UNIVERSITY OF MINNESOTA TWIN CITIES

Numerical Linear Algebra for data-related applications

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

Modelling 2019

Olomouc, September 20, 2019

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:
"Solving linear systems can be interesting".

➤ Survey of the state of the art linear algebra at that time: direct methods, iterative methods, conditioning, preconditioning, The Conjugate Gradient method, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solving linear systems

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:
"Solving linear systems can be interesting".

➤ Survey of the state of the art linear algebra at that time: direct methods, iterative methods, conditioning, preconditioning, The Conjugate Gradient method, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solving linear systems

➤ 66 years later – we can certainly state that:

"Linear Algebra problems in Machine Learning can be interesting"

Focus of numerical linear algebra changed many times over the years

- This is because linear algebra is a key tool when solving challenging new problems in various disciplines

1940s–1950s: Major issue: the flutter problem in aerospace engineering $\longrightarrow$ eigenvalue problem [cf. Olga Taussky Todd]

- Then came the discoveries of the LR and QR algorithms. The package Eispack followed a little later

1960s: Problems related to the power grid promoted what we would call today general sparse matrix techniques

Late 1980s: Thrust on parallel matrix computations.

Late 1990s: Spur of interest in "financial computing"

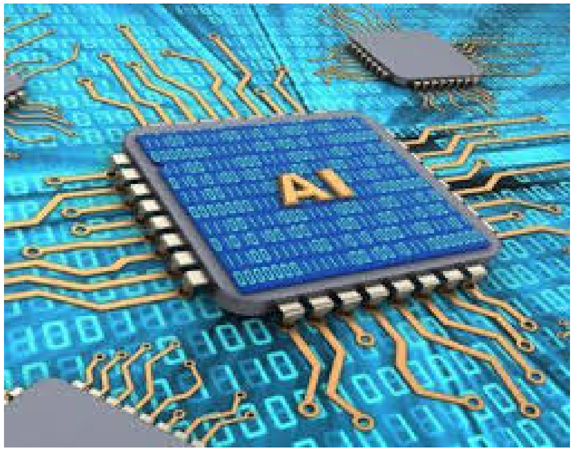
Solution of PDEs (e.g., Fluid Dynamics) and problems in mechanical eng. (e.g. structures) major force behind numerical linear algebra algorithms in the past few decades.

➤ Strong new forces are now reshaping the field today: Applications related to the use of "data"

➤ Machine learning is appearing in unexpected places:

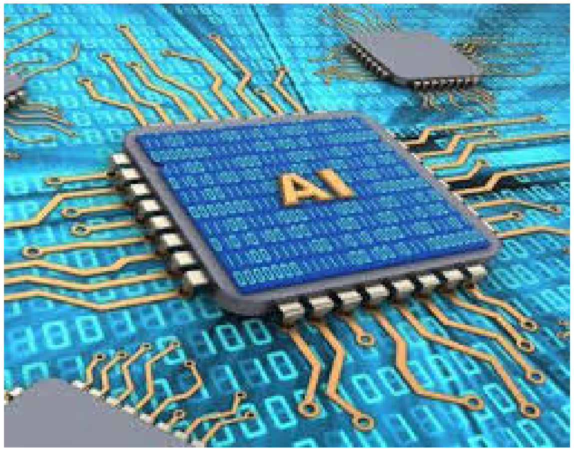
- design of materials
- machine learning in geophysics
- self-driving cars, ..
- ....

Big impact on the economy



- New economy driven by Google, Facebook, Netflix, Amazon, Twitter, Ali-Baba, Tencent, ..., and even the big department stores (Walmart, ...)
- Huge impact on **Jobs**

Big impact on the economy



➤ New economy driven by Google, Facebook, Netflix, Amazon, Twitter, Ali-Baba, Tencent, ..., and even the big department stores (Walmart, ...)

➤ Huge impact on **Jobs**

➤ In contrast: Old economy [driven by Boeing, GM, Ford, Mining industry, US Steel, Aerospatiale, ...] does not have as much to offer...

➤ Imperative to look at what you we do under new lenses: DATA

$Ax=b$

$-\Delta u = f$

Graph Partitioning

Preconditioning

Model reduction

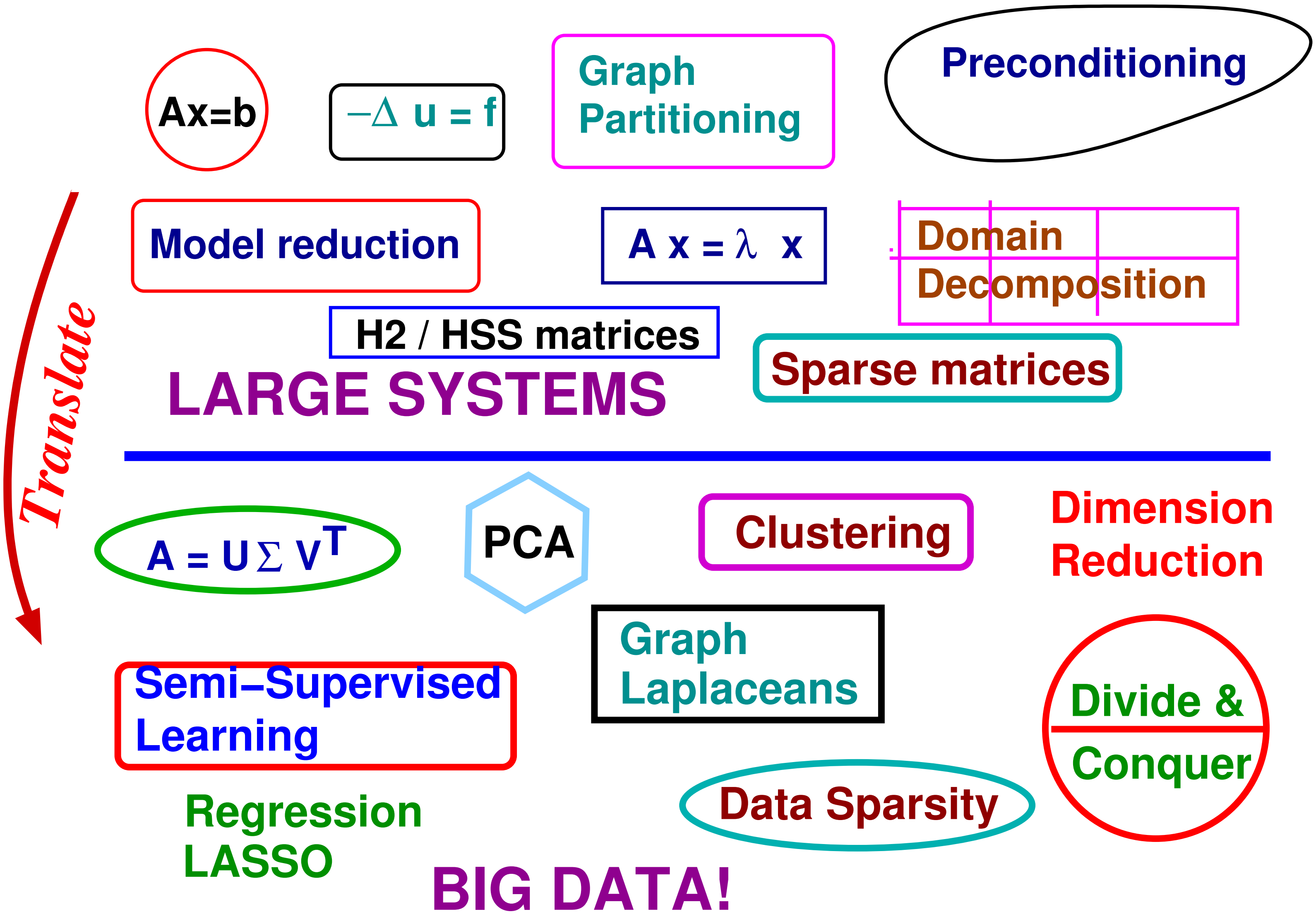
$A x = \lambda x$

Domain Decomposition

H2 / HSS matrices

Sparse matrices

LARGE SYSTEMS



Ax=b
$-\Delta u = f$
Graph Partitioning
Preconditioning
Model reduction
$A x = \lambda x$
Domain Decomposition
H2 / HSS matrices
Sparse matrices
LARGE SYSTEMS
Translate
$A = U \Sigma V^T$
PCA
Clustering
Dimension Reduction
Graph Laplaceans
Semi–Supervised Learning
Divide & Conquer
Regression
LASSO
Data Sparsity
BIG DATA!

Introduction, background, and motivation

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

- Main tools used: linear algebra; Statistics; Optimization; graph theory; approximation theory; ...

