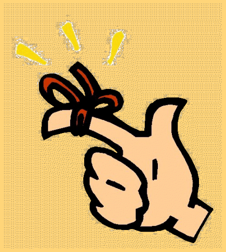
15-826: Multimedia (Databases) and Data Mining

Lecture #18: SVD - part III (more case studies)

C. Faloutsos

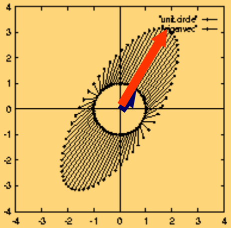
Problem

- Q1: most important node(s) in a graph?
  - A1.1:
  - A1.2:
- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
  - A2:

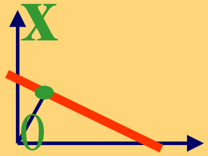


Conclusions

- Q1: most important node(s) in a graph?
  - A1.1: HITS (= SVD)
  - A1.2: PageRank (= fixed point)



- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
  - A2: SVD (< - > Moore-Penrose pseudo-inverse)



Must-read Material

- MM Textbook Appendix D
- Graph Mining Textbook, chapter 15.
- Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. Proc. 9th ACM-SIAM Symposium on Discrete Algorithms.
- Brin, S. and L. Page (1998). Anatomy of a Large-Scale Hypertextual Web Search Engine. 7th Intl World Wide Web Conf.

Must-read Material, cont'd

- Haveliwala, Taher H. (2003) Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. Extended version of the WWW2002 paper.
- Chen, C. M. and N. Roussopoulos (May 1994). Adaptive Selectivity Estimation Using Query Feedback. Proc. of the ACM-SIGMOD , Minneapolis, MN.

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

Parenthesis: intuition behind eigenvectors

- Definition
- 2 properties
- intuition

Formal definition

If **A** is a (n x n) square matrix

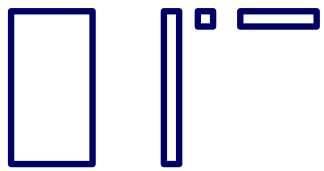
(λ , **x**) is an **eigenvalue/eigenvector** pair of **A** if

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

CLOSELY related to singular values:

Property #1: Eigen- vs singular-values

if



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

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

$$\mathbf{B}^{\mathbf{T}} \, \mathbf{B} \, \mathbf{v}_{\mathbf{i}} = \boldsymbol{\lambda}_{\mathbf{i}}^{\mathbf{2}} \, \mathbf{v}_{\mathbf{i}}$$

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

Property #2

- If $\mathbf{A}_{[nxn]}$ is a real, symmetric matrix
- Then it has *n* real eigenvalues

(if **A** is not symmetric, some eigenvalues may be complex)

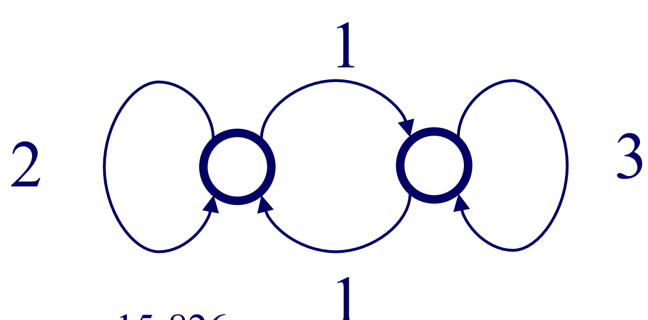
Parenthesis: intuition behind eigenvectors

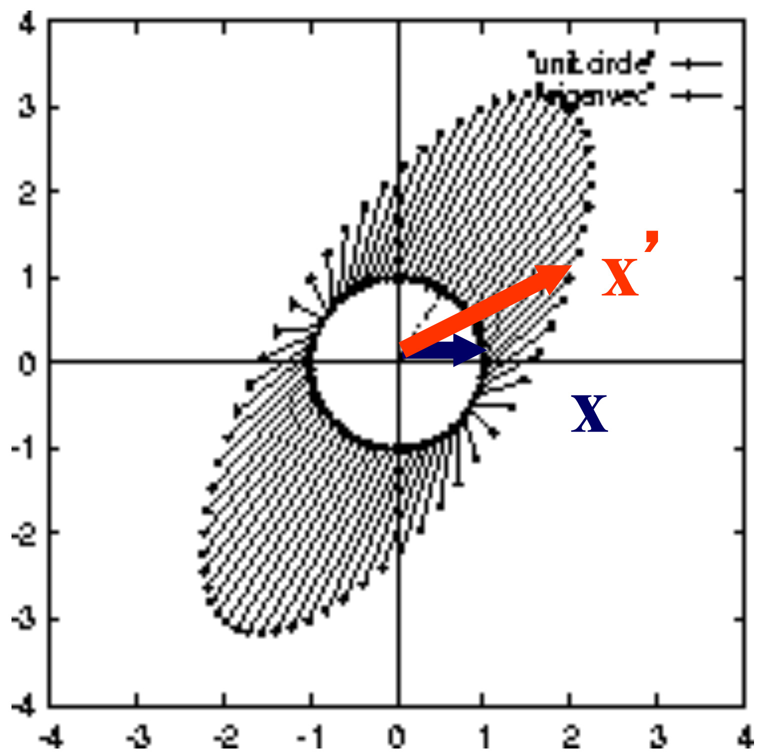
- Definition
- 2 properties
- **intuition**

Intuition

- **A** as vector transformation

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

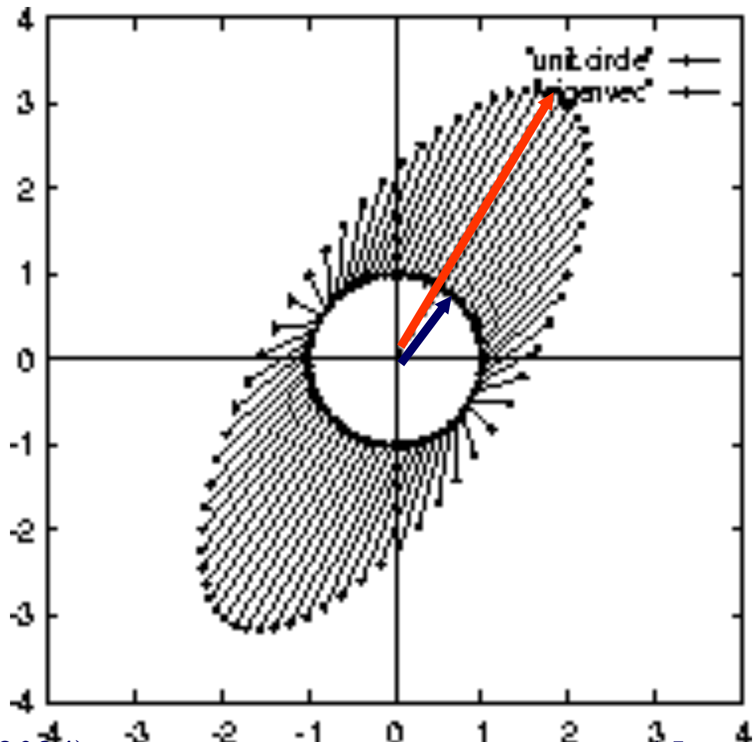




Intuition

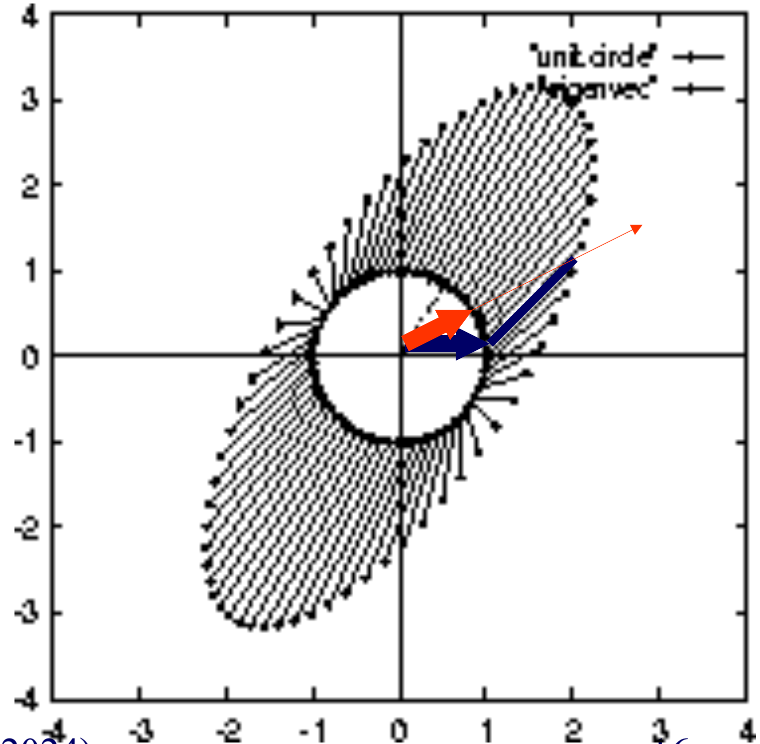
- By defn., eigenvectors remain parallel to themselves (‘**fixed points**’)

$$\overset{\lambda_1}{3.62} * \overset{\mathbf{v}_1}{\begin{bmatrix}0.52\\0.85\end{bmatrix}} = \overset{\mathbf{A}}{\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}} \overset{\mathbf{v}_1}{\begin{bmatrix}0.52\\0.85\end{bmatrix}}$$



