15-826: Multimedia Databases and Data Mining

SVD - part III (more case studies)

C. Faloutsos

Outline

Goal: 'Find similar / interesting things'
- Intro to DB
- Indexing - similarity search
- Data Mining

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
- fractals
- text
- Singular Value Decomposition (SVD)
- multimedia
- ...

SVD - Detailed outline

- Motivation
- Definition - properties
- Interpretation
- Complexity
- Case studies
- SVD properties
- More case studies
- Conclusions

SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks
- Conclusions

SVD - Other properties - summary

- can produce orthogonal basis (obvious) (who cares?)
- can solve over- and under-determined linear problems (see C(1) property)
- can compute 'fixed points' (= 'steady state prob. in Markov chains') (see C(4) property)

CMU SCS

SVD -outline of properties

- (A): obvious
- (B): less obvious
- (C): least obvious (and most powerful!)

CMU SCS

Properties - by defn.:

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

A(1): $\mathbf{U}^T_{[r \times n]} \mathbf{U}_{[n \times r]} = \mathbf{I}_{[r \times r]}$ (identity matrix)

A(2): $\mathbf{V}^T_{[r \times n]} \mathbf{V}_{[n \times r]} = \mathbf{I}_{[r \times r]}$

A(3): $\mathbf{\Lambda}^k = \text{diag}(\lambda_1^k, \lambda_2^k, \ldots \lambda_r^k)$ (k: ANY real number)

A(4): $\mathbf{A}^T = \mathbf{V} \mathbf{\Lambda} \mathbf{U}^T$

CMU SCS

Less obvious properties

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

B(1): $\mathbf{A}_{[n \times m]} (\mathbf{A}^T)_{[m \times n]} = ??$

CMU SCS

Less obvious properties

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

B(1): $\mathbf{A}_{[n \times m]} (\mathbf{A}^T)_{[m \times n]} = \mathbf{U} \mathbf{\Lambda}^2 \mathbf{U}^T$

symmetric; Intuition?

CMU SCS

Less obvious properties

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

B(1): $\mathbf{A}_{[n \times m]} (\mathbf{A}^T)_{[m \times n]} = \mathbf{U} \mathbf{\Lambda}^2 \mathbf{U}^T$

symmetric; Intuition?

'document-to-document' similarity matrix

B(2): symmetrically, for '**V**'

$(\mathbf{A}^T)_{[m \times n]} \mathbf{A}_{[n \times m]} = \mathbf{V} \mathbf{\Lambda}^2 \mathbf{V}^T$

Intuition?

CMU SCS

Less obvious properties

A: term-to-term similarity matrix

B(3): $((\mathbf{A}^T)_{[m \times n]} \mathbf{A}_{[n \times m]})^k = \mathbf{V} \mathbf{\Lambda}^{2k} \mathbf{V}^T$

and

B(4): $(\mathbf{A}^T \mathbf{A})^k \sim \mathbf{v}_1 \lambda_1^{2k} \mathbf{v}_1^T$ for k>>1

where

$\mathbf{v}_1$: [m x 1] first column (singular-vector) of **V**

λ_1: strongest singular value

CMU SCS

Less obvious properties

B(4): $(\mathbf{A}^T \mathbf{A})^k \sim \mathbf{v}_1 \lambda_1^{2k} \mathbf{v}_1^T$ for k>>1

$\boxed{\text{B(5): } (\mathbf{A}^T \mathbf{A})^k \mathbf{v'} \sim \text{(constant) } \mathbf{v}_1}$

ie., for (almost) any **v'**, it converges to a vector parallel to $\mathbf{v}_1$

Thus, useful to compute first singular vector/value (as well as the next ones, too...)

