Efficient Linear Algebra Methods in Data Mining

Yousef Saad
University of Minnesota
Dept. of Computer Science and Engineering

M2A07 – Luminy, France

Introduction and Background:

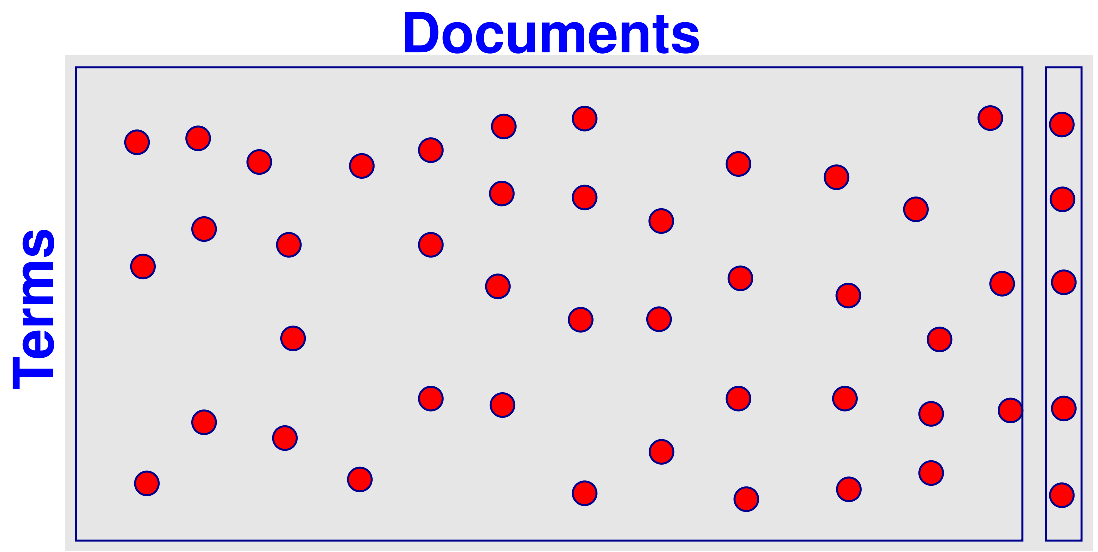
- Information sciences : Data Mining, Data Analysis, Machine Learning, Classification, ... are a huge source of interesting matrix problems

- Effective linear algebra methods are just starting to be deployed

- In this talk 3 sample problems:
  1. Information retrieval
  2. Face recognotion
  3. Clustering

Information Retrieval: Vector Space Model

Given: 1) set of documents (columns of a matrix A); 2) a query vector q. Entry a_{ij} of A = frequency of term i in document j + weighting.



➤ Queries ('pseudo-documents') q represented similarly to columns

Problem: find columns of A that best match q

Vector Space Model and the Truncated SVD

- **Similarity metric: angle between column $A_{j,:}$ and query q**

Use Cosines:

$$\frac{|q^T A_{:,j}|}{\|A_{:,j}\|_2 \|q\|_2}$$

- **To rank all documents compute the similarity vector:**

$$s = A^T q$$

- **'Litteral' matching – not very effective. Problems : *polysemy*, *synonymy*, ...**

- **LSI: replace matrix A by low rank approximation**

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

- **U_k : term space, V_k: document space.**

- **Called TSVD – Expensive, hard to update, ..**

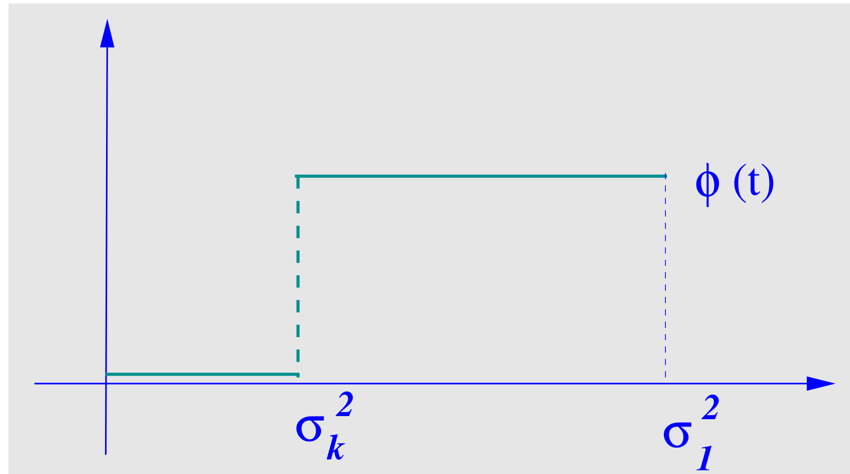
IR: Use of approximation theory

- **Use of polynomial filters * Joint work with E. Kokiopoulou**

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

- **Consider the step-function:**

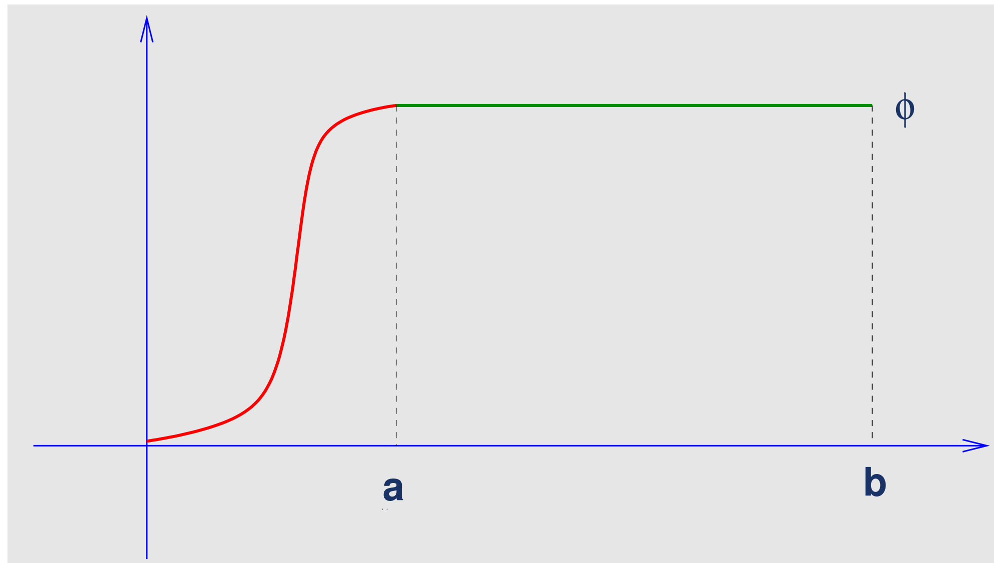
$$\phi(x) = \begin{cases} 0, & 0 \le x \le \sigma_k^2 \\ 1, & \sigma_k^2 \le x \le \sigma_1^2 \end{cases}$$



- **This would yield the same result as with TSVD but...**
- **... Not easy to use this function directly**
- **Solution : use a polynomial approximation to ϕ**
- **Note: $\boxed{s^T = q^TA\phi(A^TA)}$, requires only Mat-Vec's**

How to get the polynomial filter?