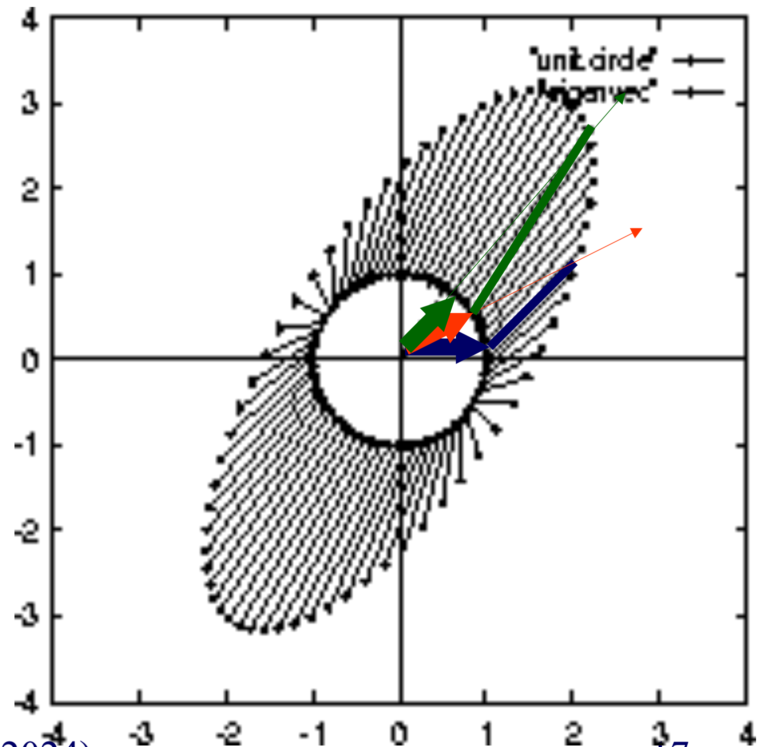
Convergence

- Usually, fast:



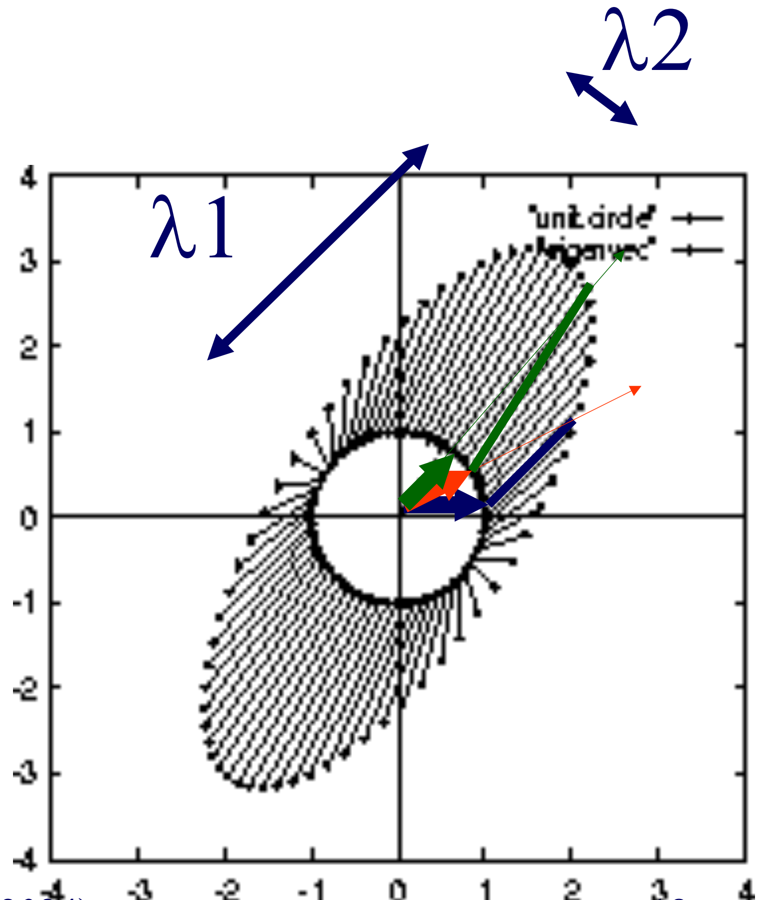
Convergence

- Usually, fast:



Convergence

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



Carnegie Mellon

What happens if $\lambda_1 = \lambda_2$?

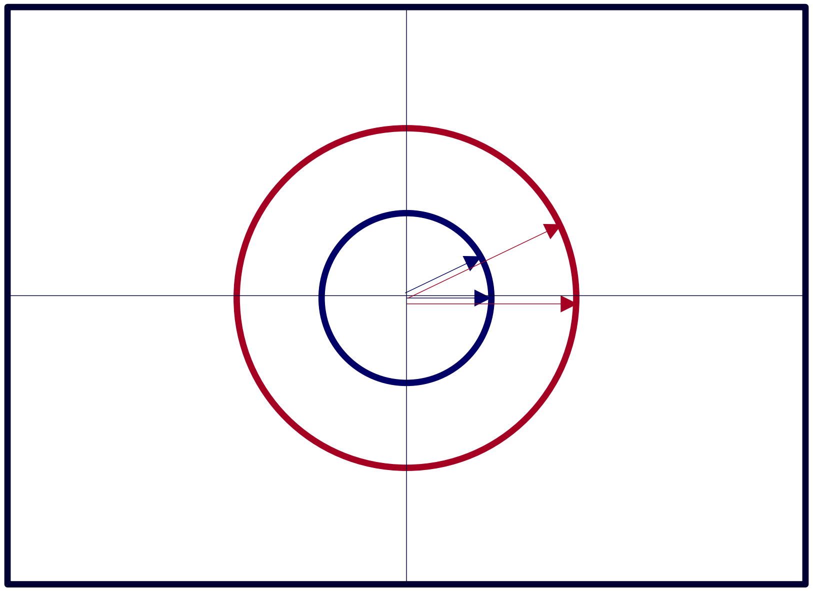
Say, $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

Carnegie Mellon

What happens if $\lambda_1 = \lambda_2$?

Say, $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

- No convergence
- NO unique eigenvector

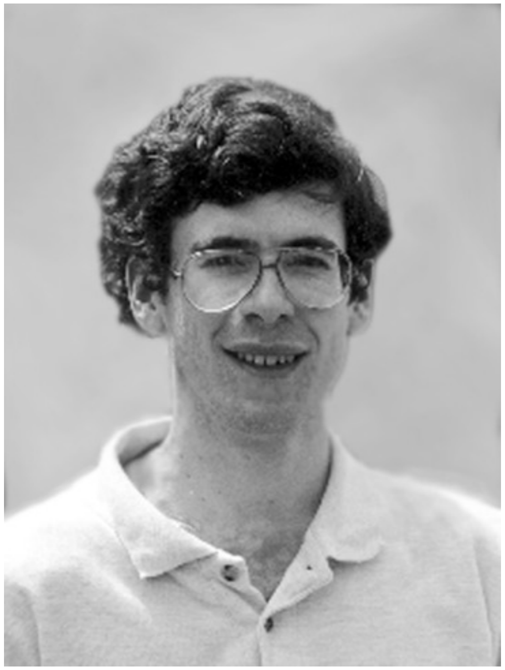


Closing the parenthesis wrt intuition behind eigenvectors

SVD - detailed outline

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

Kleinberg's algo (HITS)



Kleinberg, Jon (1998). *Authoritative sources in a hyperlinked environment*. Proc. 9th ACM-SIAM Symposium on Discrete Algorithms.

