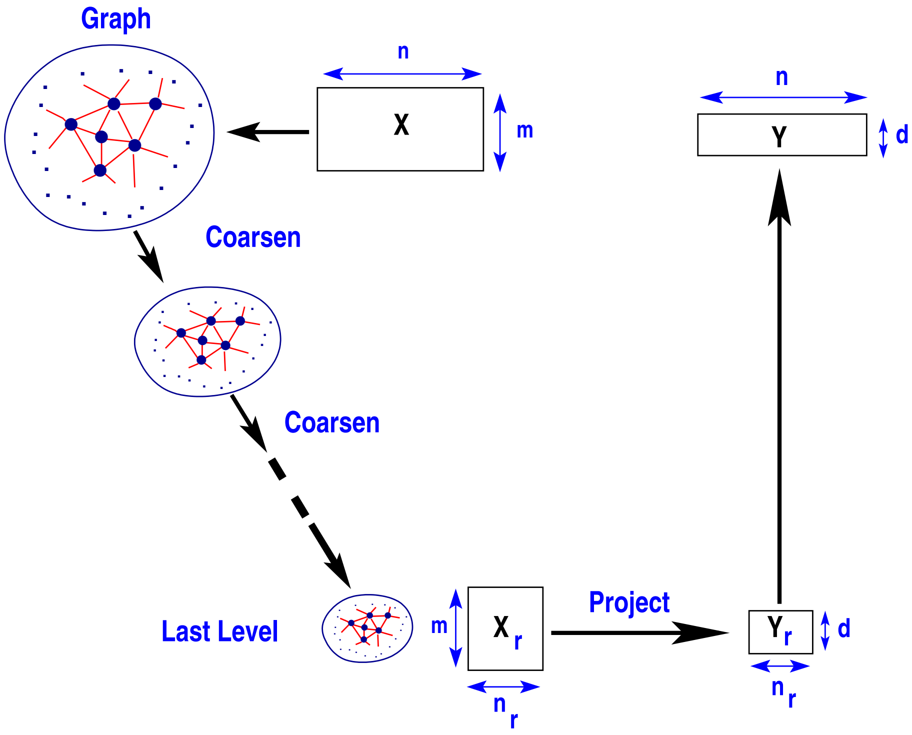
- Main principle: very costly to do an SVD [or Lanczos] on the whole set. Why not find a smaller set on which to do the analysis – without too much loss?

- Tools used: graph coarsening, divide and conquer –

- For text data we use hypergraphs

Multilevel Dimension Reduction

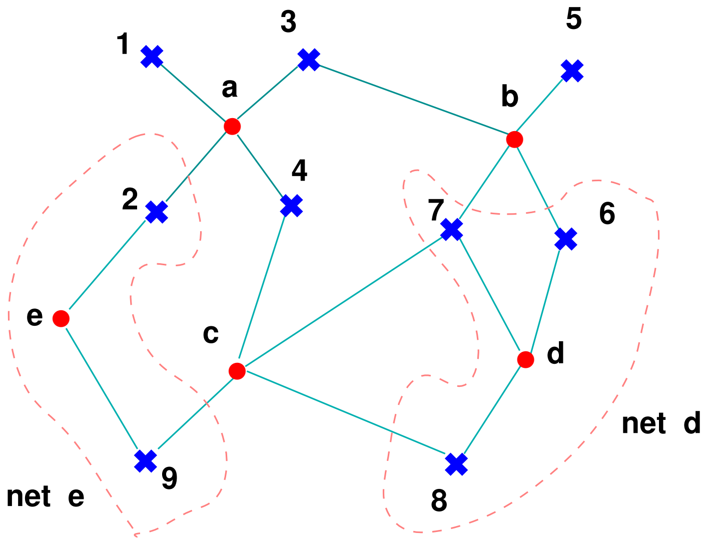
Main Idea: coarsen for a few levels. Use the resulting data set $\hat{X}$ to find a projector P from $\mathbb{R}^m$ to $\mathbb{R}^d$. P can be used to project original data or new data



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

Making it work: Use of Hypergraphs for sparse data

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



Left: a (sparse) data set of n entries in $\mathbb{R}^m$ represented by a matrix $A \in \mathbb{R}^{m \times n}$

Right: corresponding hypergraph $H = (V, E)$ with vertex set V representing to the columns of A.

- Hypergraph Coarsening uses *column matching* – similar to a common one used in graph partitioning

- Compute the non-zero inner product $\langle a^{(i)}, a^{(j)} \rangle$ between two vertices i and j, i.e., the ith and jth columns of A.

- Note: $\langle a^{(i)}, a^{(j)} \rangle = \|a^{(i)}\| \|a^{(j)}\| \cos \theta_{ij}$.