A few sample (classes of) problems & methods:

- Classification: 'Benign – Malignant', 'Dangerous-Safe', Face recognition, digit recognition, pattern recognition, ..
- Graphs/ networks analysis: Pagerank, communicability, node centrality, ...
- Matrix completion: Recommender systems
- Projection type methods: PCA, LSI, Clustering, Eigenmaps, LLE, Isomap, ...
- (Deep) Neural Networks: Convolutional Neural Networks; Image recognition; Speech recognition; ...
- Computational statistics: [Diag(inv(Cov)), Log det (A), ...]

- Two broad classes methods: *supervised* and *unsupervised* learning.

- General approach in both methods: Reduce dimension first then tackle task in lower dimension space.

Major tool of Data Mining: Dimension reduction

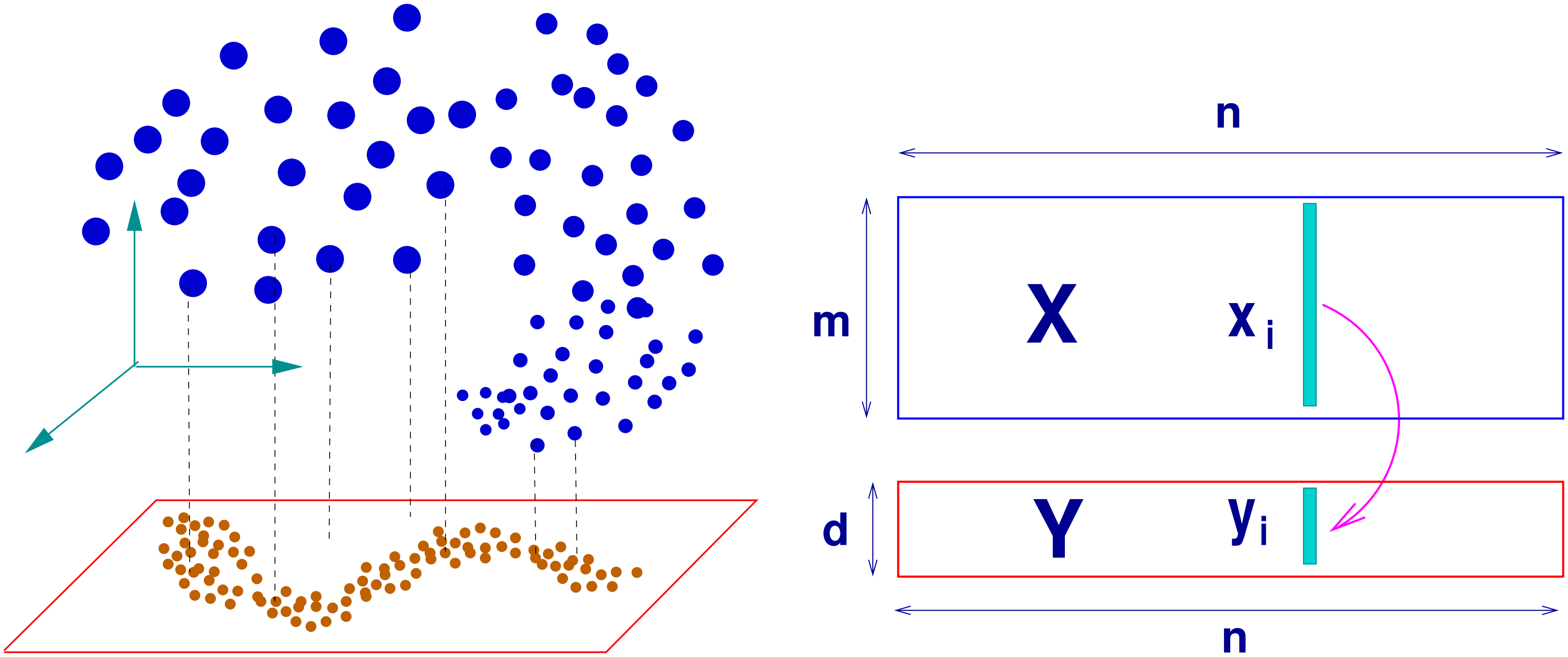
➤ Goal is not as much to reduce size (& cost) but to:

- 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: $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m\times n}$, find a low-dimens. representation $Y = [y_1, \cdots, y_n] \in \mathbb{R}^{d\times n}$ of X

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

$$\phi(x_i) = y_i, \quad i = 1, \cdots, n$$



➤ 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* $\boldsymbol{W}$ is computed to maximize variance of projected data:

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

- Leads to maximizing

$$\mathrm{Tr}\left[\boldsymbol{W}^\top(\boldsymbol{X}-\boldsymbol{\mu}\boldsymbol{e}^\top)(\boldsymbol{X}-\boldsymbol{\mu}\boldsymbol{e}^\top)^\top\boldsymbol{W}\right], \quad \boldsymbol{\mu}=\tfrac{1}{n}\Sigma_{i=1}^{n}\boldsymbol{x}_i$$

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

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' ..

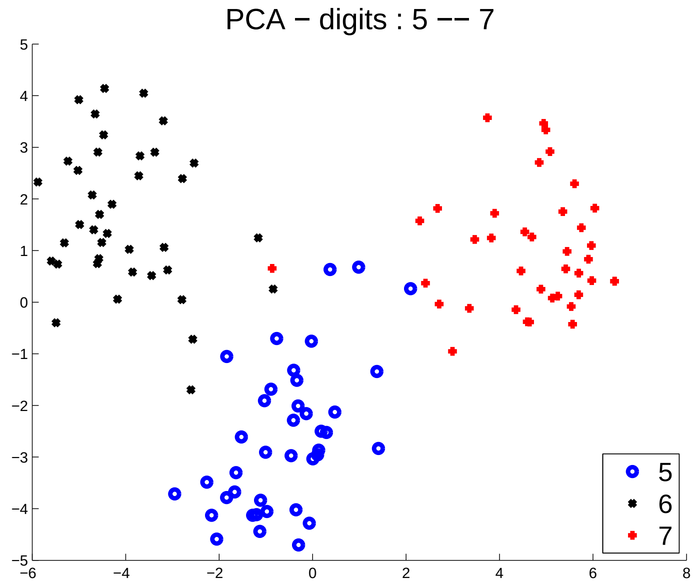
$$\sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i\|^2 = \sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{y}_i\|^2$$

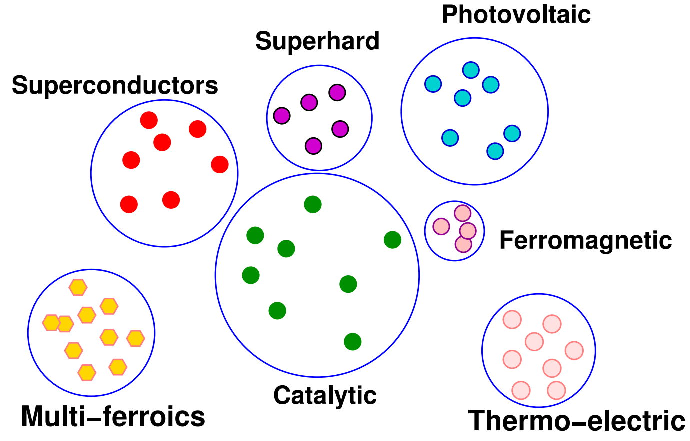
- In some methods recentering to zero is not done, i.e., $\bar{\boldsymbol{X}}$ replaced by $\boldsymbol{X}$.