Kleinberg's algorithm

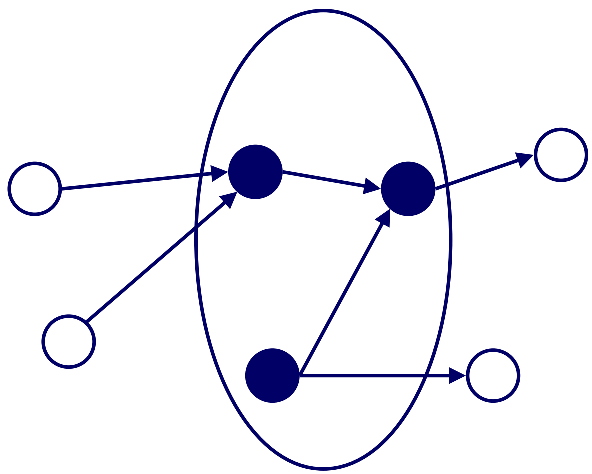
- Problem dfn: given the web and a query
- find the most 'authoritative' web pages for this query

Step 0: find all pages containing the query terms

Step 1: expand by one move forward and backward

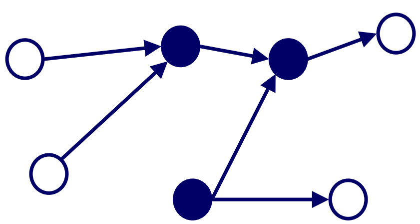
Kleinberg's algorithm

- Step 1: expand by one move forward and backward

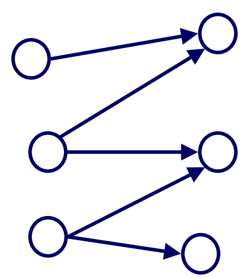


Kleinberg's algorithm

- 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')



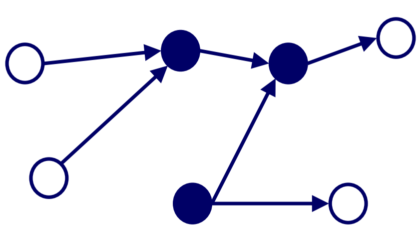
hubs

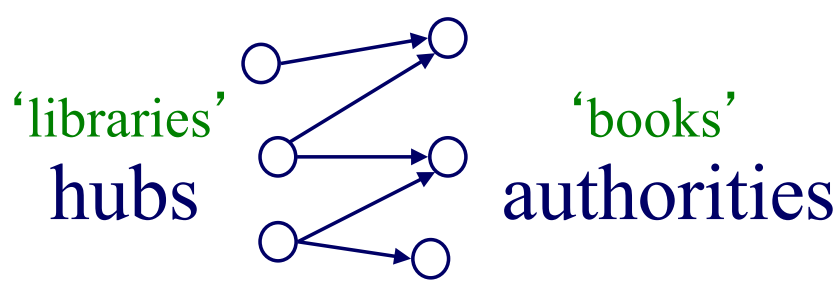


authorities

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')





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

Kleinberg's algorithm

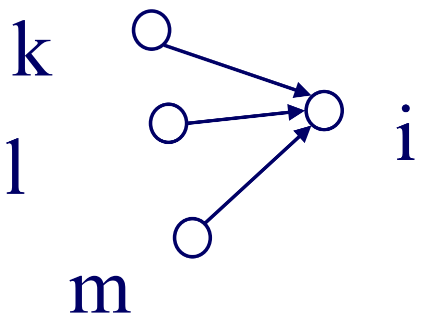
Let E be the set of edges and **A** be the adjacency matrix:

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:

Kleinberg's algorithm



Then:

$$a_i = h_k + h_l + h_m$$

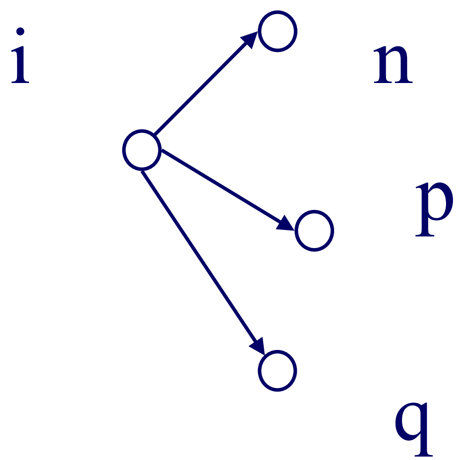
that is

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

or

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

Kleinberg's algorithm



symmetrically, for the 'hubness' :

$$h_i = a_n + a_p + a_q$$

that is

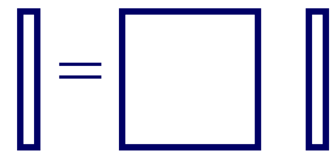
h_i = Sum (q_j) over all j that (i,j) edge exists

or

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

Kleinberg's algorithm

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



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

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

Properties:

- $\mathbf{A}_{[n \times m]}\ \mathbf{v_1}_{[m \times 1]} = \lambda_1\ \mathbf{u_1}_{[n \times 1]}$
- $\mathbf{u_1}^{\mathbf{T}}\ \mathbf{A} = \lambda_1\ \mathbf{v_1}^{\mathbf{T}}$

Kleinberg's algorithm

In short, the solutions to

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

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

are the <u>left- and right- singular-vectors</u> of the adjacency matrix **A.**

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

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

Kleinberg's algorithm

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

A: to the ones of the strongest singular-value, because of property:

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

Proof, and intuition

Carnegie Mellon

OPTIONAL

Proof (sketch)

$$A = U \Lambda V^T \qquad U^T U = I \qquad V^T V = I$$

1) $A^T A = V \Lambda \underbrace{U^T U}_{} \Lambda V^T = V \Lambda^2 V^T$

OPTIONAL

Proof (sketch)

$$A = U \Lambda V^T \qquad U^T U = I \qquad V^T V = I$$

1) $A^T A = V \Lambda \overbrace{U^T U} \Lambda V^T = V \Lambda^2 V^T$

2) $(A^T A)^k = V \Lambda^{2k} V^T$

Spectral form

$$\lambda_1^{2k} v_1 v_1^T + \lambda_2^{2k} v_2 v_2^T + \ldots$$

$$\approx \lambda_1^{2k} v_1 v_1^T$$

$\lambda_1 > \lambda_2 > \ldots$

OPTIONAL

Proof (sketch)

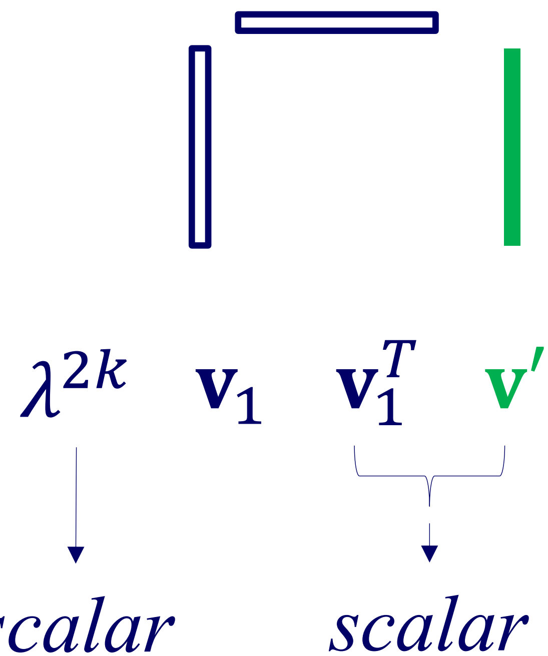
$$A = U \Lambda V^T \quad U^T U = I \quad V^T V = I$$