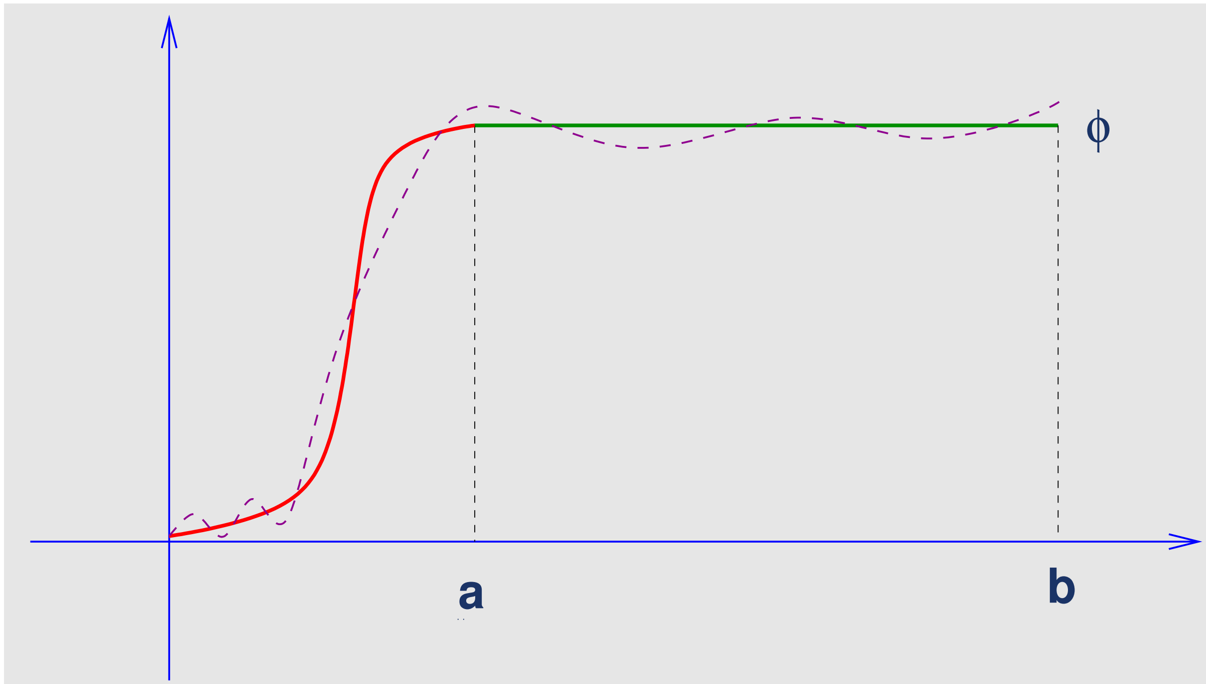
Idea: **First select an "ideal fiter"**



- **e.g. a piecewise polynomial function**

- **For example $\phi =$ Hermite interpolating pol. in [0,a], and $\phi = 1$ in [a, b]**

➤ **Then approximate this filter by an 'optimal' (least-squares) polynomial**



Main advantage: **Extremely flexible.**

Method: **Build a sequence of polynomials ϕ_k which approximate the ideal PP filter ϕ, in the L_2 sense.**

- If $\{\mathcal{P}_j\}$ is a basis of polynomials that are orthogomal w.r.t. some L_2 inner-product, then

$$\phi_k(t) = \sum_{j=1}^{k} \langle \phi, \mathcal{P}_j \rangle \mathcal{P}_j(t),$$

- Can use Stieljes procedure to compute orthogonal polynomials [Erhel, Guyomarch, YS'99]
- Or can use a Conjugate residual-type algorithm in polynomial space [YS'05, Bekas-Kokiopoulou-YS'05]
- Accuracy close to that of TSVD – But no SVD required
- Experiments and details skipped.

IR: Use of the Lanczos algorithm

* Joint work with Jie Chen – in progress

➤ Lanczos is good at catching large (and small) eigenvalues: can compute singular vectors with Lanczos, & use them in LSI

➤ Can do better: Use the Lanczos vectors directly for the projection..

➤ First advocated by: K. Blom and A. Ruhe [SIMAX, vol. 26, 2005]. Use Lanczos bidiagonalization.

➤ Use a similar approach – But directly with AA^T or A^TA.

IR: Use of the Lanczos algorithm (1)

- **Let $A \in \mathbb{R}^{m\times n}$. Apply the Lanczos procedure to $M = AA^T$.**

Result:

$$Q_k^T AA^T Q_k = T_k$$

with Q_k orthogonal, T_k tridiagonal.

- **Define $s_i \equiv$ orth. projection of Ab on subspace span$\{Q_i\}$**

$$s_i := Q_i Q_i^T Ab.$$

- **s_i can be easily updated from s_{i-1}:**

$$s_i = s_{i-1} + q_i q_i^T Ab.$$

IR: Use of the Lanczos algorithm (2)

- If $n < m$ it may be more economial to apply Lanczos to $M = A^TA$ which is $n \times n$. Result:

$$\bar{Q}_k^T A^T A \bar{Q}_k = \bar{T}_k$$

- Define:

$$t_i := A\bar{Q}_i\bar{Q}_i^T b,$$

- Project b first before applying A to result.

Why does this work?

- **First, recall a result on Lanczos algorithm [YS 83]**

Let $\{\lambda_j, u_j\}$ = j-th eigen-pair of M (label ↓)

$$\frac{\|(I - Q_k Q_k^T) u_j\|}{\|Q_k Q_k^T u_j\|} \leq \frac{K_j}{T_{k-j}(\gamma_j)} \frac{\|(I - Q_1 Q_1^T) u_j\|}{\|Q_1 Q_1^T u_j\|},$$

where

$$\gamma_j = 1 + 2\frac{\lambda_j - \lambda_{j+1}}{\lambda_{j+1} - \lambda_n}, \qquad K_j = \begin{cases} 1 & j = 1 \\ \prod_{i=1}^{j-1} \frac{\lambda_i - \lambda_n}{\lambda_i - \lambda_j} & j \neq 1 \end{cases},$$

and $T_l(x)$ = Chebyshev polynomial of 1st kind of degree l.