Unsupervised learning

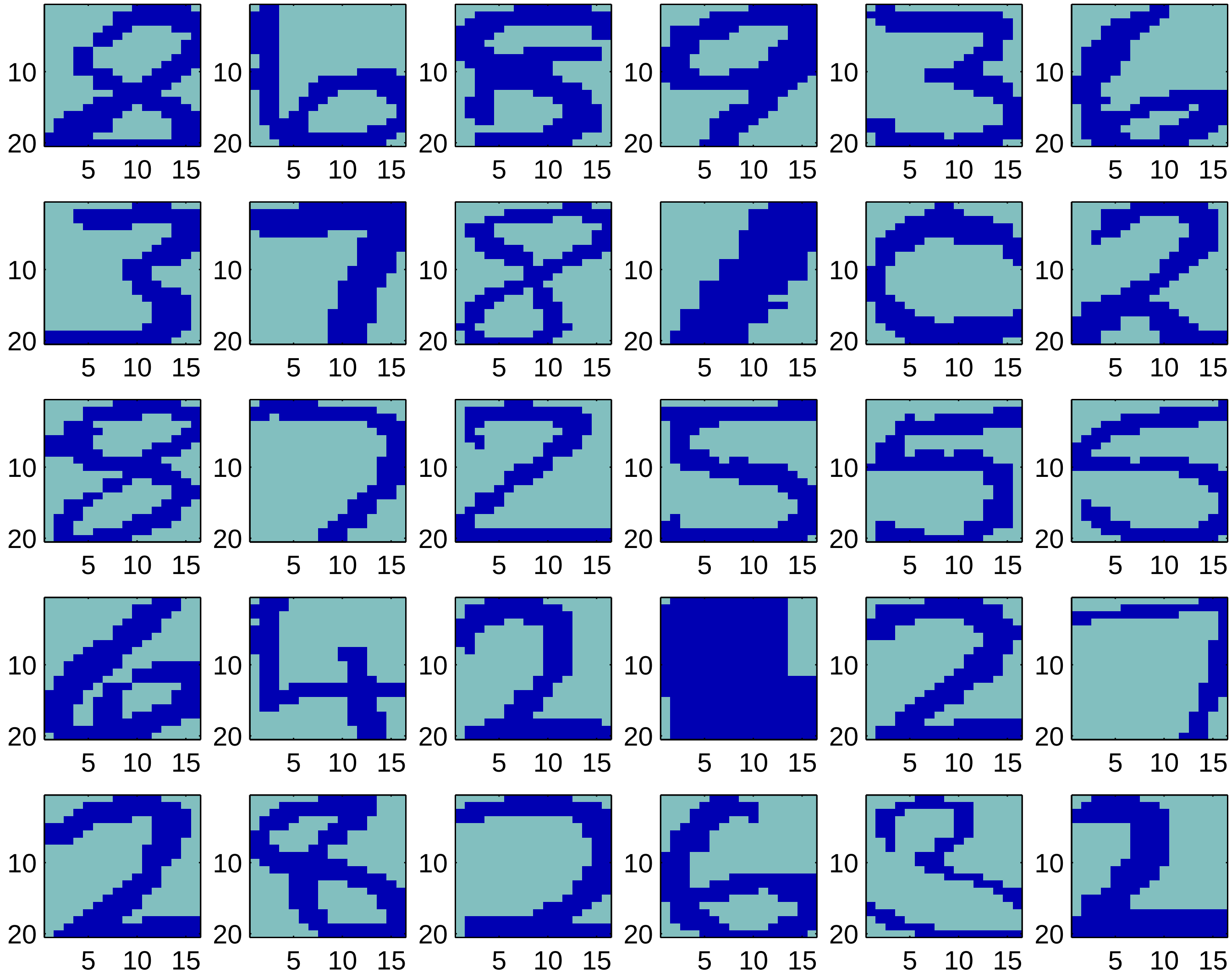
"Unsupervised learning" : methods do not exploit labeled data

- ➤ 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

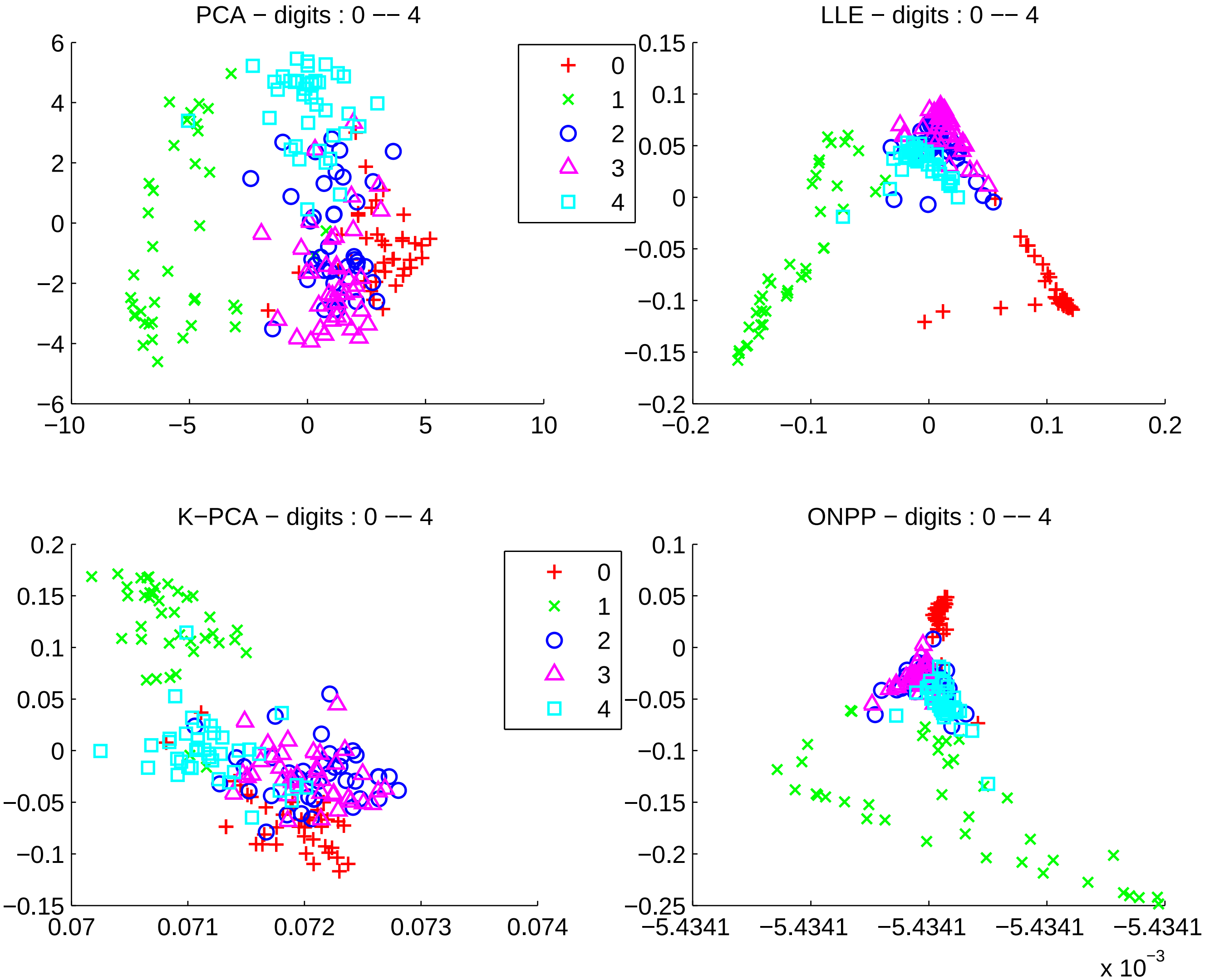




Example: Digit images (a random sample of 30)



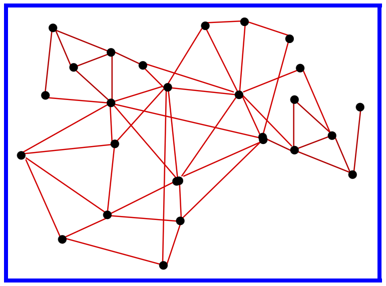
2-D 'reductions':



GRAPH PARTITIONING → CLUSTERING

Graph Laplaceans - Definition

➤ "Laplace-type" matrices associated with general undirected graphs –



$\longrightarrow L = \begin{bmatrix} & ? & \end{bmatrix}$

➤ Given a graph $G = (V, E)$ define

- A matrix W of weights w_{ij} for each edge with:

$$\boxed{w_{ij} \geq 0, \quad w_{ii} = 0, \quad \text{and} \quad w_{ij} = w_{ji} \, \forall (i,j)}$$

- The diagonal matrix $D = diag(d_i)$ with $d_i = \sum_{j \neq i} w_{ij}$

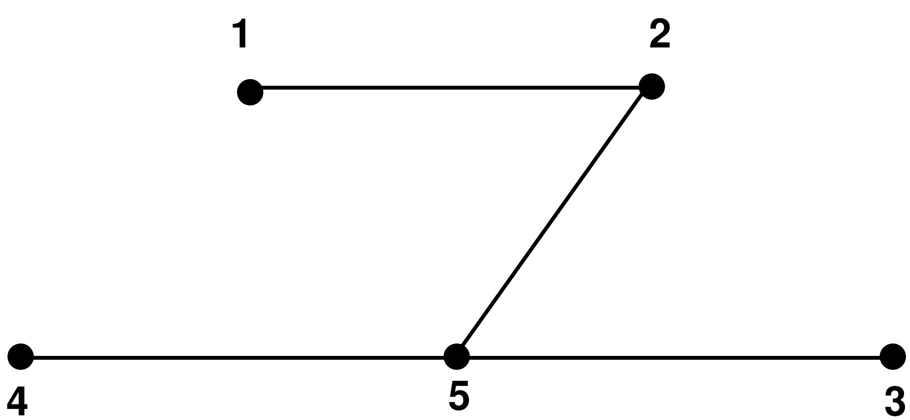
➤ Corresponding *graph Laplacean* of G is:

$$L = D - W$$

- Gershgorin's theorem $\rightarrow$ L is positive semidefinite.
- One eigenvalue equal to zero
- Simplest case:

$$w_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \ \& \ i \neq j \\ 0 \quad \text{else} \end{cases} \qquad d_i = \sum_{j \neq i} w_{ij}$$

Example: Consider the graph



$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{pmatrix}$$

Basic results on graph Laplaceans

Proposition:

(i) L is symmetric semi-positive definite.