3) $(A^T A)^k v' \approx \lambda_1^{2k} v_1 v_1^T v'$

OPTIONAL

Proof (sketch) - pictorial

$$(A^T A)^k v' \approx \lambda_1^{2k} v_1 v_1^T v'$$

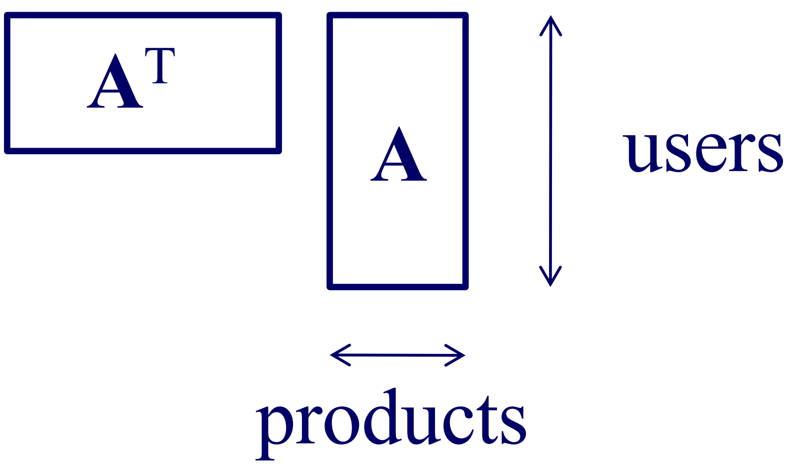


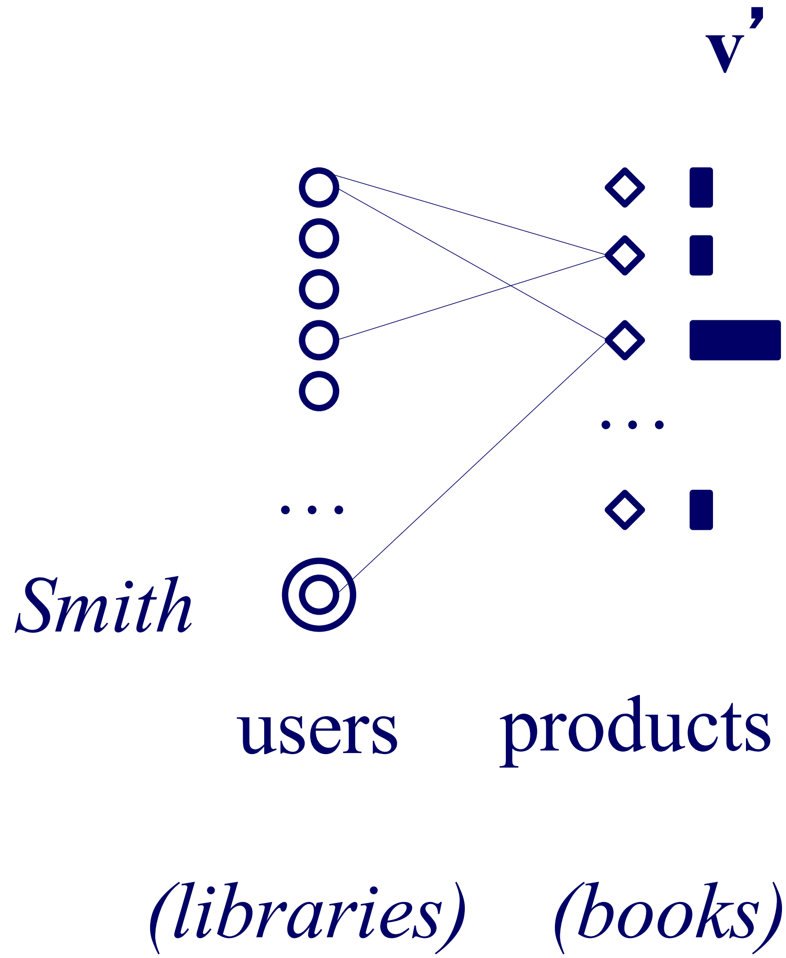
More intuition

- $(\mathbf{A}^T \mathbf{A})^k \mathbf{v}' \sim (\text{constant})\, \mathbf{v}_1$
- $== \underbrace{(\mathbf{A}^T \mathbf{A}) \ldots (\mathbf{A}^T \mathbf{A})}_{k \text{ times}} \mathbf{v}' \sim (\text{constant})\, \mathbf{v}_1$

More intuition

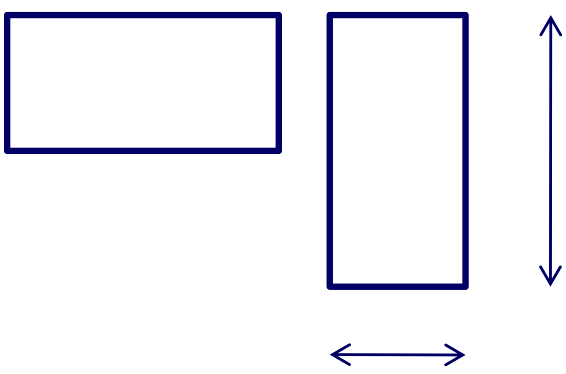
- Intuition:
  - $(\mathbf{A}^T \mathbf{A})\ \mathbf{v}'$
  - $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v}'$

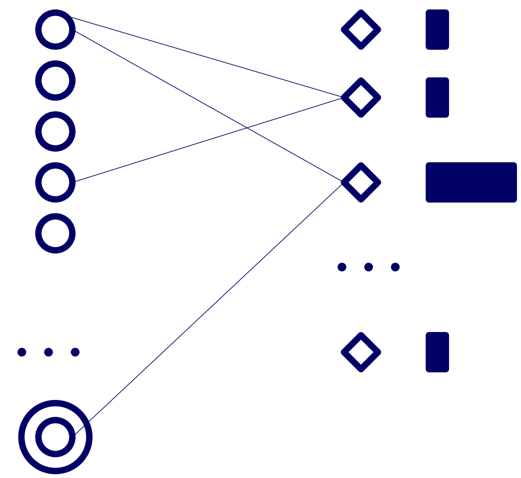




More intuition

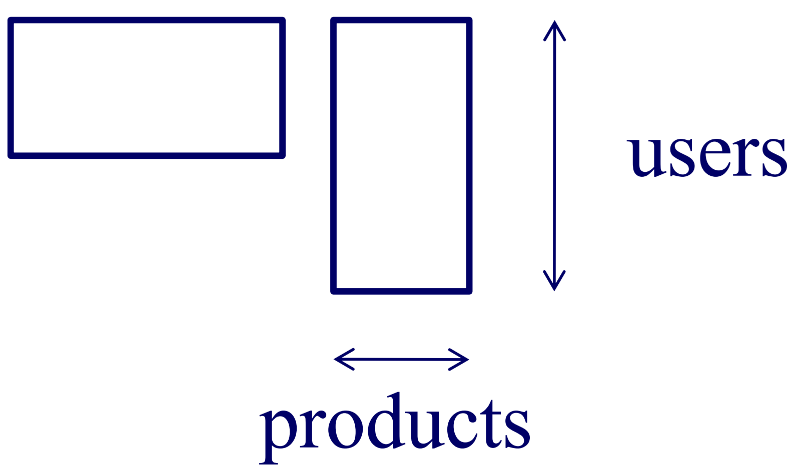
- Intuition:
  - $(\mathbf{A}^T \mathbf{A})\ \mathbf{v'}$
  - $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v'}$

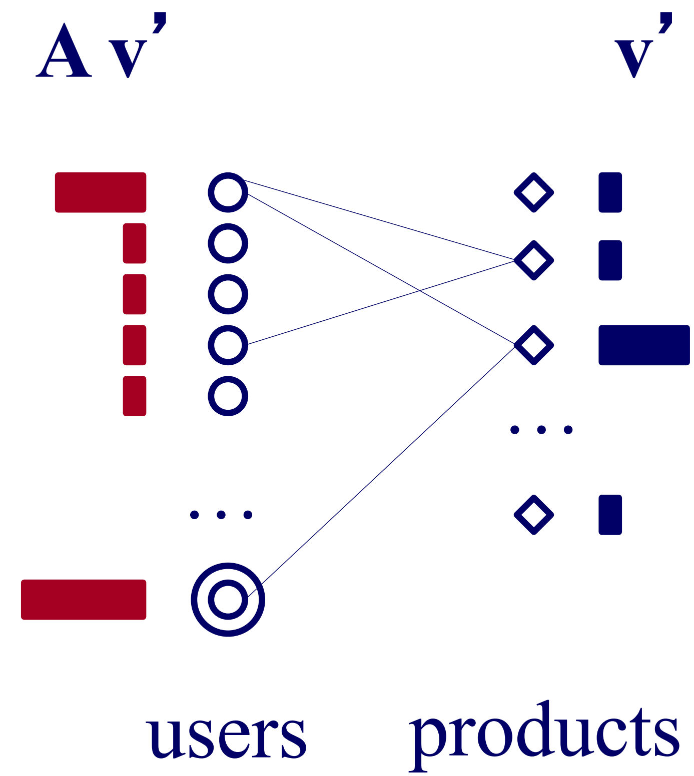




More intuition

- Intuition:
  - $(\mathbf{A}^T \mathbf{A})\ \mathbf{v}'$
  - $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v}'$