CMU SCS

Less obvious properties - repeated:

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

B(1): $\mathbf{A}_{[n \times m]} (\mathbf{A}^T)_{[m \times n]} = \mathbf{U} \mathbf{\Lambda}^2 \mathbf{U}^T$

B(2): $(\mathbf{A}^T)_{[m \times n]} \mathbf{A}_{[n \times m]} = \mathbf{V} \mathbf{\Lambda}^2 \mathbf{V}^T$

B(3): $((\mathbf{A}^T)_{[m \times n]} \mathbf{A}_{[n \times m]})^k = \mathbf{V} \mathbf{\Lambda}^{2k} \mathbf{V}^T$

B(4): $(\mathbf{A}^T \mathbf{A})^k \sim v_1 \lambda_1^{2k} v_1^T$

B(5): $(\mathbf{A}^T \mathbf{A})^k \mathbf{v'} \sim \text{(constant) } \mathbf{v}_1$

CMU SCS

Least obvious properties

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

C(1): $\mathbf{A}_{[n \times m]} \mathbf{x}_{[m \times 1]} = \mathbf{b}_{[n \times 1]}$

let $\mathbf{x}_0 = \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b}$

if under-specified, $\mathbf{x}_0$ gives 'shortest' solution

if over-specified, it gives the 'solution' with the smallest least squares error

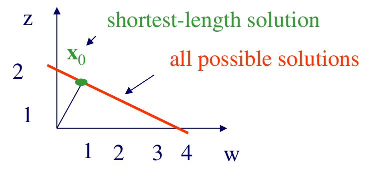
(see Num. Recipes, p. 62)

CMU SCS

Least obvious properties

Illustration: under-specified, eg

[1 2] [w z] ^{T} = 4 (ie, 1 w + 2 z = 4)

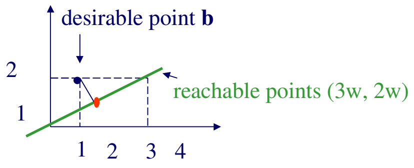


CMU SCS

Least obvious properties

Illustration: over-specified, eg

$[3\ 2]^T\ [w] = [1\ 2]^T$ (ie, 3 w = 1; 2 w = 2)



CMU SCS

Least obvious properties - cont'd

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

C(2): $\mathbf{A}_{[n \times m]} \mathbf{v}_{1\,[m \times 1]} = \lambda_1 \mathbf{u}_{1\,[n \times 1]}$

where $\mathbf{v}_1$, $\mathbf{u}_1$ the first (column) vectors of **V**, **U.** ($\mathbf{v}_1$ == right-singular-vector)

C(3): symmetrically: $\mathbf{u}_1^T \mathbf{A} = \lambda_1 \mathbf{v}_1^T$

$\mathbf{u}_1$ == left-singular-vector

Therefore:

CMU SCS

Least obvious properties - cont'd

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

C(4): $\boxed{\mathbf{A}^T \mathbf{A} \mathbf{v}_1 = \lambda_1^2 \mathbf{v}_1}$

(**fixed point** - the dfn of eigenvector for a symmetric matrix)

CMU SCS

Least obvious properties - altogether

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

C(1): $\mathbf{A}_{[n \times m]} \mathbf{x}_{[m \times 1]} = \mathbf{b}_{[n \times 1]}$
then, $\mathbf{x}_0 = \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b}$: shortest, actual or least-squares solution

C(2): $\mathbf{A}_{[n \times m]} \mathbf{v}_{1\,[m \times 1]} = \lambda_1 \mathbf{u}_{1\,[n \times 1]}$

C(3): $\mathbf{u}_1^T \mathbf{A} = \lambda_1 \mathbf{v}_1^T$

C(4): $\mathbf{A}^T \mathbf{A} \mathbf{v}_1 = \lambda_1^2 \mathbf{v}_1$

CMU SCS

Properties - conclusions

A(0): $\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} \mathbf{V}^T_{[r \times m]}$

B(5): $(\mathbf{A}^T \mathbf{A})^k \mathbf{v}' \sim$ (constant) $\mathbf{v}_1$

C(1): $\mathbf{A}_{[n \times m]} \mathbf{x}_{[m \times 1]} = \mathbf{b}_{[n \times 1]}$
then, $\mathbf{x}_0 = \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b}$: shortest, actual or least-squares solution