This has the form

$$\|(I - Q_k Q_k^T) u_j\| \leq c_j / T_{k-j}(\gamma_j),$$

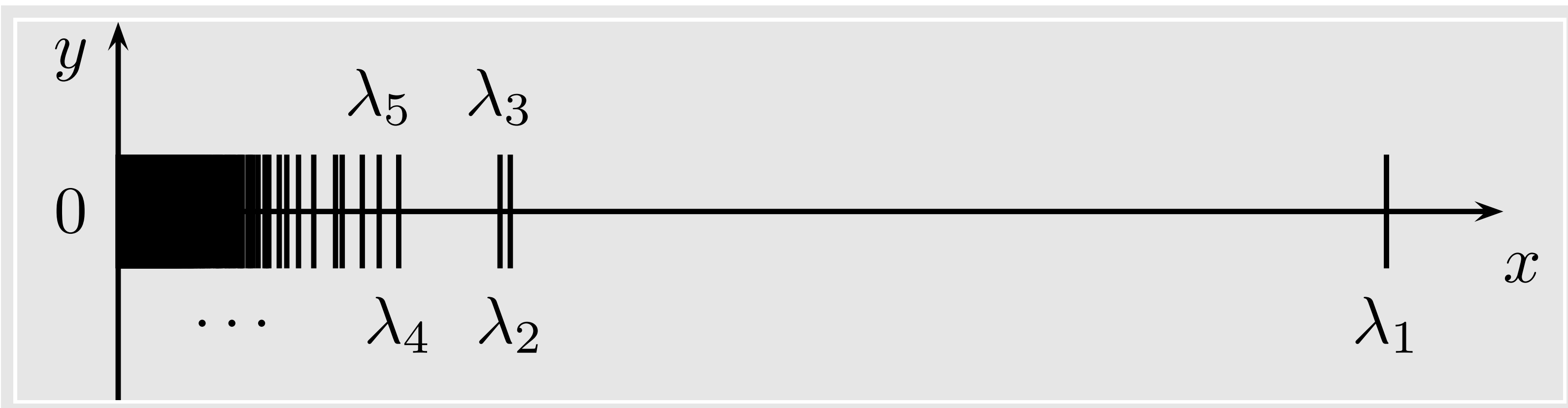
where c_j = constant independent of k

- **Result: Distance between unit eigenvector u_j and Krylov subspace span(Q_k) decays fast (for small j)**
- **Consider component of difference between $Ab - s_k$ along left singular directions of A. If $A = U\Sigma V^T$, then u_j's (columns of U) are eigenvectors of $M = AA^T$. So:**

$$\begin{aligned} |\langle Ab - s_k, u_j \rangle| &= \left|\langle (I - Q_k Q_k^T) Ab, u_j \rangle\right| \\ &= \left|\langle (I - Q_k Q_k^T) u_j, Ab \rangle\right| \\ &\leq \|(I - Q_k Q_k^T) u_j\| \|Ab\| \\ &\leq c_j \|Ab\| T_{k-j}^{-1}(\gamma_j) \end{aligned}$$

- **$\{s_i\}$ converges rapidly to Ab in directions of the major left singular vectors of A.**

- Similar result for left projection sequence t_j
- Here is a typical distribution of eigenvalues of M: [Matrix of size 1398×1398]



- Convergence toward first few singular vectors very fast –

Advantages of Lanczos over polynomial filters:

(1) No need for eigenvalue estimates

(2) Mat-vecs performed only in preprocessing

Disadvantages:

(1) Need to store Lanczos vectors;

(2) Preprocessing must be redone when A changes.

(3) Need for reorthogonalization – expensive for large k.

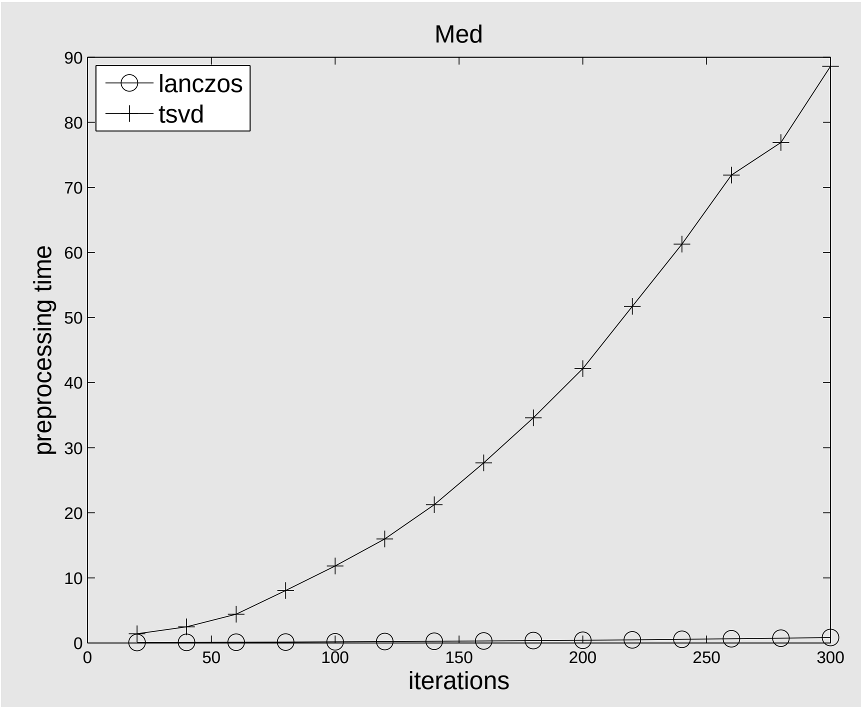
Tests: IR

Information retrieval datasets

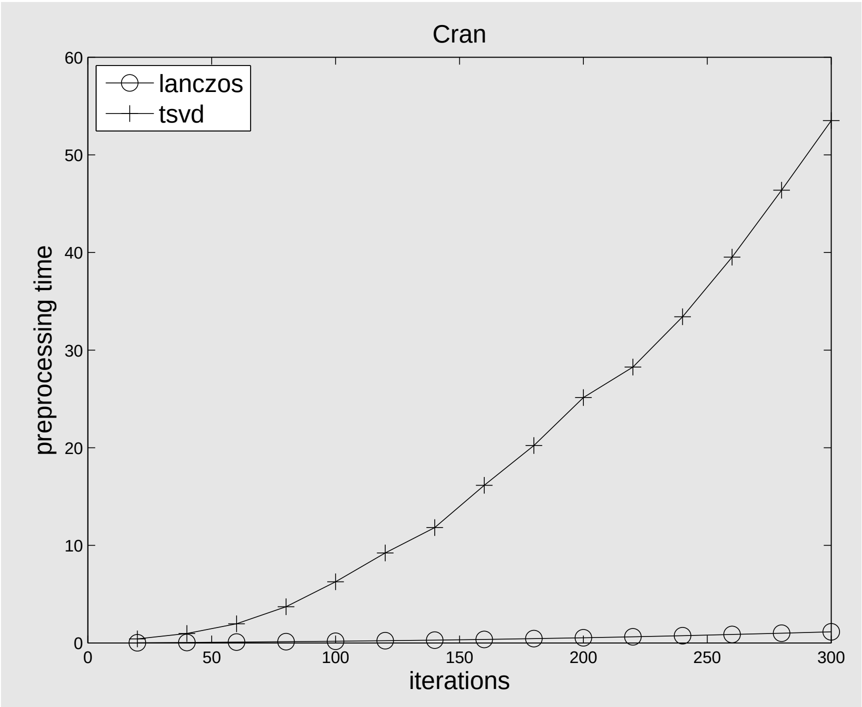
	# Terms	# Docs	# queries	sparsity
MED	7,014	1,033	30	0.735
CRAN	3,763	1,398	225	1.412

Preprocessing times

Med dataset.