(ii) L is singular with $\mathbb{1}$ as a null vector.

(iii) If G is connected, then $\mathbf{Null}(L) = \mathbf{span}\{\ \mathbb{1}\}$

(iv) If G has $k > 1$ connected components $G_1, G_2, \cdots, G_k$, then the nullity of L is k and $\mathbf{Null}(L)$ is spanned by the vectors $z^{(j)}$, $j = 1, \cdots, k$ defined by:

$$(z^{(j)})_i = \begin{cases} 1 \text{ if } i \in G_j \\ 0 \text{ if not.} \end{cases}$$

A few properties of graph Laplaceans

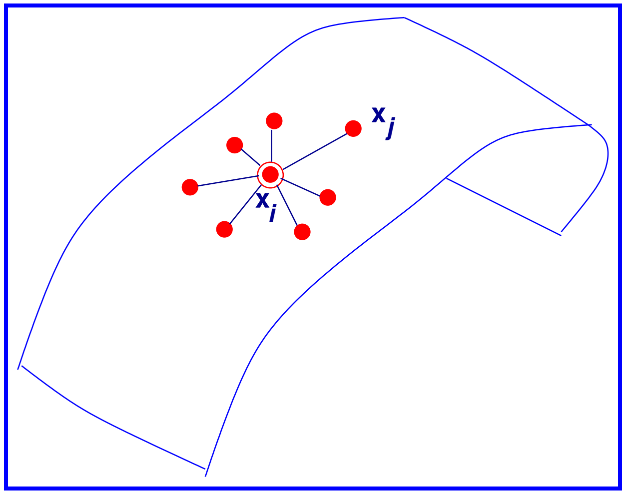
Define: oriented incidence matrix H: (1)First orient the edges $i \sim j$ into $i \to j$ or $j \to i$. (2) Rows of H indexed by vertices of G. Columns indexed by edges. (3) For each (i, j) in E, define the corresponding column in H as $\sqrt{w(i, j)}(e_i - e_j)$.

Example: In previous example, orient $i \to j$ so that $j > i$ [lower triangular matrix representation]. Then matrix L is: $\longrightarrow$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 \end{pmatrix}$$

Property 1 $\boxed{L = HH^T}$

A few properties of graph Laplaceans



Strong relation between $x^T L x$ and local distances between entries of x

➤ Let L = any graph Laplacean

Then:

Property 2: for any $x \in \mathbb{R}^n$: [Recall $L = D - W$]

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

Property 3: (generalization) for any $Y \in \mathbb{R}^{d\times n}$:

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

➤ Note: y_j = j-th colunm of Y. Usually $d < n$. Each column can represent a data sample.

Property 4: For the particular $L = I - \frac{1}{n}\mathbb{1}\,\mathbb{1}^\top$

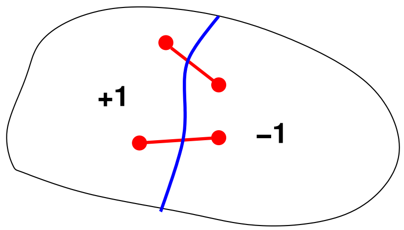
$$XLX^\top = \bar{X}\bar{X}^\top == n \times \text{Covariance matrix}$$

Property 5: L is singular and admits the null vector $\mathbb{1}$ =`ones(n,1)`

Property 6: (Graph partitioning) Consider situation when $w_{ij} \in \{0,1\}$. If x is a vector of signs (± 1) then

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

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



- Consequence: Can be used to partition graphs....

- ...by minimizing (Lx, x) subject to $x \in \{-1, 1\}^n$ and $\mathbb{1}^T x = 0$ [balanced sets]

$$\min_{x \in \{-1,1\}^n;\ \mathbb{1}^T x = 0} \frac{(Lx, x)}{(x, x)}$$

- This problem is hard [combinatorial] $\rightarrow$

- Instead we solve a relaxed form of this problem :

$$\min_{x\in\{-1,1\}^n;\ \mathbb{1}^T x=0} \frac{(Lx,x)}{(x,x)} \quad \to \quad \min_{x\in\mathbb{R}^n;\ \mathbb{1}^T x=0} \frac{(Lx,x)}{(x,x)}$$

- Define $v = u_2$ then $lab = sign(v - med(v))$

Background:

- Consider any symmetric (real) matrix A with eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ and eigenvectors $u_1, \cdots, u_n$

- Recall that:

(Min reached for $x = u_1$)

$$\min_{x\in\mathbb{R}^n} \frac{(Ax,x)}{(x,x)} = \lambda_1$$

➤ In addition:
(Min reached for $x = u_2$)

$$\min_{x \perp u_1} \frac{(Ax, x)}{(x, x)} = \lambda_2$$

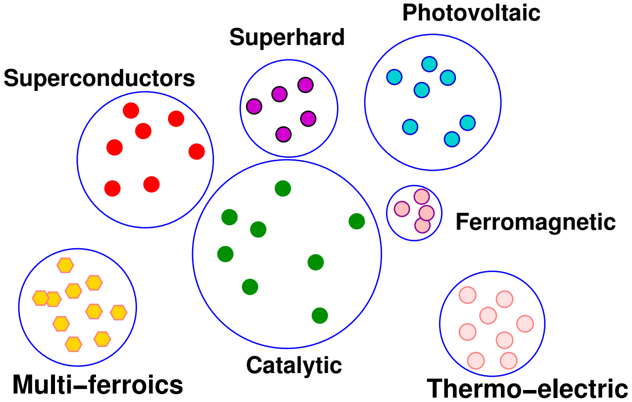
➤ For a graph Laplacean $u_1 = \mathbb{1}$ = vector of all ones and

➤ ...vector u_2 is called the Fiedler vector. It solves the relaxed optimization problem -

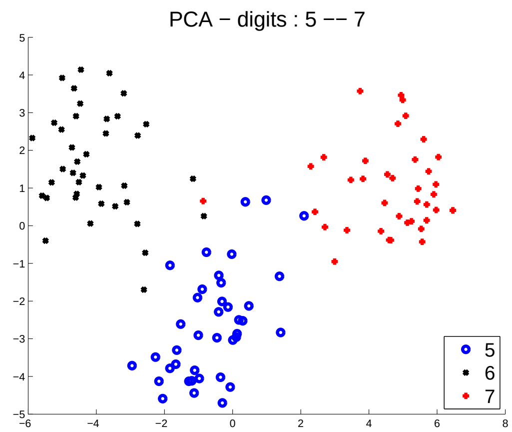
Clustering

➤ Problem: we are given n data items: $x_1, x_2, \cdots, x_n$. Would like to *'cluster'* them, i.e., group them so that each group or cluster contains items that are similar in some sense.

➤ Example: materials



➤ Example: Digits

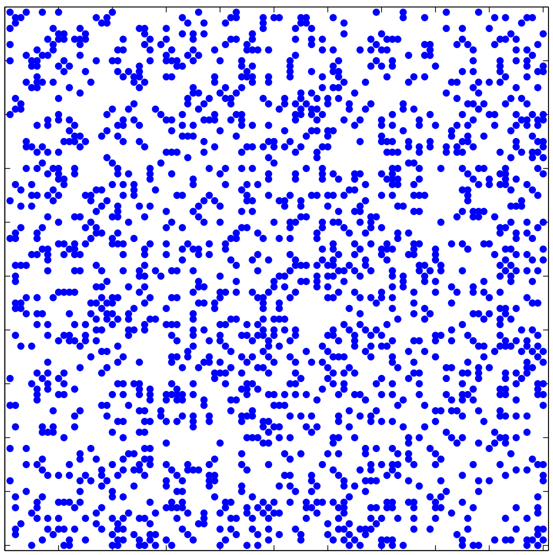


➤ Refer to each group as a 'cluster' or a 'class'

➤ Unsupervised learning

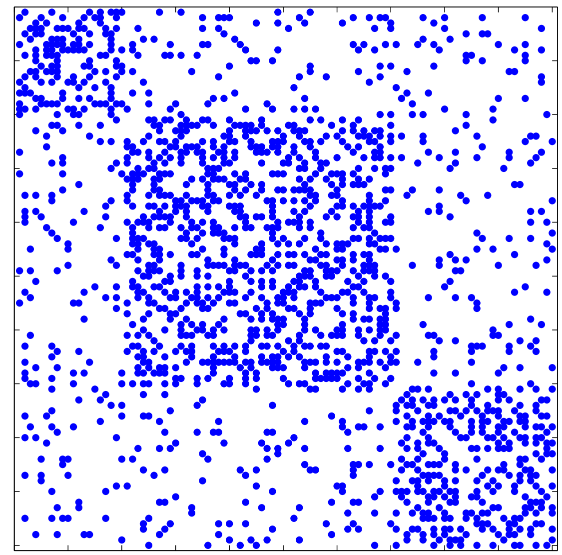
Example: Block structure detection $\to$ Community Detection

- Communities modeled by an 'affinity' graph [e.g., 'user A sends frequent e-mails to user B']
- Adjacency Graph represented by a sparse matrix



$\leftarrow$ Original matrix

Goal: Find ordering so blocks are as dense as possible $\rightarrow$



- Use 'blocking' techniques for sparse matrices
- Advantage of this viewpoint: need not know # of clusters.