C(4): $\mathbf{A}^T \mathbf{A} \mathbf{v}_1 = \lambda_1^2 \mathbf{v}_1$

CMU SCS

SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - Kleinberg/google algorithms
  - query feedbacks
- Conclusions

CMU SCS

Kleinberg's algorithm

- Problem dfn: given the web and a query
- find the most 'authoritative' web pages for this query

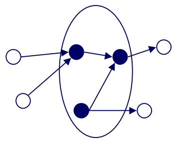
Step 0: find all pages containing the query terms

Step 1: expand by one move forward and backward

CMU SCS

Kleinberg's algorithm

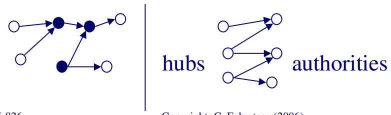
- Step 1: expand by one move forward and backward



CMU SCS

Kleinberg's algorithm

- on the resulting graph, give high score (= 'authorities') to nodes that many important nodes point to
- give high importance score ('hubs') to nodes that point to good 'authorities')



CMU SCS

Kleinberg's algorithm

observations

- recursive definition!
- each node (say, 'i'-th node) has both an authoritativeness score a_i and a hubness score h_i

CMU SCS

Kleinberg's algorithm

Let E be the set of edges and **A** be the adjacency matrix:

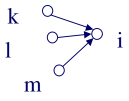
the (i,j) is 1 if the edge from i to j exists

Let h and a be [n x 1] vectors with the 'hubness' and 'authoritativiness' scores.

Then:

CMU SCS

Kleinberg's algorithm



Then:

$$a_i = h_k + h_l + h_m$$

that is

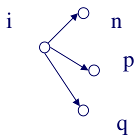
a_i = Sum (h_j) over all j that (j,i) edge exists

or

$$\mathbf{a} = \mathbf{A}^T \mathbf{h}$$

CMU SCS

Kleinberg's algorithm



symmetrically, for the 'hubness':

$$h_i = a_n + a_p + a_q$$

that is

h_i = Sum (q_j) over all j that (i,j) edge exists

or

$$\mathbf{h} = \mathbf{A}\,\mathbf{a}$$

CMU SCS

Kleinberg's algorithm

In conclusion, we want vectors **h** and **a** such that:

$$\mathbf{h} = \mathbf{A}\,\mathbf{a}$$

$$\mathbf{a} = \mathbf{A}^T \mathbf{h}$$

Recall properties:

C(2): $\mathbf{A}_{[n \times m]} \mathbf{v_1}_{[m \times 1]} = \lambda_1 \mathbf{u_1}_{[n \times 1]}$

C(3): $\mathbf{u_1}^T \mathbf{A} = \lambda_1 \mathbf{v_1}^T$

CMU SCS

Kleinberg's algorithm

In short, the solutions to

$$\mathbf{h} = \mathbf{A}\,\mathbf{a}$$
$$\mathbf{a} = \mathbf{A}^T\,\mathbf{h}$$

are the left- and right- singular-vectors of the adjacency matrix **A.**

Starting from random **a'** and iterating, we'll eventually converge

(Q: to which of all the singular-vectors? why?)

CMU SCS

Kleinberg's algorithm

(Q: to which of all the singular-vectors? why?)

A: to the ones of the strongest singular-value, because of property B(5):

B(5): $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v'} \sim$ (constant) $\mathbf{v}_1$

CMU SCS

Kleinberg's algorithm - results

Eg., for the query 'java':

0.328 www.gamelan.com

0.251 java.sun.com

0.190 www.digitalfocus.com ("the java developer")

CMU SCS

Kleinberg's algorithm - discussion

- 'authority' score can be used to find 'similar pages' (how?)
- closely related to 'citation analysis', social networs / 'small world' phenomena

CMU SCS

google/page-rank algorithm

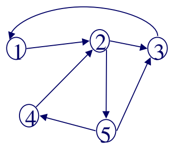
- closely related: imagine a particle randomly moving along the edges (*)
- compute its steady-state probabilities