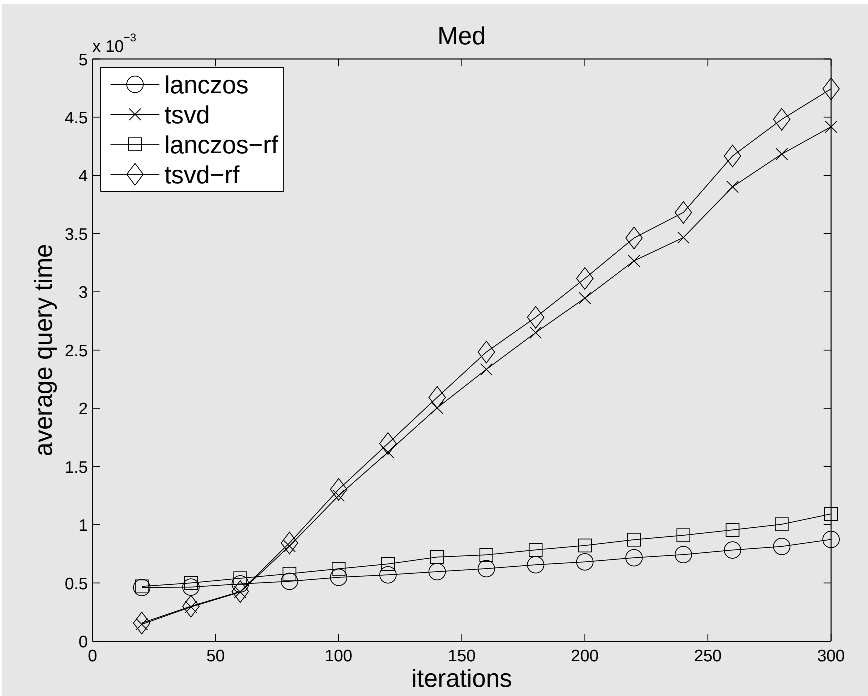


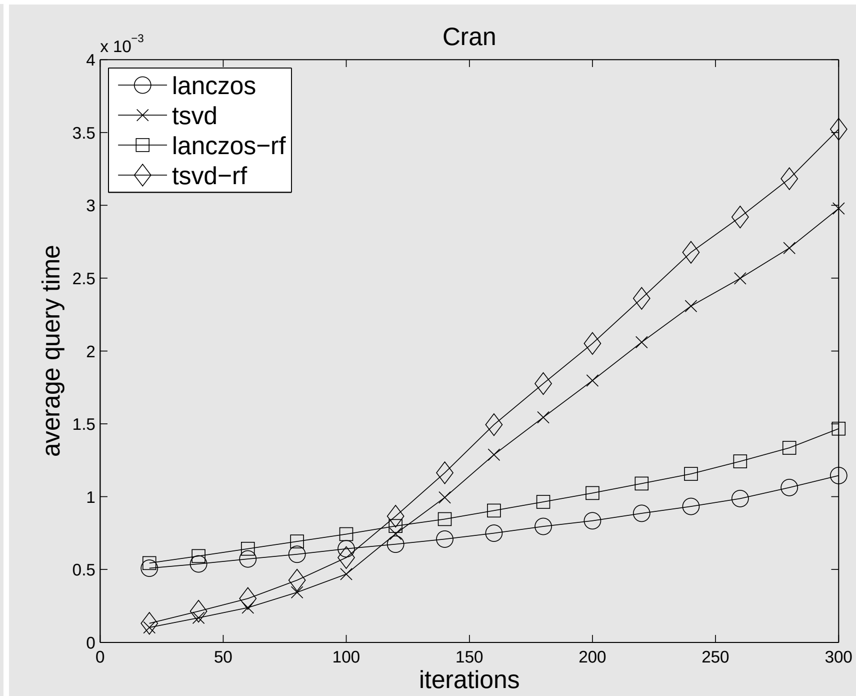
Cran dataset.