Modif. 1: Parameter: $0 < \epsilon < 1$. Match two vertices, i.e., columns, only if angle between the vertices satisfies:

$$\boxed{\tan \theta_{ij} \le \epsilon}$$

Modif. 2: Scale coarsened columns. If i and j matched and if $\|a^{(i)}\|_0 \ge \|a^{(j)}\|_0$ replace $a^{(i)}$ and $a^{(j)}$ by

$$c^{(\ell)} = \left(\sqrt{1 + \cos^2 \theta_{ij}}\right) a^{(i)}$$

➤ Call C the coarsened matrix obtained from A using the approach just described

Lemma: Let $C \in \mathbb{R}^{m\times c}$ be the coarsened matrix of A obtained by one level of coarsening of $A \in \mathbb{R}^{m\times n}$, with columns $a^{(i)}$ and $a^{(j)}$ matched if $\tan \theta_i \le \epsilon$. Then

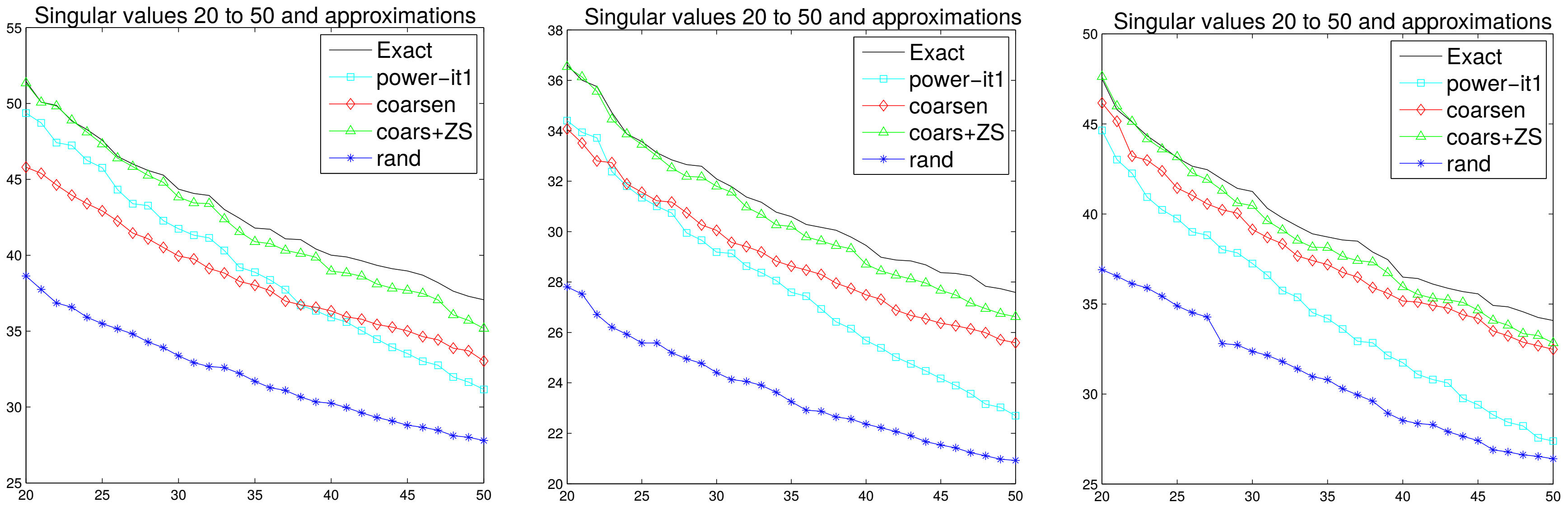
$$|x^T A A^T x - x^T C C^T x| \le 3\epsilon \|A\|_F^2,$$

for any $x \in \mathbb{R}^m$ with $\|x\|_2 = 1$.

➤ Very simple bound for Rayleigh quotients for any x.

➤ Implies some bounds on singular values and norms - skipped.

Tests: Comparing singular values



Results for the datasets CRANFIELD (left), MEDLINE (middle), and TIME (right).

Low rank approximation: Coarsening, random sampling, and rand+coarsening. Err1 $= \|A - H_k H_k^T A\|_F$; *Err2*$= \frac{1}{k}\sum_k \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$

Dataset	n	k	c	Coarsen		Rand Sampl	
				Err1	Err2	Err1	Err2
Kohonen	4470	50	1256	86.26	0.366	93.07	0.434
aft01	8205	50	1040	913.3	0.299	1006.2	0.614
FA	10617	30	1504	27.79	0.131	28.63	0.410
chipcool0	20082	30	2533	6.091	0.313	6.199	0.360
brainpc2	27607	30	865	2357.5	0.579	2825.0	0.603
scfxm1-2b	33047	25	2567	2326.1	–	2328.8	–
thermomechTC	102158	30	6286	2063.2	–	2079.7	–
Webbase-1M	1000005	25	15625	–	–	3564.5	–

Conclusion

- *Many* interesting new matrix problems in areas that involve the effective mining of data
- Among the most pressing issues is that of reducing computational cost - [SVD, SDP, ..., too costly]
- Many online resources available
- Huge potential in areas like materials science though inertia has to be overcome
- To a researcher in computational linear algebra : Tsunami of change on types or problems, algorithms, frameworks, culture,..
- But change should be welcome

When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one which has opened for us.

Alexander Graham Bell (1847-1922)

- In the words of Lao Tzu:

If you do not change directions, you may end-up where you are heading

Thank you !

- Visit my web-site at `www.cs.umn.edu/~saad`