(*) with occasional random jumps

CMU SCS

google/page-rank algorithm

- ~identical problem: given a Markov Chain, compute the steady state probabilities p1 ... p5

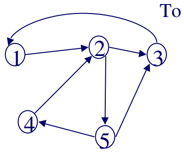


CMU SCS

(Simplified) PageRank algorithm

- Let **A** be the transition matrix (= adjacency matrix); let $\mathbf{A^T}$ become column-normalized - then

From / To



$\mathbf{A^T}$

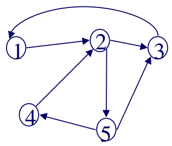
		1		
1			1	
	1/2			1/2
				1/2
	1/2			

× [p1, p2, p3, p4, p5] = [p1, p2, p3, p4, p5]

CMU SCS

(Simplified) PageRank algorithm

- $\mathbf{A^T\, p = p}$



$\mathbf{A^T}$

		1		
1			1	
	1/2			1/2
				1/2
	1/2			

p = **p**

[p1, p2, p3, p4, p5] = [p1, p2, p3, p4, p5]

CMU SCS

(Simplified) PageRank algorithm

- $\mathbf{A^T\, p} = 1 * \mathbf{p}$
- thus, **p** is the **eigenvector** that corresponds to the highest **eigenvalue** (=1, since the matrix is column-normalized)
- formal definition of eigenvector/value: soon

CMU SCS

(Simplified) PageRank algorithm

- In short: imagine a particle randomly moving along the edges
- compute its steady-state probabilities (ssp)

Full version of algo: with occasional random jumps

CMU SCS

Formal definition

If **A** is a (n x n) square matrix
(λ , **x**) is an **eigenvalue/eigenvector** pair of **A** if

$$\boxed{\mathbf{A\, x} = \lambda\, \mathbf{x}}$$

CLOSELY related to singular values:

CMU SCS

Eigen- vs singular-values

if

$$\mathbf{B}_{[n\times m]} = \mathbf{U}_{[n\times r]}\, \mathbf{\Lambda}_{[r\times r]}\, (\mathbf{V}_{[m\times r]})^T$$

then $\mathbf{A} = (\mathbf{B^T B})$ is symmetric and

$$C(4):\ \mathbf{B^T\, B\, v_i} = \lambda_i^2\, \mathbf{v_i}$$

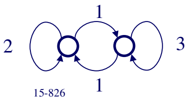
ie, $\mathbf{v_1}$, $\mathbf{v_2}$, ...: eigenvectors of $\mathbf{A} = (\mathbf{B^T B})$

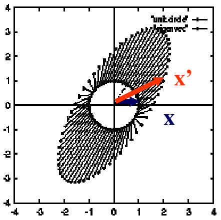
CMU SCS

Intuition

- **A** as vector transformation

$$\underset{x'}{\begin{bmatrix}2\\1\end{bmatrix}} = \underset{A}{\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}} \underset{x}{\begin{bmatrix}1\\0\end{bmatrix}}$$



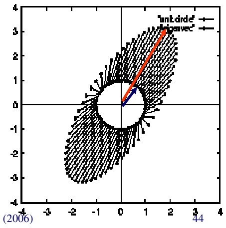


CMU SCS

Intuition

- By defn., eigenvectors remain parallel to themselves ('**fixed points**')

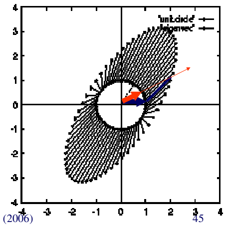
$$\underset{\lambda_1}{3.62} * \underset{v_1}{\begin{bmatrix}0.52\\0.85\end{bmatrix}} = \underset{A}{\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}} \underset{v_1}{\begin{bmatrix}0.52\\0.85\end{bmatrix}}$$



CMU SCS

Convergence

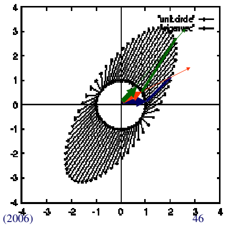
- Usually, fast:



CMU SCS

Convergence

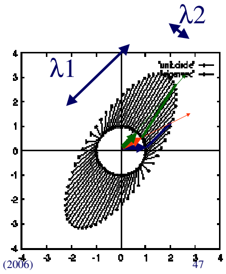
- Usually, fast:



CMU SCS

Convergence

- Usually, fast:
- depends on ratio $\lambda 1 : \lambda 2$



CMU SCS

Kleinberg/google - conclusions

SVD helps in graph analysis:

hub/authority scores: strongest left- and right-singular-vectors of the adjacency matrix

random walk on a graph: steady state probabilities are given by the strongest eigenvector of the transition matrix

CMU SCS

SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks
- Conclusions

CMU SCS

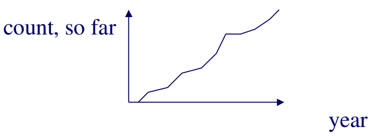
Query feedbacks

[Chen & Roussopoulos, sigmod 94]
sample problem:
estimate selectivities (e.g., '*how many movies were made between 1940 and 1945*?'
for query optimization,
LEARNING from the query results so far!!

CMU SCS

Query feedbacks

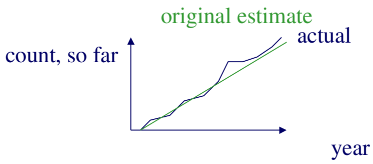
Idea #1: consider a function for the CDF (cummulative distr. function), eg., 6-th degree polynomial (or splines, or whatever)



CMU SCS

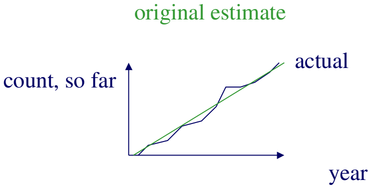
Query feedbacks

GREAT Idea #2: adapt your model, as you see the actual counts of the actual queries



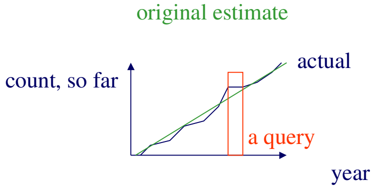
CMU SCS

Query feedbacks

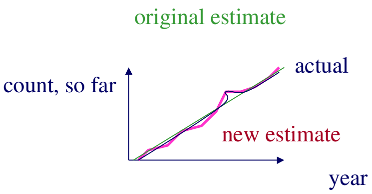


CMU SCS

Query feedbacks



Query feedbacks



Query feedbacks

Eventually, the problem becomes:
- estimate the parameters a1, ... a7 of the model
- to minimize the least squares errors from the real answers so far.

Formally:

Query feedbacks

Formally, with n queries and 6-th degree polynomials:

$$\begin{bmatrix} X11 & X12 & & & X17 \\ & & & & \\ & & & & \\ Xn1 & Xn2 & & & Xn7 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ \\ \\ a7 \end{bmatrix} = \begin{bmatrix} b1 \\ b2 \\ \\ \\ bn \end{bmatrix}$$

Query feedbacks

where $x_{i,j}$ such that Sum ($x_{i,j}$ * a_i) = our estimate for the # of movies and b_j: the actual

$$\begin{bmatrix} X11 & X12 & & & X17 \\ & & & & \\ & & & & \\ Xn1 & Xn2 & & & Xn7 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ \\ \\ a7 \end{bmatrix} = \begin{bmatrix} b1 \\ b2 \\ \\ \\ bn \end{bmatrix}$$

Query feedbacks

In matrix form:

$$\begin{array}{ccccc} & \mathbf{X} & \mathbf{a} & = & \mathbf{b} \\ \begin{array}{r} \text{1st query} \\ \\ \\ \text{n-th query} \end{array} & \begin{bmatrix} X11 & X12 & & & X17 \\ & & & & \\ & & & & \\ Xn1 & Xn2 & & & Xn7 \end{bmatrix} & \begin{bmatrix} a1 \\ a2 \\ \\ \\ a7 \end{bmatrix} & = & \begin{bmatrix} b1 \\ b2 \\ \\ \\ bn \end{bmatrix} \end{array}$$