Average query times

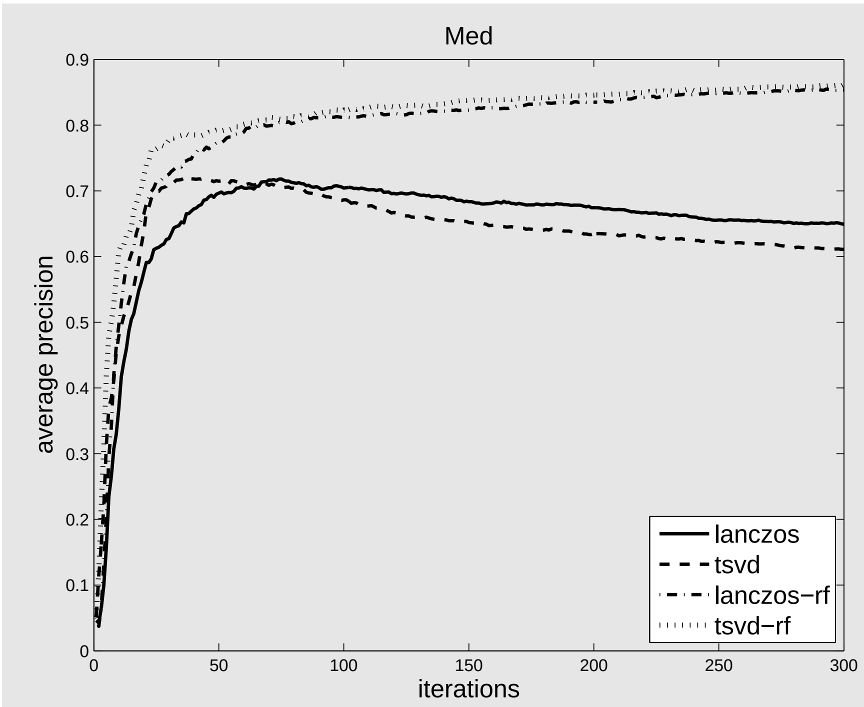
Med dataset **Cran dataset.**



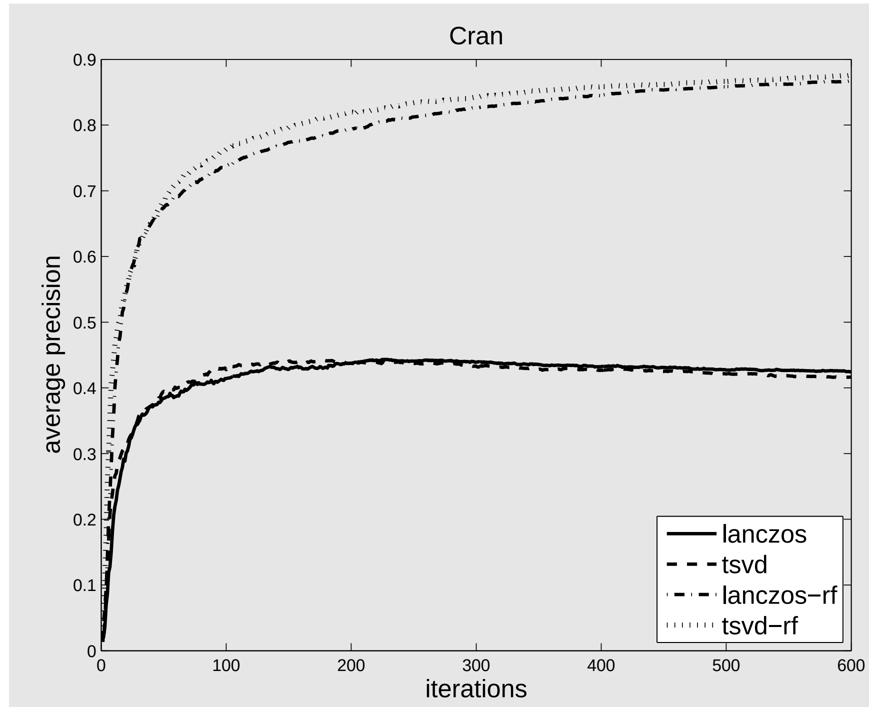


Average retrieval precision

Med dataset



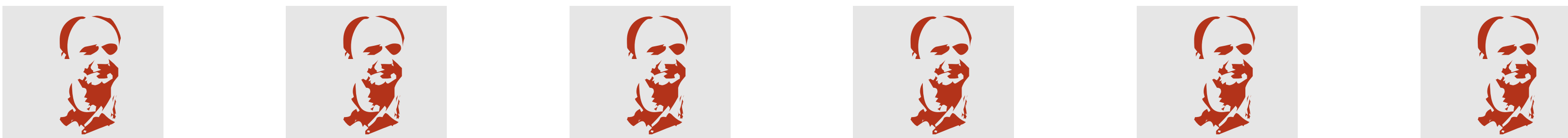
Cran dataset



Retrieval precision comparisons

Problem 2: 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 :

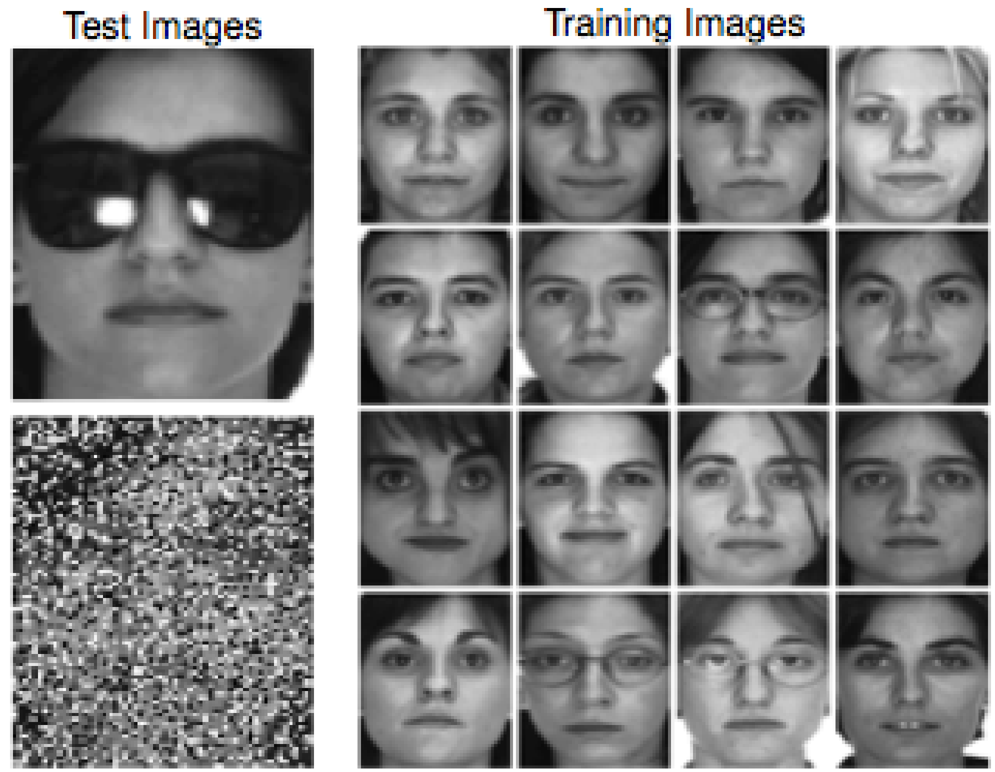
Common approach: eigenfaces – Principal Component Analysis technique

Example: Occlusion.

See recent paper by John Wright et al.

Top test image: deliberate disguise.

Bottom: 50% pixels randomly changed



Source: `http://perception.csl.uiuc.edu/ ...`

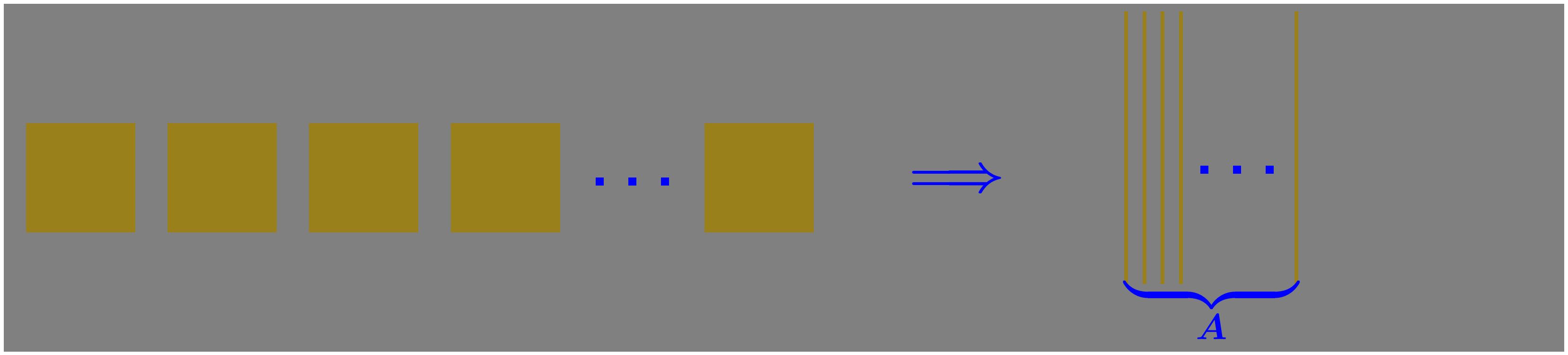
`... recognition/Robust_face.html`

- See also: Recent real-life example – international man-hunt

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

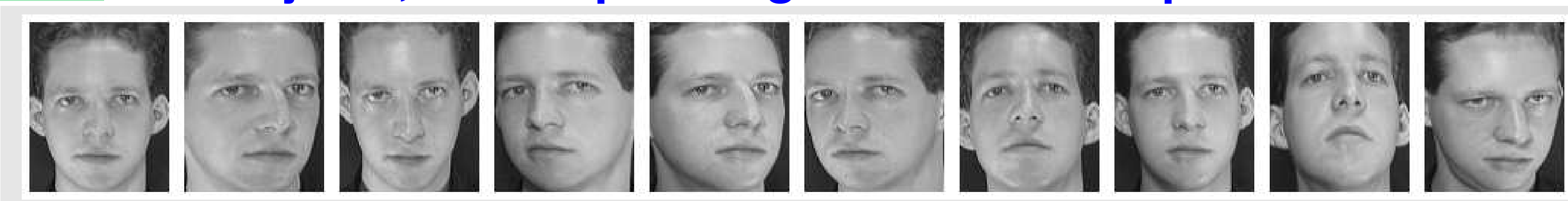
– Similar to LSI in spirit – but data is not sparse.