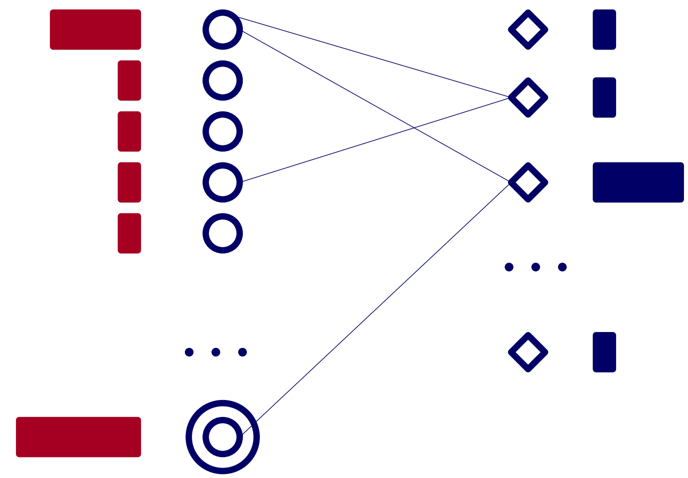


More intuition

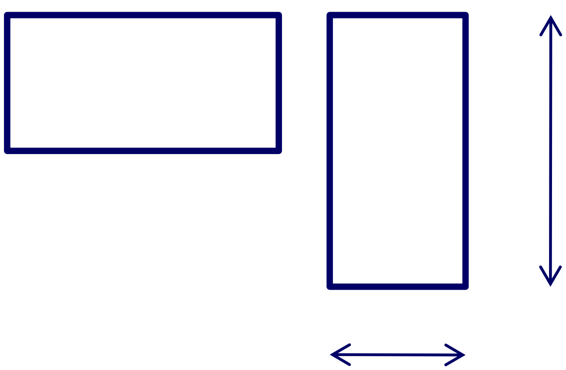
- Intuition:
  - $(\mathbf{A}^T \mathbf{A})\ \mathbf{v'}$
  - $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v'}$

similarities
to Smith

$\mathbf{A\ v'}$ $\mathbf{v'}$



users products

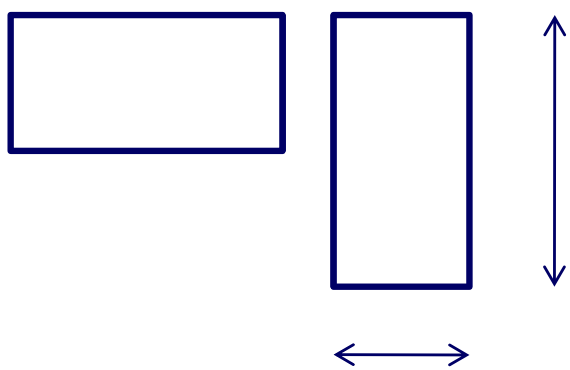


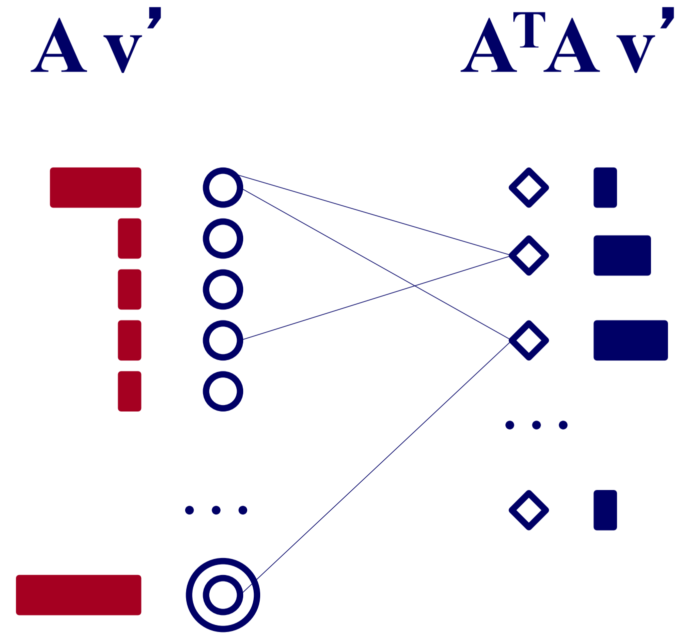
users

products

More intuition

- Intuition:
  - $(A^T A)\ v'$
  - $(A^T A)^k\ v'$



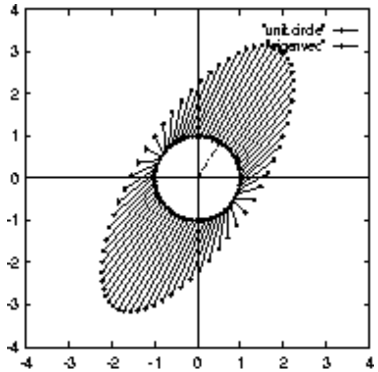


More intuition

- Intuition:
  - $(\mathbf{A}^T \mathbf{A})\ \mathbf{v}'$ what Smith's 'friends' like
  - $(\mathbf{A}^T \mathbf{A})^k\ \mathbf{v}'$ what k-step-away-friends like

(ie., after *k* steps, we get what everybody likes, and Smith's initial opinions don't count)

More intuition



~Markov chain: initial state does not matter$^{(*)}$

~ matrix-vector mult. -> eigenvector

(ie., after *k* steps, we get what everybody likes, and Smith's initial opinions don't count)

Proof, and intuition

Kleinberg's algorithm - results

Eg., for the query 'java':

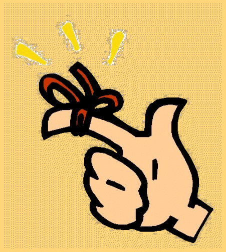
0.328 www.gamelan.com

0.251 java.sun.com

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

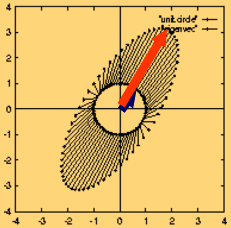
Kleinberg’s algorithm - discussion

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

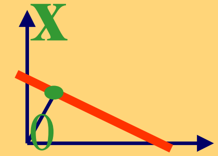


Conclusions

- Q1: most important node(s) in a graph?
  - A1.1: HITS (= SVD)
  - A1.2: PageRank (= fixed point)



- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
  - A2: SVD (< - > Moore-Penrose pseudo-inverse)



SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - Kleinberg's algorithm (HITS)
  - Google algorithm
  - query feedbacks
- Conclusions

PageRank (google)

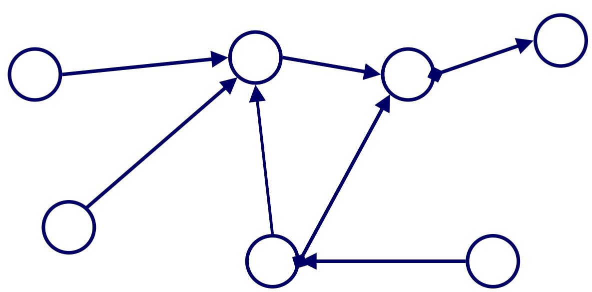
Larry Page   Sergey Brin

- Brin, Sergey and Lawrence Page (1998). *Anatomy of a Large-Scale Hypertextual Web Search Engine*. 7th Intl World Wide Web Conf.

Carnegie Mellon

Problem: PageRank

Given a directed graph, find its most interesting/central node

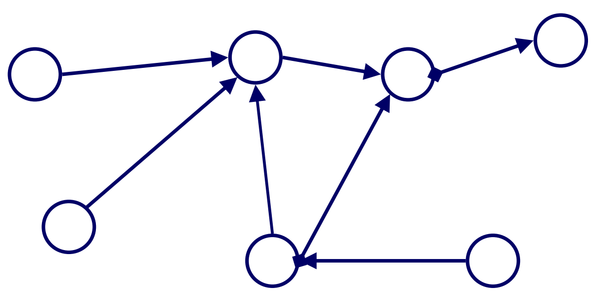


A node is important, if it is connected with important nodes (recursive, but OK!)

Problem: PageRank - solution

Given a directed graph, find its most interesting/central node

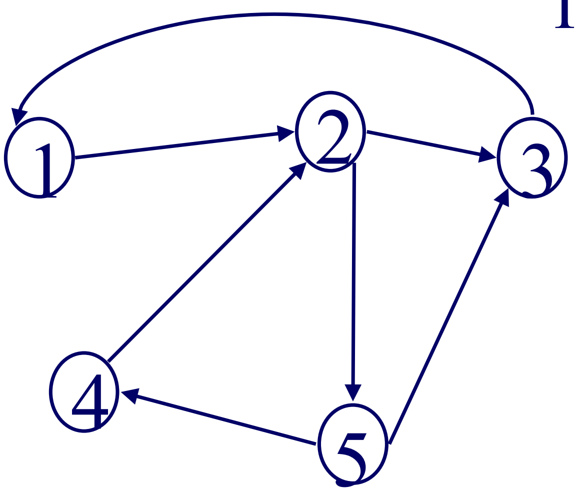
Proposed solution: Random walk; spot most 'popular' node (-> steady state prob. (ssp))



A node has high ssp, if it is connected with high ssp nodes (recursive, but OK!)