[data: `www-personal.umich.edu/~mejn/netdata/`]

Example of application 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): 758 liberal, 732 conservative. Edge: $i \to j$: a citation between blogs i and j
- Used blocking algorithm (Density theshold=0.4): subgraphs [note: density = $|E|/|V|^2$.]
- Details in [J. Chen & YS, IEEE TKDE (24), 2012]

A basic clustering method: K-means (Background)

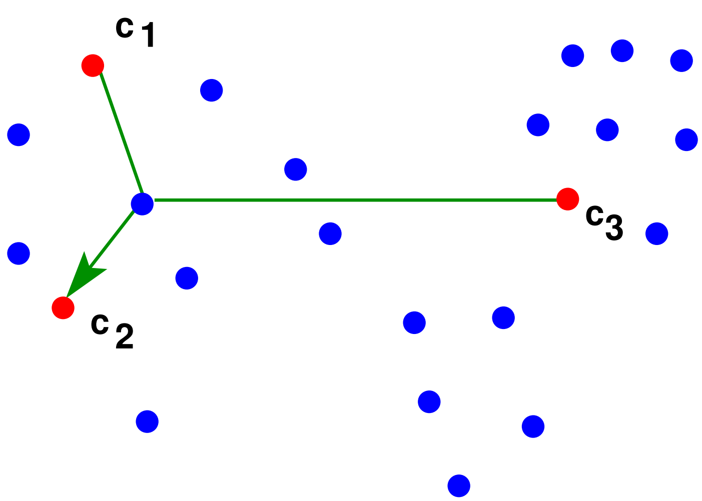
➤ A basic algorithm that uses Euclidean distance

1 Select p initial centers: $c_1, c_2, ..., c_p$ for classes $1, 2, \cdots, p$

2 For each x_i do: determine *class* of x_i as $\text{argmin}_k \|x_i - c_k\|$

3 Redefine each c_k to be the centroid of class k

4 Repeat until convergence



➤ Simple algorithm

➤ Works well (gives good results) but can be slow

➤ Performance depends on initialization

Clustering methods based on similarity graphs

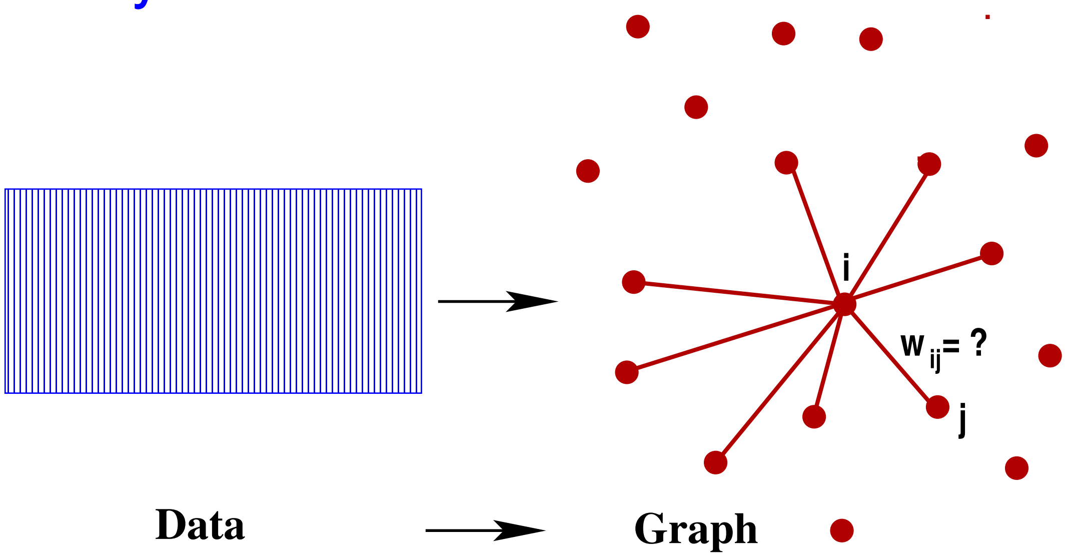
- Perform clustering by exploiting a graph that describes the similarities between any two items in the data.
- Need to:

1. decide what nodes are in the neighborhood of a given node
2. quantify their similarities - by assigning a weight to any pair of nodes.

Example: For text data: Can decide that any columns i and j with a cosine greater than 0.95 are 'similar' and assign that cosine value to w_{ij}

First task: build a 'similarity' graph

Need: a similarity graph, i.e., a graph that captures the similarity between any two items



➤ For each data item find a small number of its nearest neighbors

- Two techniques are often used:

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

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.
- See [J. Chen, H-R Fang, YS, JMLR, 2009] for algorithms.

Similarity graphs: Using 'heat-kernels'

Define weight between i and j as:

$$w_{ij} = f_{ij} \times \begin{cases} e^{\frac{-\|x_i - x_j\|^2}{\sigma_X^2}} & \text{if } \|x_i - x_j\| < r \\ 0 & \text{if not} \end{cases}$$

- Note $\|x_i - x_j\|$ could be any measure of distance...
- f_{ij} = optional = some measure of similarity - other than distance
- Only nearby points kept.
- Sparsity depends on parameters

Edge cuts, ratio cuts, normalized cuts, ...

- Assume now that we have built a 'similarity graph'
- Setting is identical with that of graph partitioning.
- Need a Graph Laplacean: $L = D - W$ with $w_{ii} = 0, w_{ij} \geq 0$ and $D = diag(W * ones(n, 1))$ [matlab]
- Partition vertex set V in two sets A and B with

$$A \cup B = V, \quad A \cap B = \emptyset$$

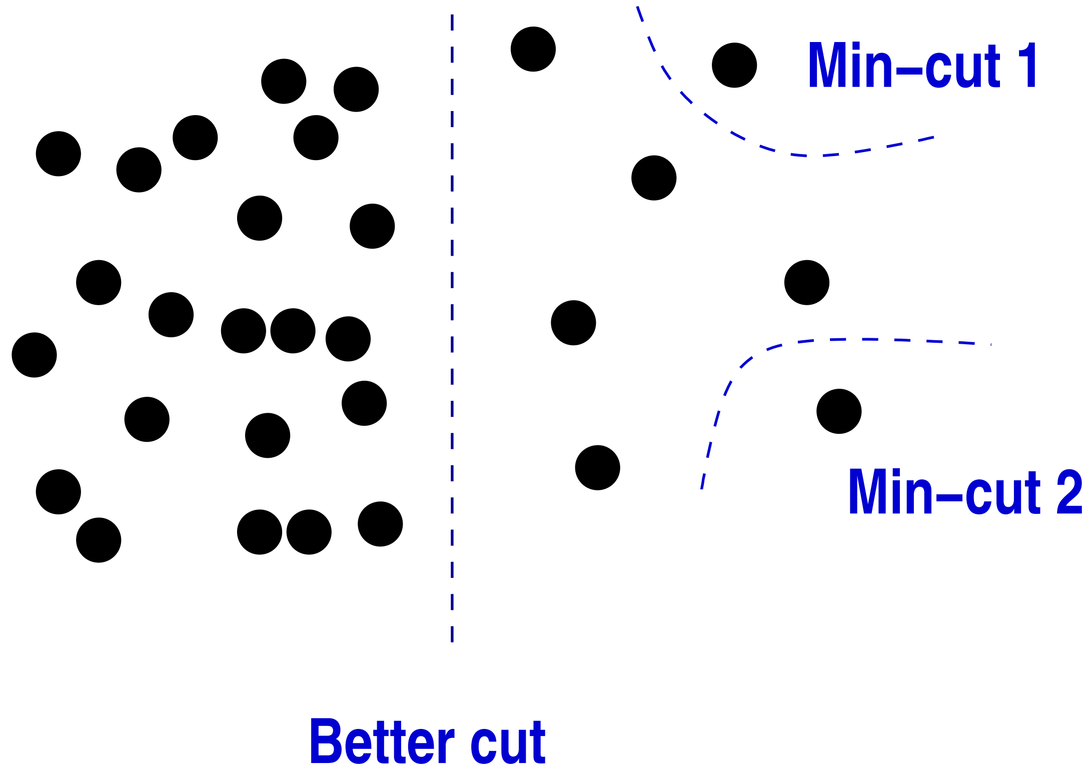
- Define

$$\boxed{cut(A, B) = \sum_{u \in A, v \in B} w(u, v)}$$

- Naive approach: use this measure to partition graph, i.e.,

... Find A and B that minimize $cut(A, B)$.