Query feedbacks

In matrix form:

$$\mathbf{X}\,\mathbf{a} = \mathbf{b}$$

and the least-squares estimate for **a** is

$$\mathbf{a} = \mathbf{V}\,\mathbf{\Lambda}^{(-1)}\,\mathbf{U}^{\mathbf{T}}\,\mathbf{b}$$

according to property C(1)

(let $\mathbf{X} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{V}^{\mathbf{T}}$)

CMU SCS

Query feedbacks - enhancements

the solution

$$\mathbf{a} = \mathbf{V}\, \mathbf{\Lambda}^{(-1)}\, \mathbf{U}^{\mathbf{T}}\, \mathbf{b}$$

works, but needs expensive SVD each time a new query arrives

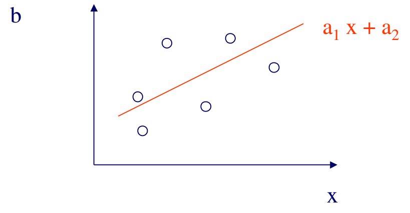
GREAT Idea #3: Use ‘Recursive Least Squares’, to adapt **a** incrementally.

Details: in paper - intuition:

CMU SCS

Query feedbacks - enhancements

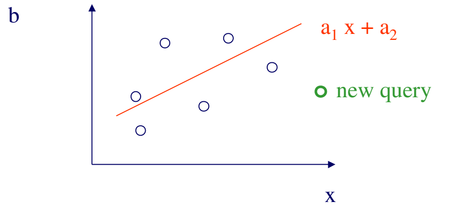
Intuition: least squares fit



CMU SCS

Query feedbacks - enhancements

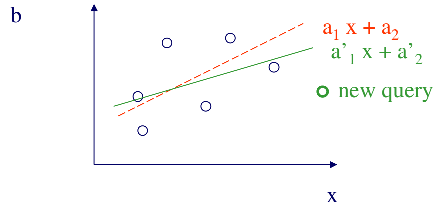
Intuition: least squares fit



CMU SCS

Query feedbacks - enhancements

Intuition: least squares fit



CMU SCS

Query feedbacks - enhancements

the new coefficients can be quickly computed from the old ones, plus statistics in a (7x7) matrix

(no need to know the details, although the RLS is a brilliant method)

CMU SCS

Query feedbacks - enhancements

GREAT idea #4: ‘forgetting’ factor - we can even down-play the weight of older queries, since the data distribution might have changed.

(comes for ‘free’ with RLS...)

CMU SCS

Query feedbacks - conclusions

SVD helps find the Least Squares solution, to adapt to query feedbacks

(RLS = Recursive Least Squares is a great method to incrementally update least-squares fits)

CMU SCS

SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks
- Conclusions

CMU SCS

Conclusions

- SVD: a **valuable** tool
- given a document-term matrix, it finds 'concepts' (LSI)
- ... and can reduce dimensionality (KL)
- ... and can find rules (PCA; RatioRules)

CMU SCS

Conclusions cont'd

- ... and can find fixed-points or steady-state probabilities (google/ Kleinberg/ Markov Chains)
- ... and can solve optimally over- and under-constraint linear systems (least squares / query feedbacks)

CMU SCS

References

- Brin, S. and L. Page (1998). Anatomy of a Large-Scale Hypertextual Web Search Engine. 7th Intl World Wide Web Conf.
- Chen, C. M. and N. Roussopoulos (May 1994). Adaptive Selectivity Estimation Using Query Feedback. Proc. of the ACM-SIGMOD , Minneapolis, MN.

CMU SCS

References cont'd

- Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. Proc. 9th ACM-SIAM Symposium on Discrete Algorithms.
- Press, W. H., S. A. Teukolsky, et al. (1992). Numerical Recipes in C, Cambridge University Press.