(Simplified) PageRank algorithm

- Let **A** be the adjacency matrix;
- let **B** be the transition matrix: transpose, column-normalized - then



From

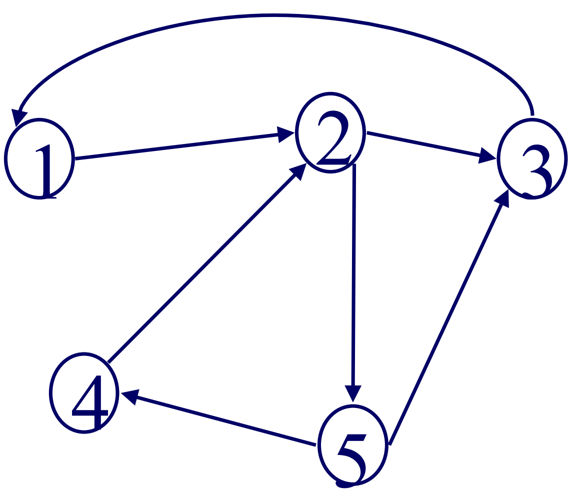
To

B

$$\begin{bmatrix} & & 1 & & \\ 1 & & & 1 & \\ & 1/2 & & & 1/2 \\ & & & & 1/2 \\ & 1/2 & & & \end{bmatrix} \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix} = \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix}$$

(Simplified) PageRank algorithm

- $\mathbf{B \; p = p}$



$$\mathbf{B} \qquad\qquad \mathbf{p} \; = \; \mathbf{p}$$

$$\begin{bmatrix} & & 1 & & \\ 1 & & & 1 & \\ & 1/2 & & & 1/2 \\ & & & & 1/2 \\ & 1/2 & & & \end{bmatrix} \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix} = \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix}$$

Carnegie Mellon

(Simplified) PageRank algorithm

- **B p** = 1 * **p**
- thus, **p** is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a **p** exist?
  - **p** exists if **B** is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

(Simplified) PageRank algorithm

- **B p** = 1 * **p**
- thus, **p** is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a **p** exist?
  - **p** exists if **B** is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

All<-> all

(Simplified) PageRank algorithm

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

Full version of algo: with occasional random jumps

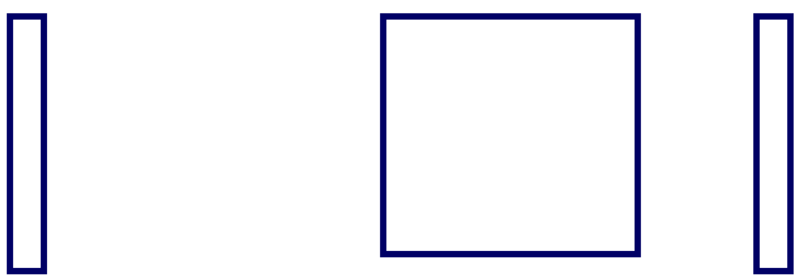
Why? To make the matrix irreducible

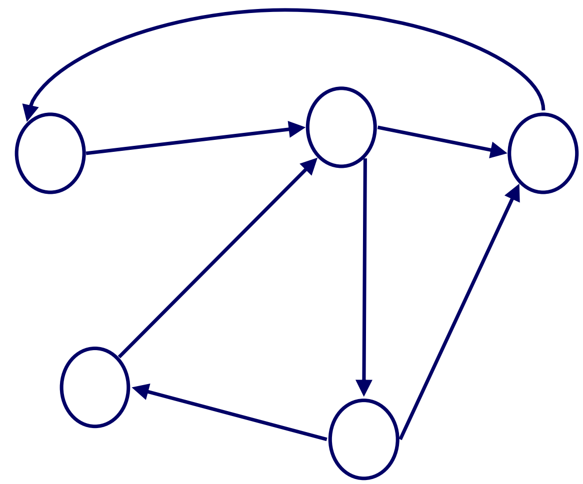
Full Algorithm

- With probability *1-c*, fly-out to a random node
- Then, we have

$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\ \mathbf{1} \Rightarrow$$

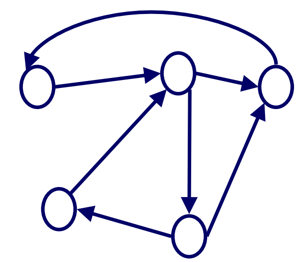
$$\mathbf{p} = (1-c)/n\ [\mathbf{I} - c\,\mathbf{B}]^{-1}\ \mathbf{1}$$





Full Algorithm

- With probability *1-c*, fly-out to a random node
- Then, we have



$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\ \mathbf{1} \Rightarrow$$

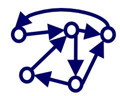
$$\mathbf{p} = (1-c)/n\ [\mathbf{I} - c\,\mathbf{B}]^{-1}\ \mathbf{1}$$

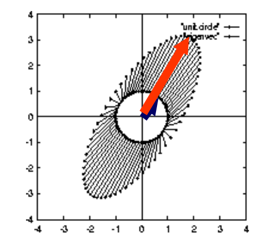
$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix}$$

Alternative notation – eigenvector viewpoint

M Modified transition matrix

$$\mathbf{M} = c\,\mathbf{B} + (1\text{-}c)/n\ \ \mathbf{1}\ \ \mathbf{1}^T$$

Then  + **CLIQUE**



$$\mathbf{p} = \mathbf{M}\,\mathbf{p}$$

That is: the steady state probabilities =

PageRank scores form the *first eigenvector* of the 'modified transition matrix'

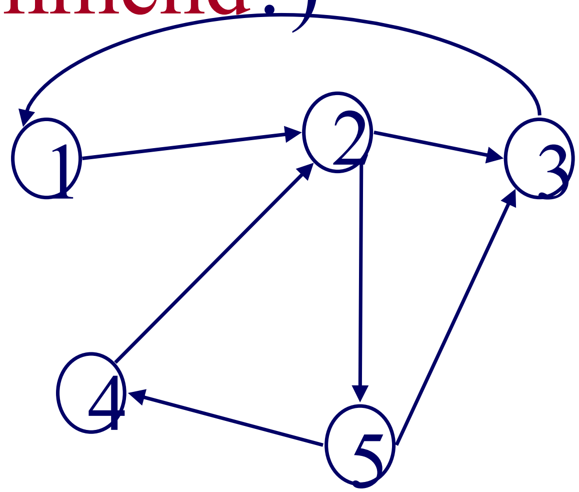
Personalized P.R.

- Taher H. Haveliwala. 2002. *Topic-sensitive PageRank*. (WWW '02). 517-526. http://dx.doi.org/10.1145/511446.511513

Carnegie Mellon

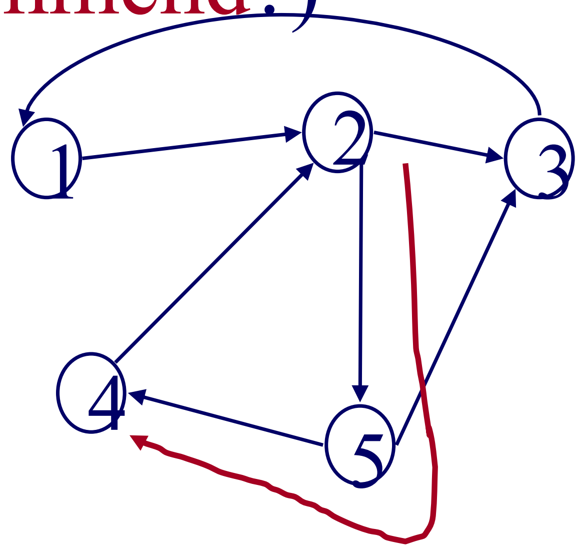
Extension: Personalized P.R.

- How close is ‘4’ to ‘2’?
- (or: if I like page/node ‘2’, what else would you recommend?)



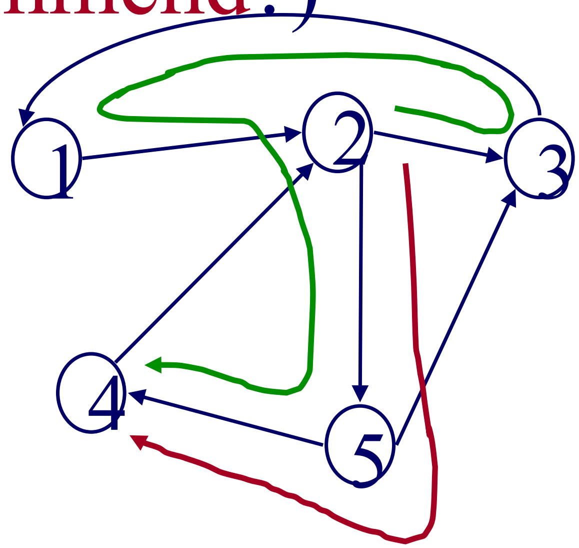
Extension: Personalized P.R.