- Issue: Small sets, isolated nodes, big imbalances,



Normalized cuts [Shi-Malik,2000]

➤ Recall notation $w(X, Y) = \sum_{x \in X, y \in Y} w(x, y)$ - then define:

$$\boxed{\text{ncut}(A, B) = \frac{cut(A,B)}{w(A,V)} + \frac{cut(A,B)}{w(B,V)}}$$

➤ Goal is to avoid small sets A, B

➤ Let x be an indicator vector:

$$x_i = \begin{cases} 1 \;\; if \;\; i \in A \\ 0 \;\; if \;\; i \in B \end{cases}$$

➤ Recall that: $x^T L x = \sum_{(i,j) \in E} w_{ij} |x_i - x_j|^2$ (note: each edge counted once)

- Let
$$\beta = \frac{w(A,V)}{w(B,V)} = \frac{x^T D \mathbb{1}}{(\mathbb{1} - x)^T D \mathbb{1}}$$
$$y = x - \beta(\mathbb{1} - x)$$

- Then we need to solve:
$$\min_{y_i \; \{0,-\beta\}} \frac{y^T L y}{y^T D y}$$
$$\text{Subject to} \quad y^T D \mathbb{1} = 0$$

- + Relax $\rightarrow$ need to solve Generalized eigenvalue problem

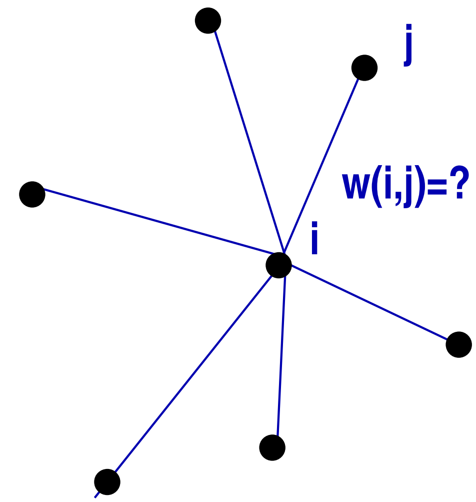
$$Ly = \lambda D y$$

- $y_1 = \mathbb{1}$ is eigenvector associated with eigenvalue $\lambda_1 = 0$
- y_2 associated with second eigenvalue solves problem.

Spectral clustering: General approach

1 Given: Collection of data samples $\{x_1, x_2, \cdots, x_n\}$

2 Build a similarity graph between items



3 Compute (smallest) eigenvector (s) of resulting graph Laplacean

4 Use k-means on eigenvector (s) of Laplacean

- For normalized cuts solve generalized eigen problem.
- Application: Image segmentation

DEMO

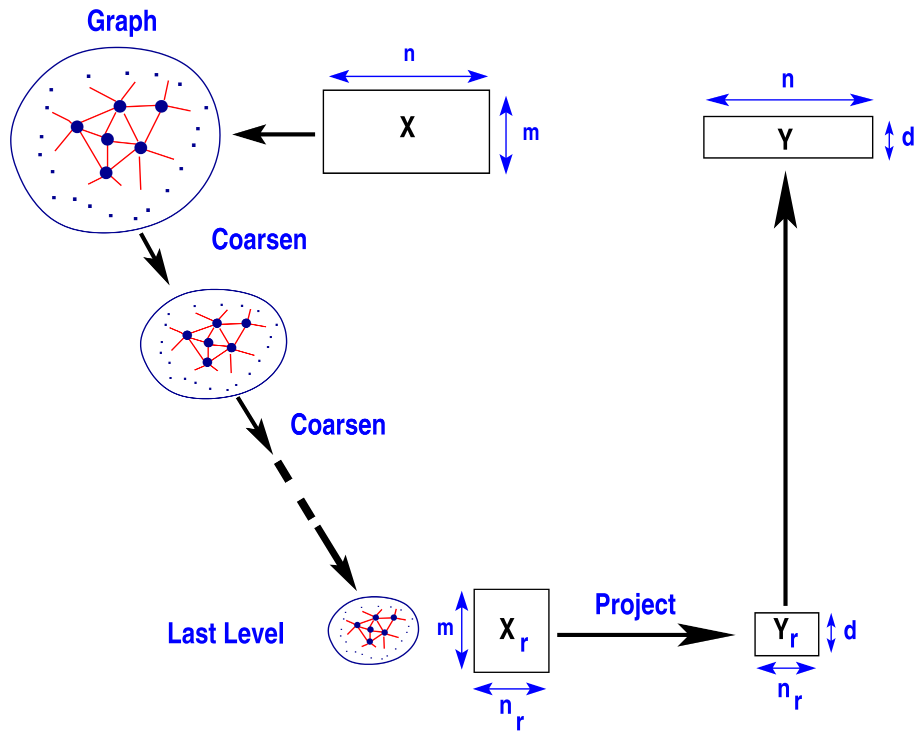
AMG $\rightarrow$ DATA COARSENING

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

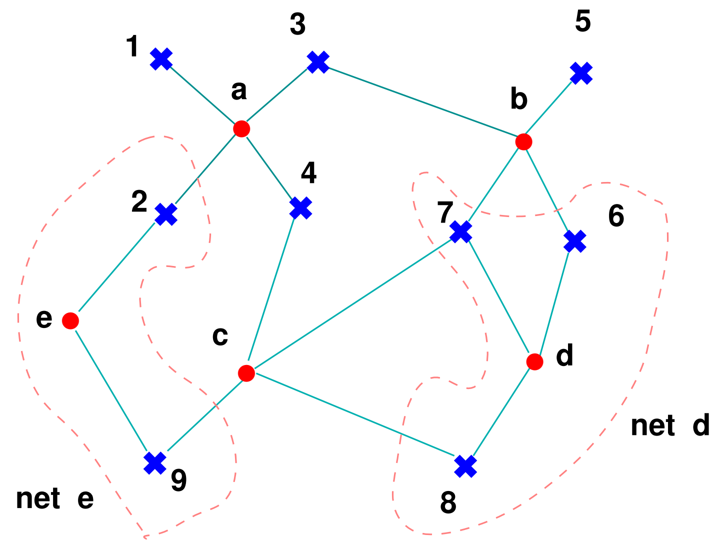
Main Idea: coarsen for a few levels. Use the resulting data set $\hat{X}$ to find a projector P from $\mathbb{R}^m$ to $\mathbb{R}^d$. P can be used to project original data or new data



➤ 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

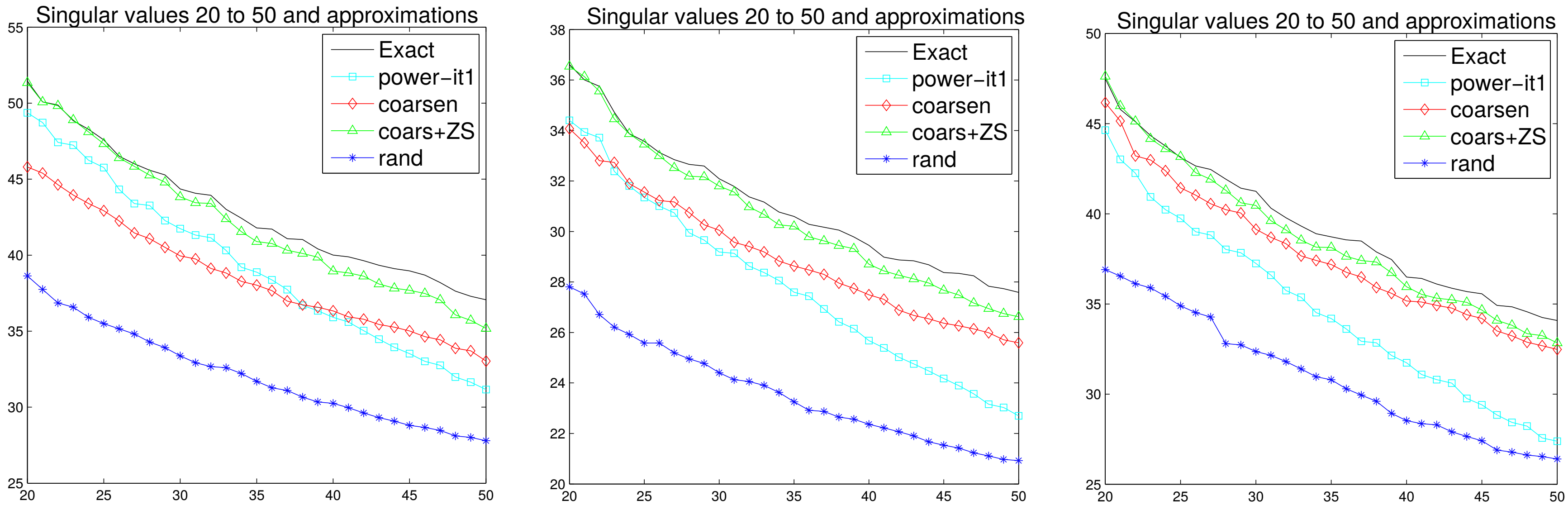
$$|x^T A A^T x - x^T C C^T x| \le 3\epsilon \|A\|_F^2,$$

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).