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

Tests: Face Recognition

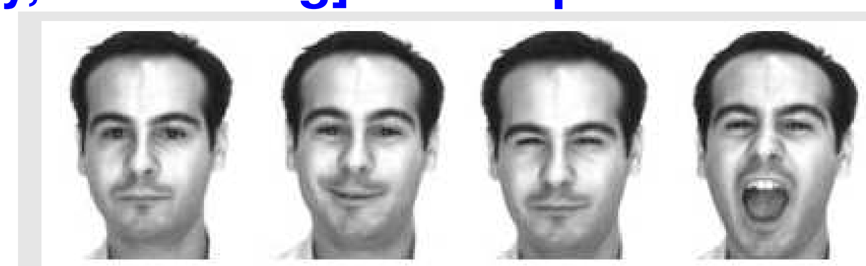
Tests with 2 well-known data sets:

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:

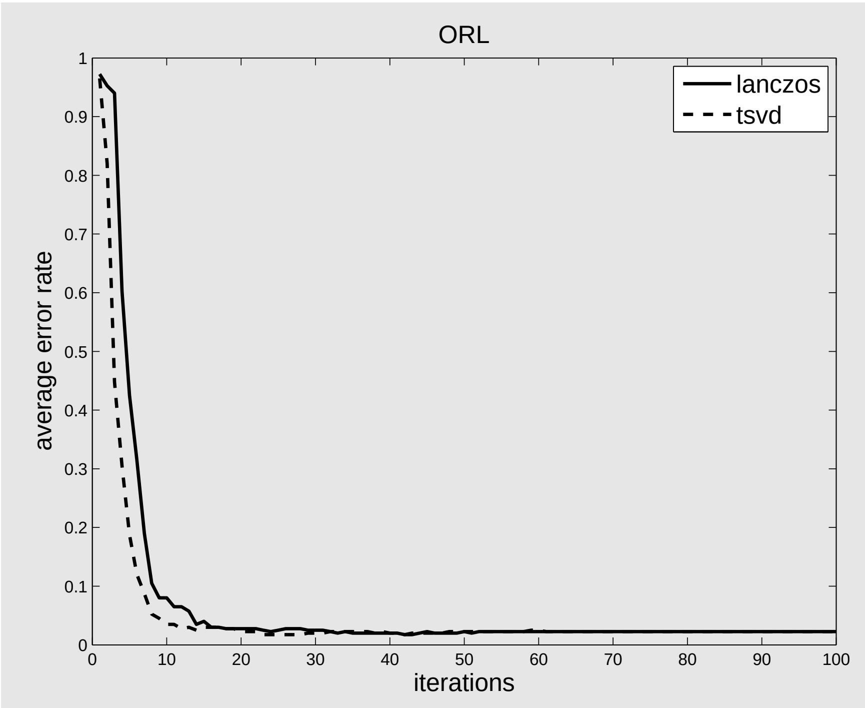


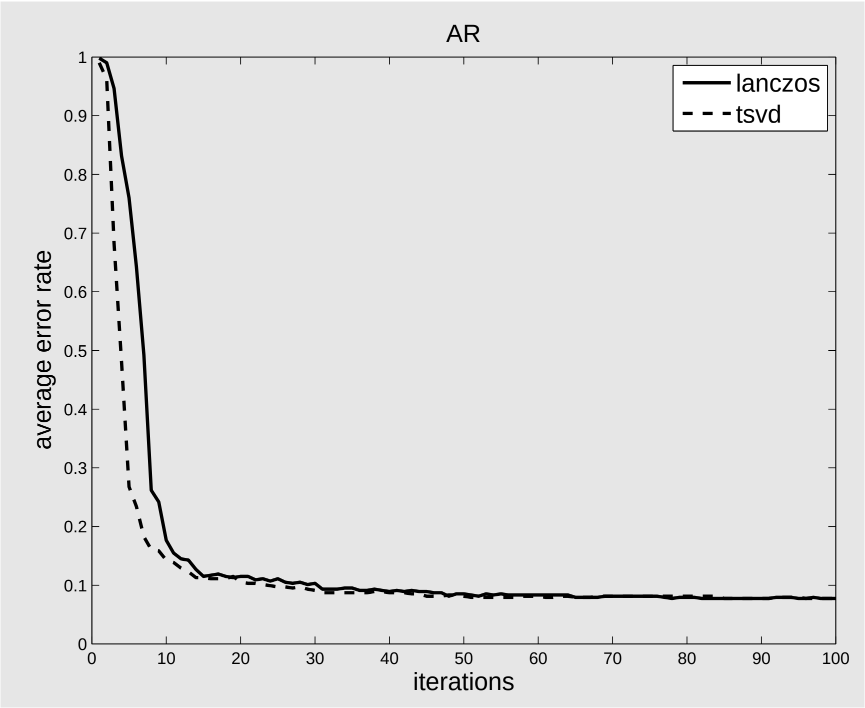
# of pixels : 112×92 **# TOT. # images : 504**

Tests: Face Recognition

Recognition accuracy of Lanczos approximation vs SVD

ORL dataset **AR dataset**





Vertical axis shows average error rate. Horizontal = Subspace dimension

Problem 3: Clustering

* Joint work with Haw-Ren Fang – in progress

Problem: A set X of n objects in some space. Find subsets of X that each contain objects that are most 'alike'

- 'Bread-and-butter problem' – arises in *many* applications
- Variation of the problem: Graph partitioning [need closeness + few edge cuts]
- Supervised clustering: Subsets are known – problem is to optimally 'classify' a new item into one of the subsets

Questions: 'alike' in what sense? How many subsets?

Clustering: using farthest centroids

- Given $X = [x_1\ x_2\ \cdots\ x_n] \in \mathbb{R}^{m\times n}$
- Centroid of a set $Y = [y_1, \cdots, y_p]$ is

$$c_Y = \frac{1}{p}\sum_{j=1}^{p} y_j = \frac{1}{p} Y e \quad e = [1, 1, \cdots, 1]^T$$

- Clustering into 2 even sets. Idea: find partition vector c:

$$\text{Maximize} \quad \|Xc\|_2$$
$$\text{subject to} \begin{cases} c_i & = \pm 1,\ i = 1, \cdots, n \\ c^T e & = 0 \end{cases}$$

- Subset X_+ = set with $c_i = 1$, Subset X_- = set with $c_i = -1$
- $c^T e = 0$ is a balance constraint between the 2 sets

- Hard problem to solve [integer programming – NP-hard]
- But: can be solved approximately [$\sim$ graph partitioning]
- Can also relax constraints.

❶ 'center' X, i.e., use $\bar{X} = X - \frac{1}{n} X e^T$ for X

❷ Replace $c_i = \pm 1$ by $c^T c = n$

$$\text{Maximize} \quad \|\bar{X} c\|_2$$
$$\text{subject to} \begin{cases} \|c\|_2 = 1, \\ c^T e = 0 \end{cases}$$

Solution = dominant singular vector.