- How close is ‘4’ to ‘2’?
- (or: if I like page/node ‘2’, what else would you recommend?)



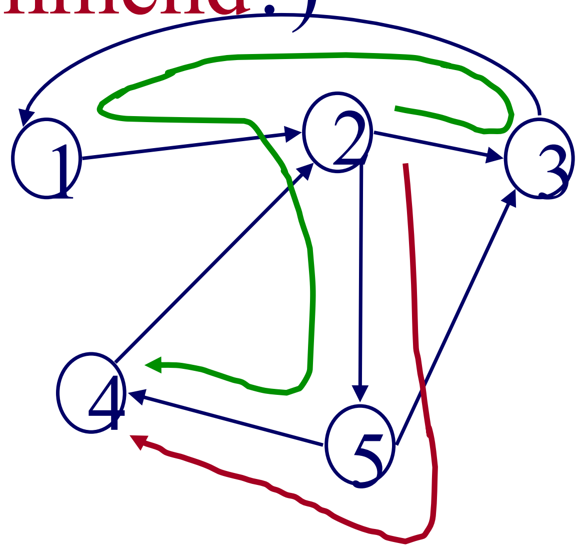
Extension: Personalized P.R.

- How close is ‘4’ to ‘2’?
- (or: if I like page/node ‘2’, what else would you recommend?)



Extension: Personalized P.R.

- How close is '4' to '2'?
- (or: if I like page/node '2', what else would you recommend?)



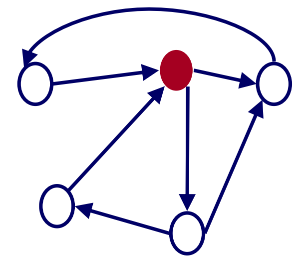
High score (A -> B) if
- Many
- Short
- Heavy

paths A->B

DETAILS

Extension: Personalized P.R.

- With probability *1-c*, fly-out to your favorite node(s)
- Then, we have



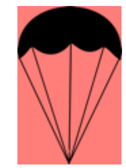
$$\boxed{\mathbf{p} = c\ \mathbf{B}\ \mathbf{p} + (1\text{-}c)/n\ \overset{\vec{e}}{\cancel{\mathbf{1}}} \Rightarrow}$$

$$\mathbf{p} = (1\text{-}c)/\cancel{n}\ [\mathbf{I} - c\ \mathbf{B}]^{-1}\ \overset{\vec{e}}{\cancel{\mathbf{1}}}$$

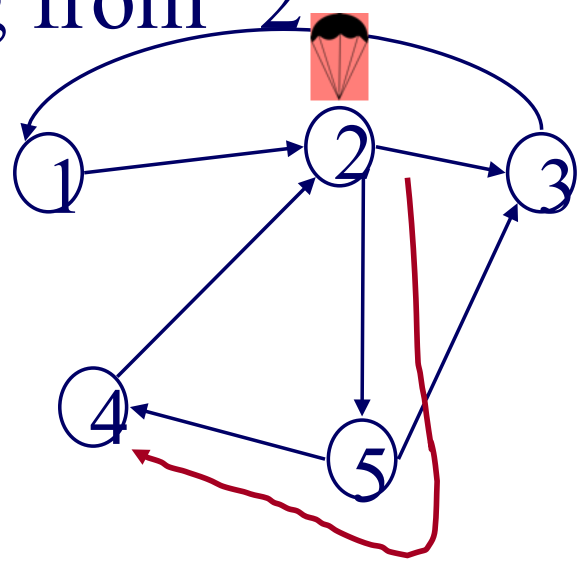
$$\vec{e} = \begin{bmatrix} \emptyset \\ 1 \\ \vdots \\ \emptyset \end{bmatrix}$$

Carnegie Mellon

Extension: Personalized P.R.

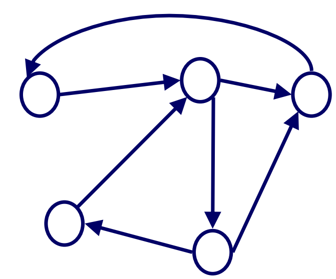


- How close is ‘4’ to ‘2’?
- A: compute Personalized P.R. of ‘4’, restarting from ‘2’



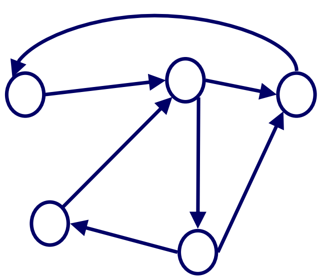
Extension: Personalized P.R.

- How close is ‘4’ to ‘2’?
- A: compute Personalized P.R. of ‘4’, restarting from ‘2’
- How to compute it quickly?



Extension: Personalized P.R.

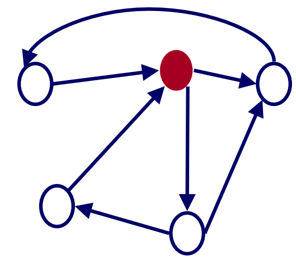
- How close is ‘4’ to ‘2’?
- A: compute Personalized P.R. of ‘4’, restarting from ‘2’
- How to compute it quickly?
- A: ‘Pixie’ algorithm

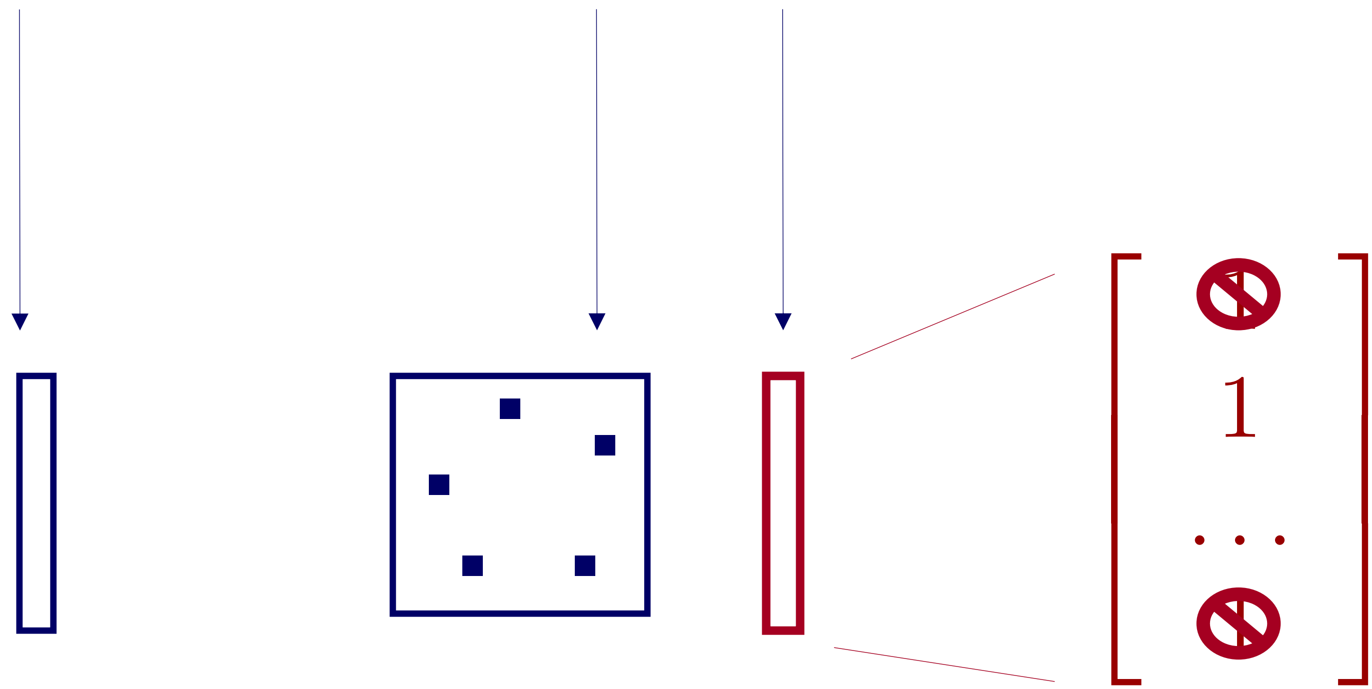


Extension: Personalized P.R.