➤ See [S. Ubaru, YS. NLAA, 2018]

LANCZOS FOR EIGENVALUES $\rightarrow$ LANCZOS FOR DIM. REDUCTION

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

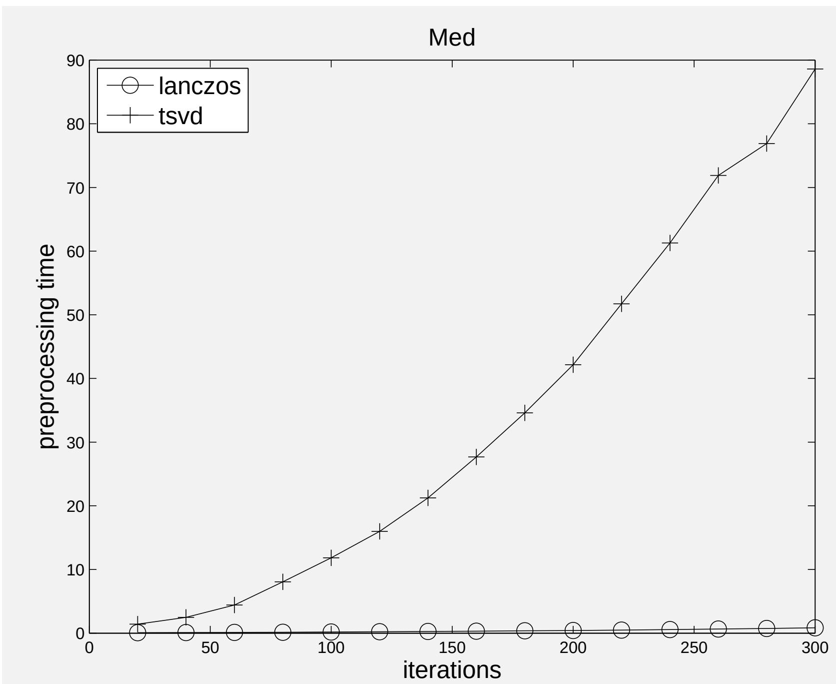
Tests: IR

Information retrieval datasets

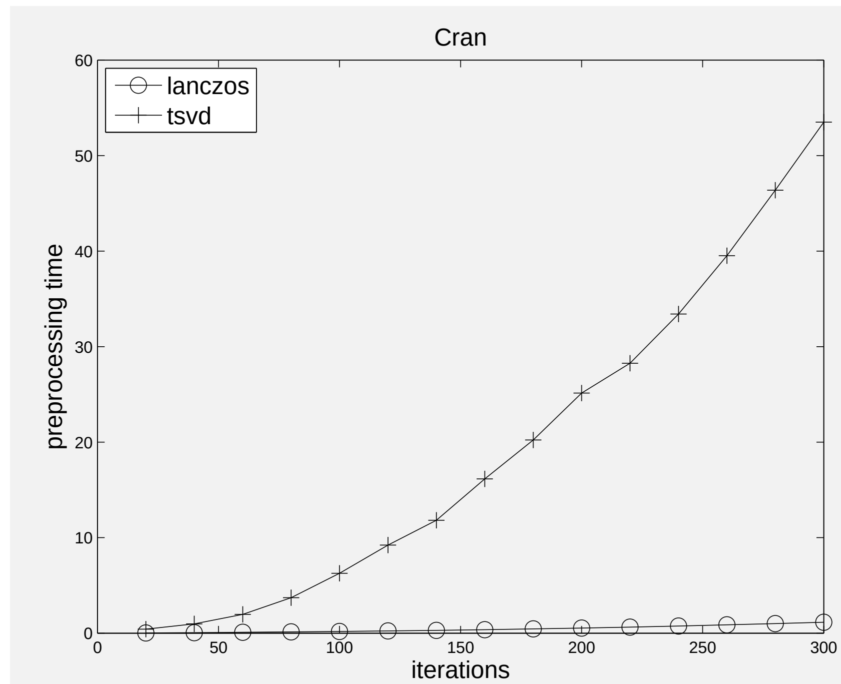
	# Terms	# Docs	# queries	sparsity
MED	7,014	1,033	30	0.735
CRAN	3,763	1,398	225	1.412

Preprocessing times

Med dataset.

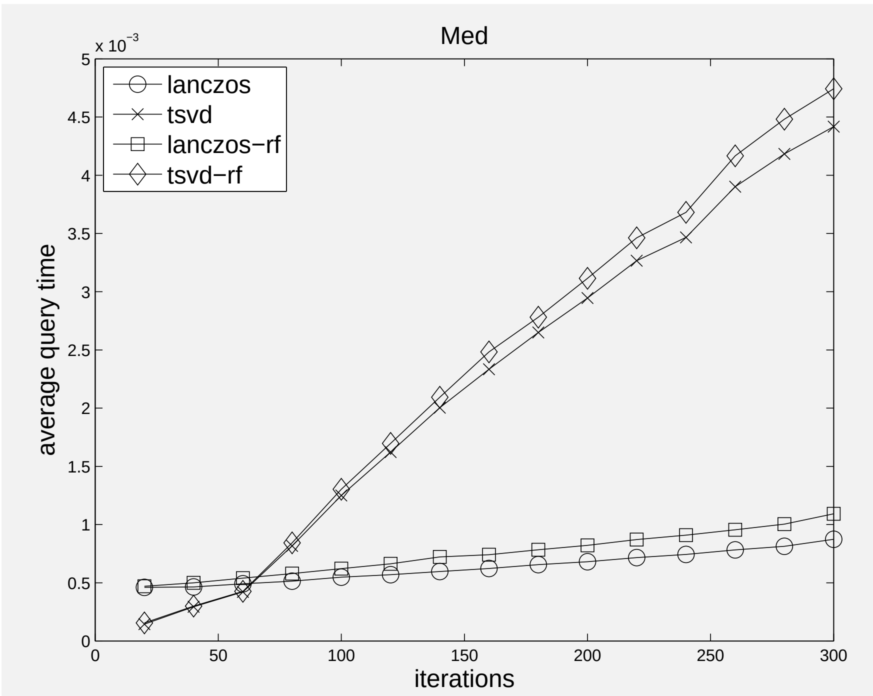


Cran dataset.