- Exploited by Boley '97 in PDDP – [See also Juhász '81]
- Similar idea exploited in graph partitioning

Even-sets clustering by exchange

- Go back to constraint $c_i = \pm 1$ – i.e., use actual centroids
- Need to improve a given partition
- Similar to Kernigan and Lin in graph partitioning
- Let $Y = [y_1, \cdots, y_{n/2}]$. $Z = [z_1, \cdots, z_{n/2}]$
- Scaled squared distance between the centroids is

$$d = \|Ye - Ze\|_2^2 = (Ye - Ze)^T (Ye - Ze)$$

- What happens if we swap $y^* \in Y$ and $z^* \in Z$?

- Call $\delta = y^* - z^*$
- New distance:

$$\begin{aligned} d_{new} &= \|(Ye - y^* + z^*) - (Ze - z^* + y^*)\|_2^2 \\ &= \|(Ye - \delta) - (Ze + \delta)\|_2^2 \\ &= \|(Ye - Ze) - 2\delta\|_2^2 \\ &= d + 4\|\delta\|_2^2 - 4((Ye - Ze), \delta) \end{aligned}$$

- Distance gains if :

$$-(Ye - Ze)^T \delta + \|\delta\|_2^2 > 0$$

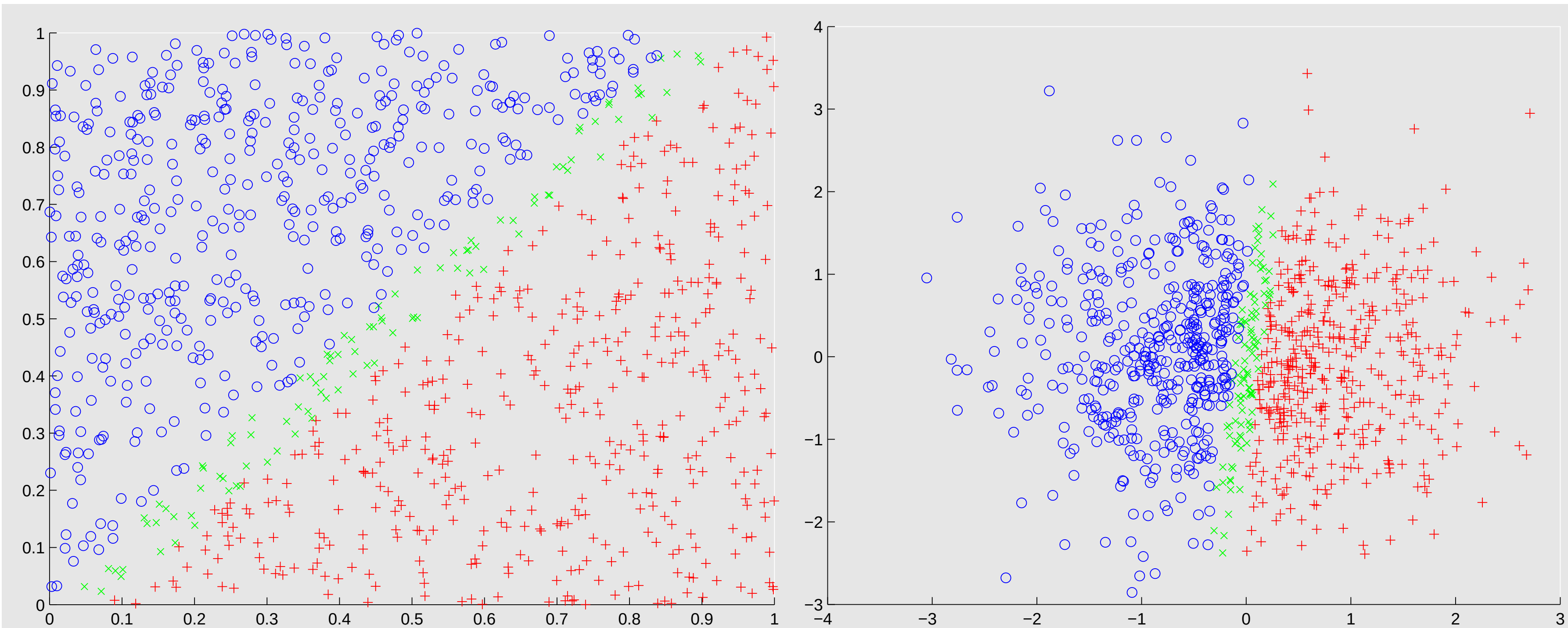
Idea:

- Begin with the Lanczos algorithm for $\bar{X}^T\bar{X}$ to get $s.\vec{v}.v_1$

- Get a marginal set among components of v_1 for refining

- Repeat: exchange marginal points (only) – until no further gains are made

Clustering: example



Initialization of two sets of $n = 1,000$ random points on two-dimensional plane. Green points are margin set (100). Left: uniform distribution; right: normal distribution.

Clustering : K-means + improvement

ALGORITHM : 1 ▪ K -means clustering algorithm

Given: K **initial centroids** $p_1, \cdots, p_K$

Do:

Set $S_j := \emptyset$ **for** $j = 1, \ldots, K$.

For $i = 1, 2 \ldots, n$

Find $k =$ **argmin**$_j \|x_i - p_j\|$

Set $S_k := S_k \cup \{x_i\}$.

EndFor

For $j = 1, 2, \ldots, K$

Set p_j **== mean of points in** S_j.