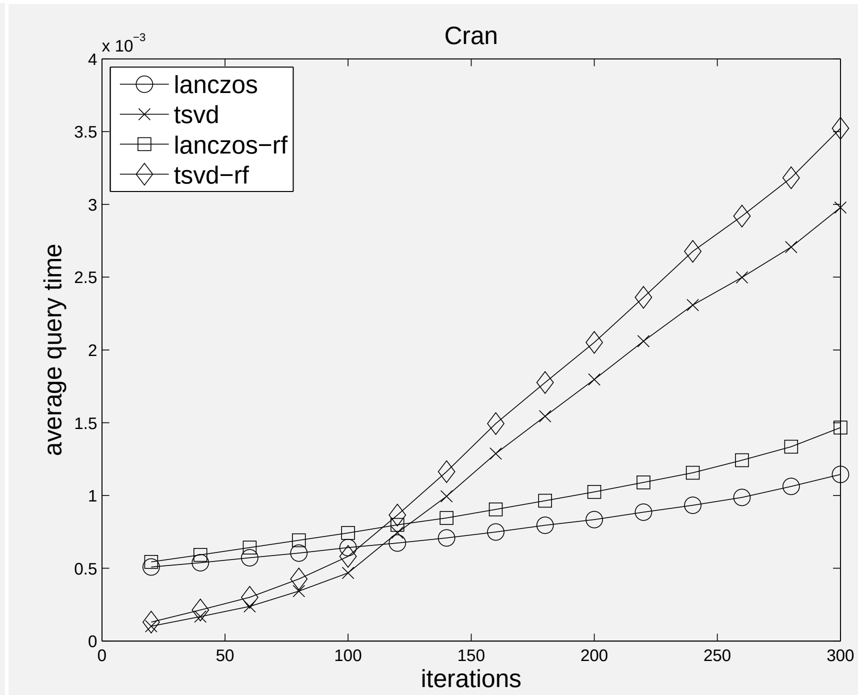


Average query times

Med dataset



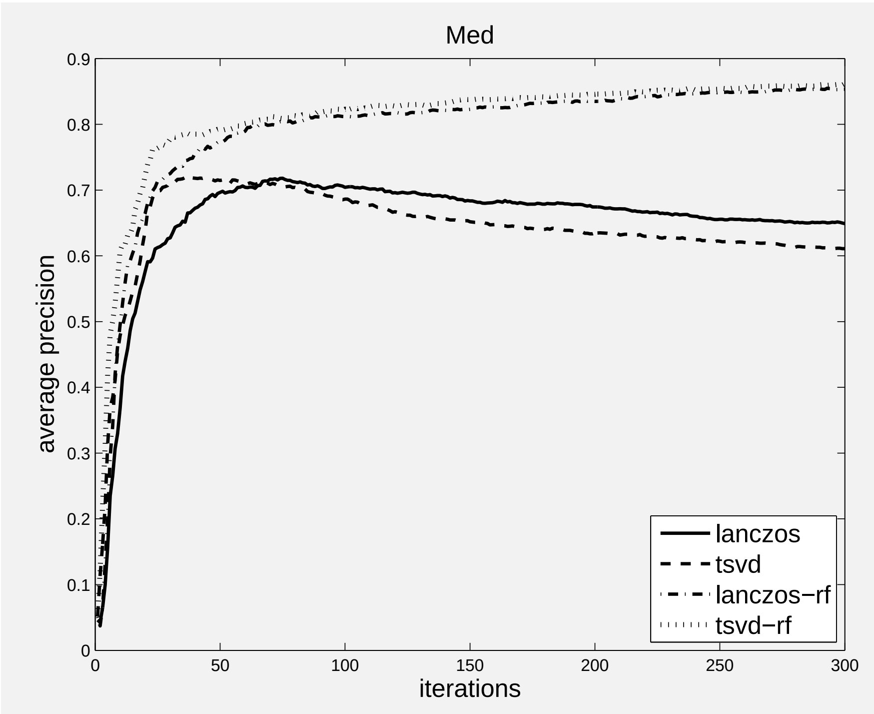
Cran dataset.

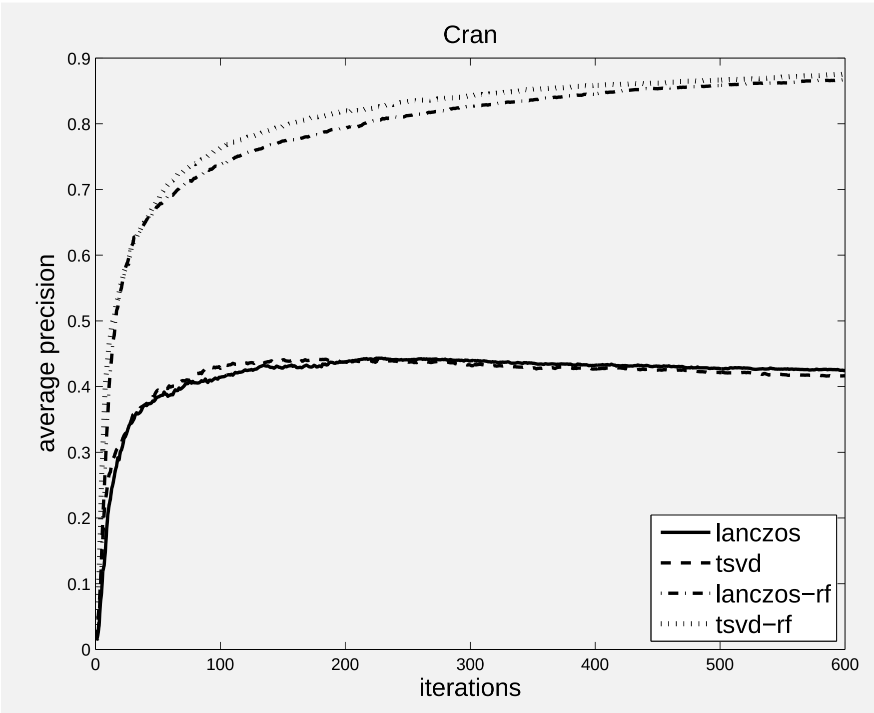


Average retrieval precision

Med dataset

Cran dataset





Retrieval precision comparisons

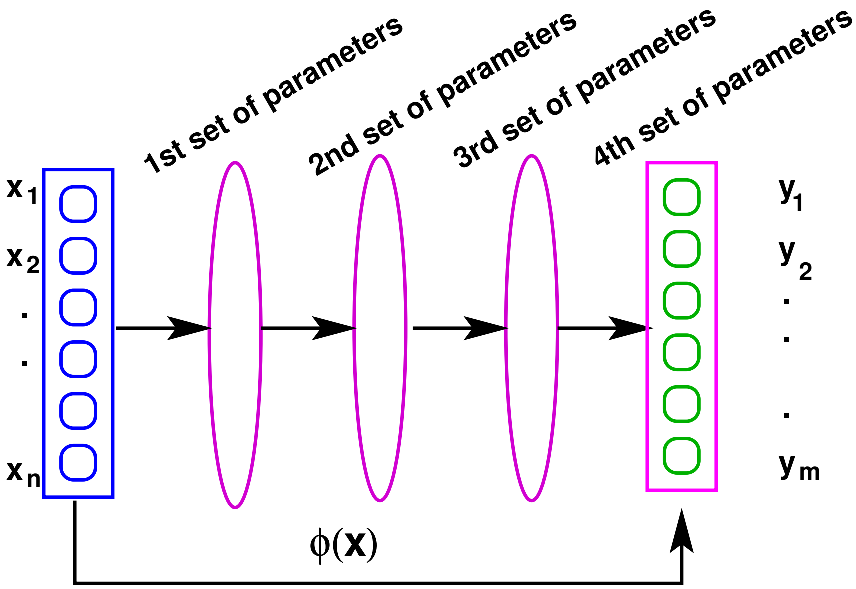
A few words on Deep Neural Networks (DNNs)

➤ Ideas of neural networks goes back to the 1960s - were popularized in early 1990s – then laid dormant until recently.

➤ Two reasons for the come-back:

- DNN are remarkably effective in some applications
- big progress made in hardware [$\longrightarrow$ affordable 'training cost']

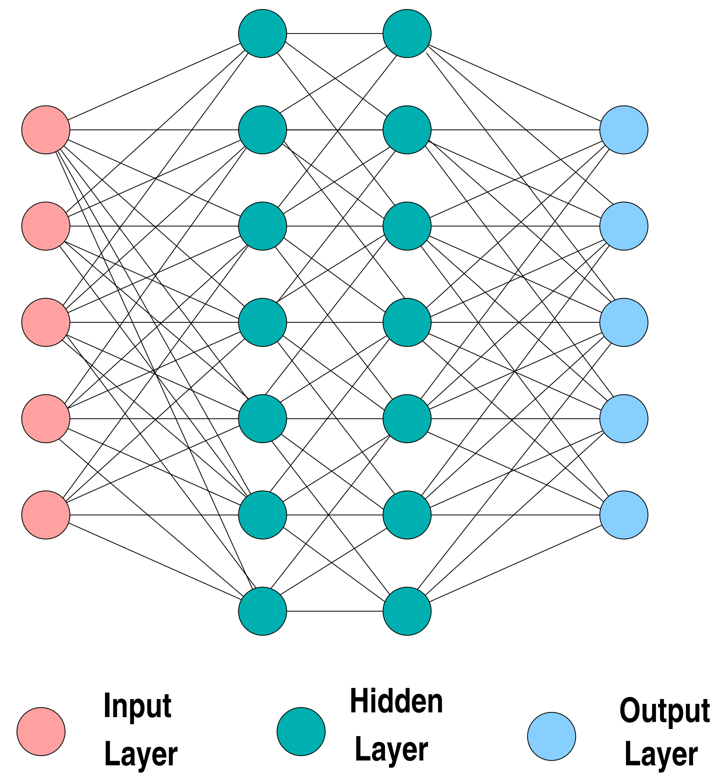
➤ Training a neural network can be viewed as a problem of approximating a function ϕ which is defined via sets of parameters:



Problem: find sets of parameters such that $\phi(x) \approx y$

Input: x, **Output:** y
Set: $z_0 = x$
For $l = 1 :$ `L+1` **Do:**
$$z_l = \sigma(W_l^T z_{l-1} + b_l)$$
End
Set: $y = \phi(x) := z_{L+1}$

- layer # 0 = input layer
- layer # $(L+1)$ = output layer



➤ A matrix W_l is associated with layers 1,2, $L+1$.

➤ Problem: Find ϕ (i.e., matrices W_l) s.t. $\phi(x) \approx y$

DNN (continued)

- Problem is not convex, highly parameterized, ...,
- .. Main method used: Stochastic gradient descent [basic]
- It all looks like alchemy... but it works well for certain applications
- Training is still quite expensive – GPUs can help
- *Very* active area of research

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 :

In the words of “Who Moved My Cheese?” [Spencer Johnson, 2002]

“If you do not change, you can become extinct !”

- In the words of Einstein:

“Life is like riding a bicycle: To keep your balance you need to keep moving”

Thank you !

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