EndFor

While $\{ p_1, \ldots, p_K \}$ **have not converged.**

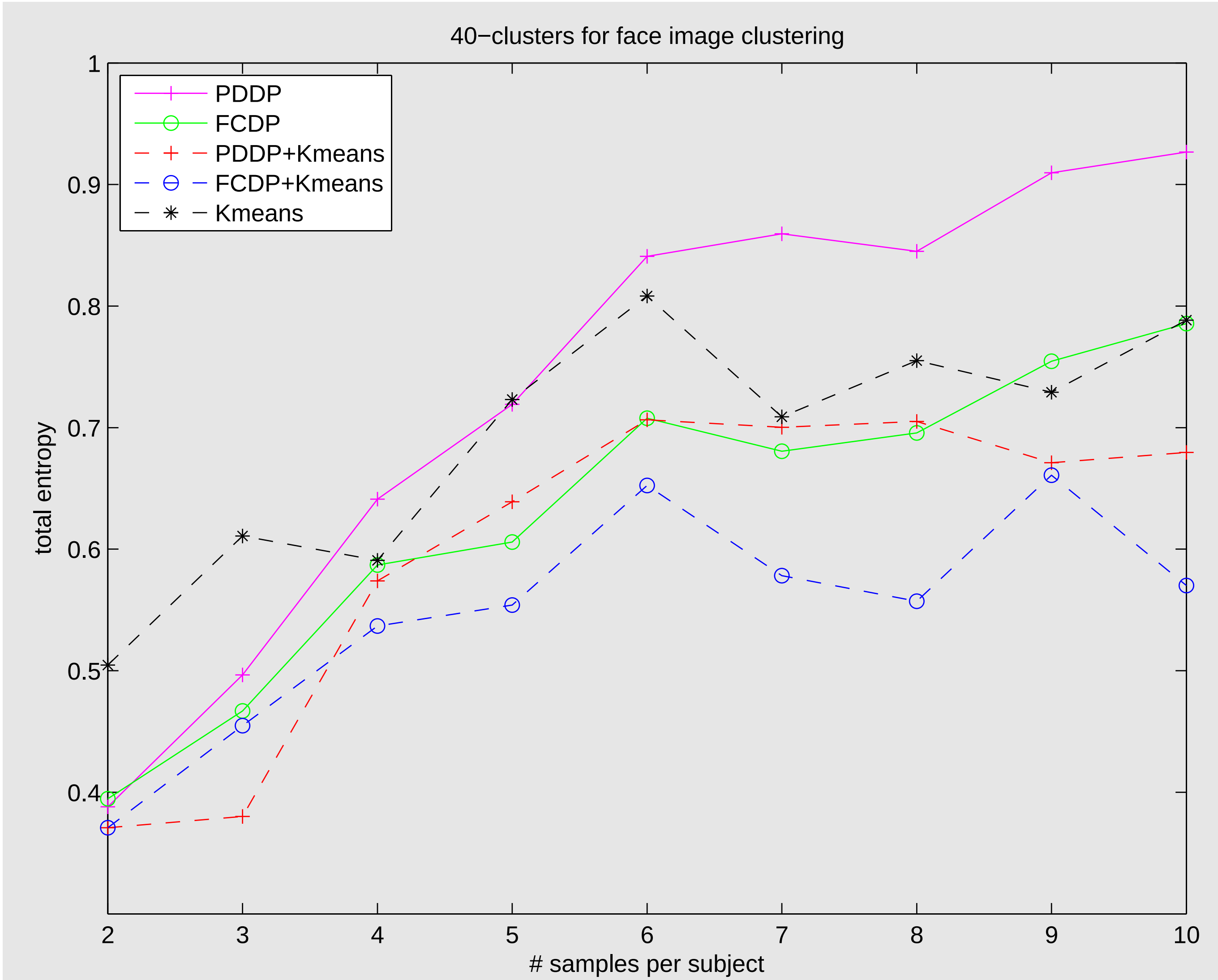
In words: Find closest centroid p_k to each x_i. Add this x_i to S_k. Get new centroids. Repeat.

- Excellent algorithm – but very slow. Depends on initial set.
- Common practice: start with something else – [cheaper]

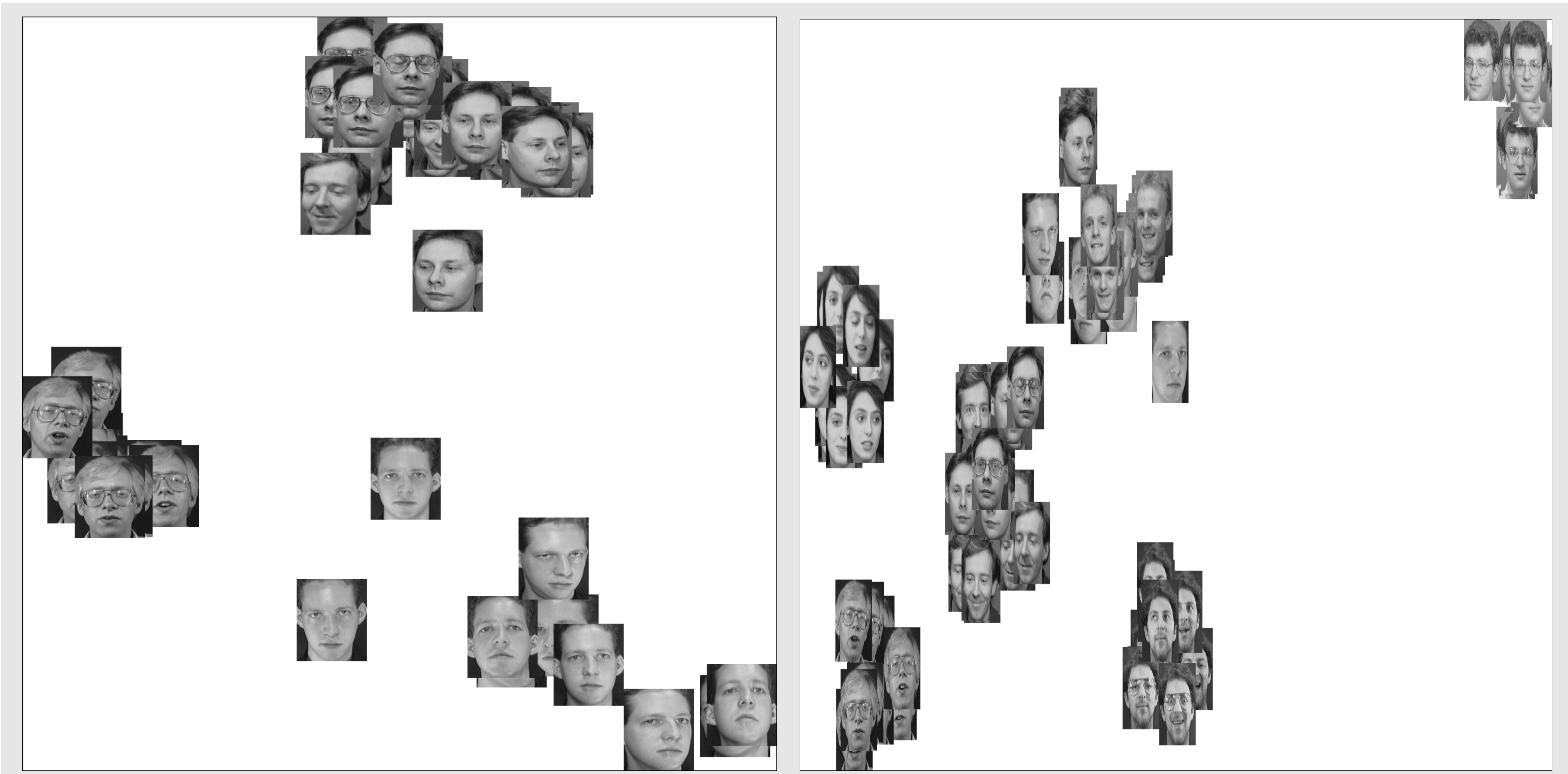
Ideas:

1. Start with PDDP [Lanczos] then refine with K-means
2. Start with FCDP [Lanczos] then refine with K-means

Clustering : test with ORL –get 40 clusters



➤ Result of clustering displayed on a 2-D plane:



Left: clustering by PCA. Right: clustering by FCDC.

Conclusion

- Many interesting linear algebra problems in data mining.

- Current methods mix 1) statistics, 2) Linear algebra 3) Differential geometry (manifold learning) 4) (Basic) graph theory

- Have shown some simple techniques put to work..

- Work on clustering still challenging..

- Modern dimension reduction techniques (LLE, Eigenmaps, Isomap, ...) exploit nearest neighbor graph. Resulting methods quite powerful