- Q: Faster computation than:

$$\mathbf{p} = (1\text{-}c)/n \; [\mathbf{I} - c\,\mathbf{B}]^{-1} \; \mathbf{e}$$





Pixie algorithm

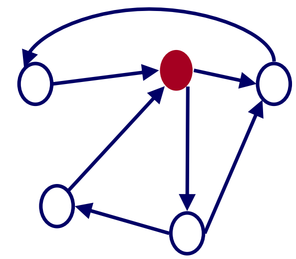
Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, Jure Leskovec: *Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time*. WWW 2018: 1775-1784

https://dl.acm.org/citation.cfm?doid=3178876.3186183

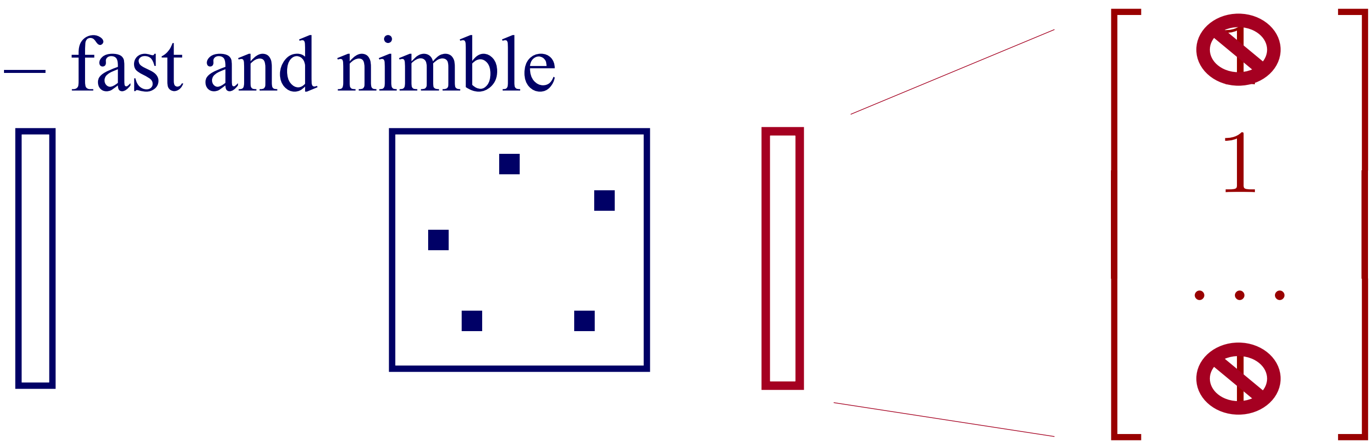
Pixie algorithm

- Q: Faster computation than:

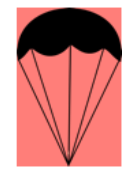
  $$\mathbf{p} = (1\text{-}c)/n \;\; [\mathbf{I} - c\,\mathbf{B}]^{-1} \;\; \mathbf{e}$$

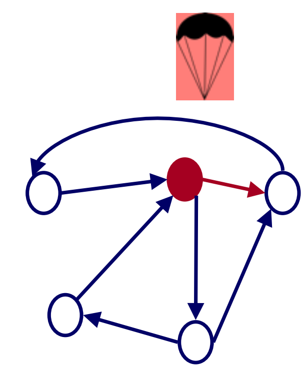


- A: **simulate** a few R.W.
  - keep visit counts C_i
  - fast and nimble



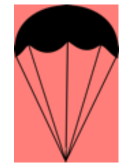
Personalized PageRank algorithm

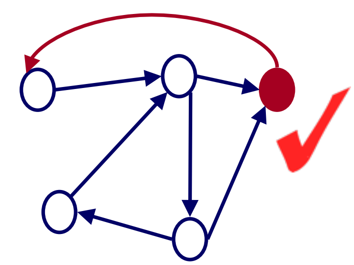




Carnegie Mellon

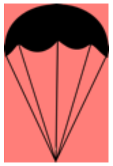
Personalized PageRank algorithm

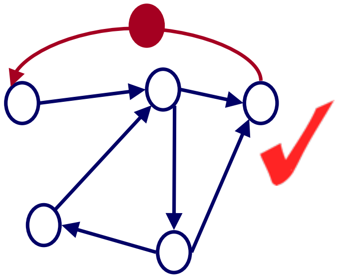




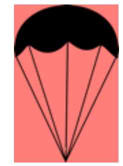
Carnegie Mellon

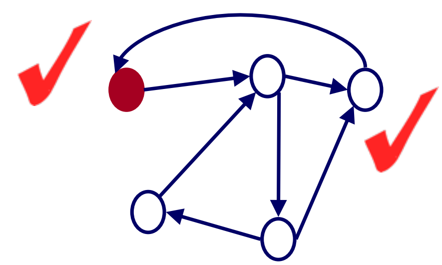
Personalized PageRank algorithm



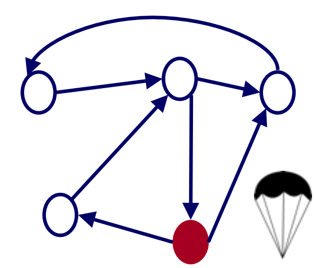


Personalized PageRank algorithm

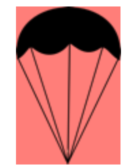


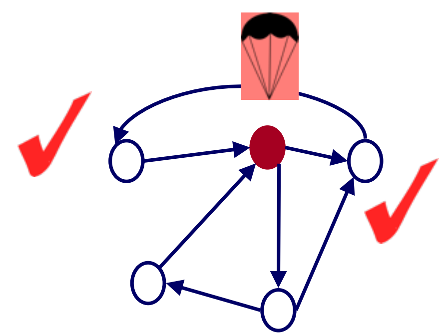


Personalized PageRank algorithm



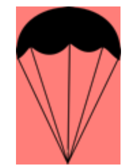
Personalized PageRank algorithm

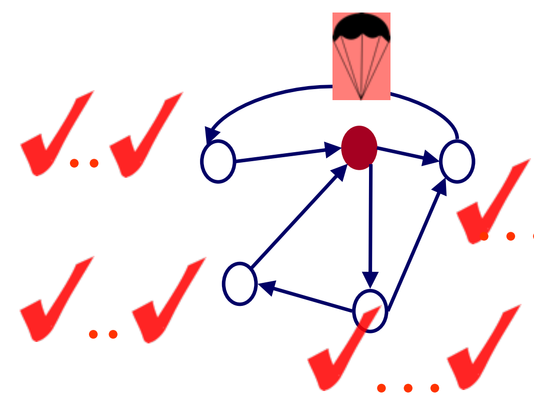




Carnegie Mellon

Personalized PageRank algorithm





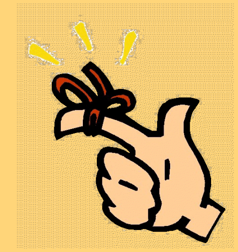
Kleinberg/PageRank - conclusions

SVD helps in graph analysis:

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

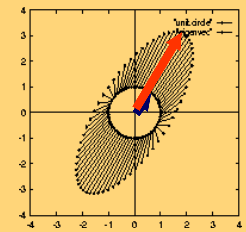
random walk on a graph: steady state probabilities are given by the strongest eigenvector of the transition matrix

Carnegie Mellon

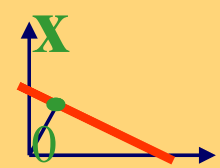


Conclusions

- Q1: most important node(s) in a graph?
  - A1.1: HITS (= SVD)
  - A1.2: PageRank (= fixed point)



- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
  - A2: SVD (< - > Moore-Penrose pseudo-inverse)



SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks – theory, and application
- Conclusions

Least obvious properties

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

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

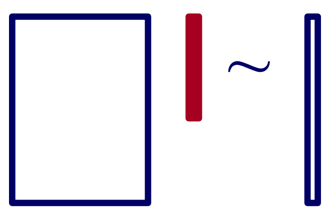
(see Num. Recipes, p. 62)

Least obvious properties

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

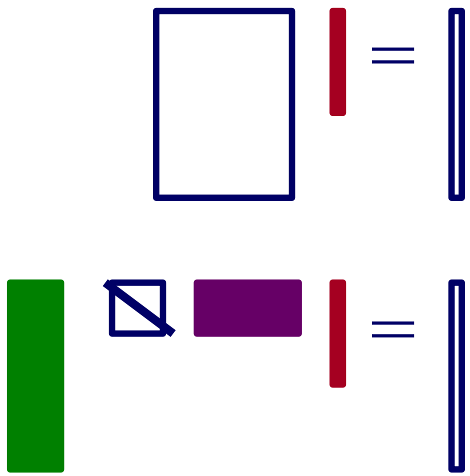
C(1): $\mathbf{A}_{[n \times m]} \, \mathbf{x}_{[m \times 1]} = \mathbf{b}_{[n \times 1]}$

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



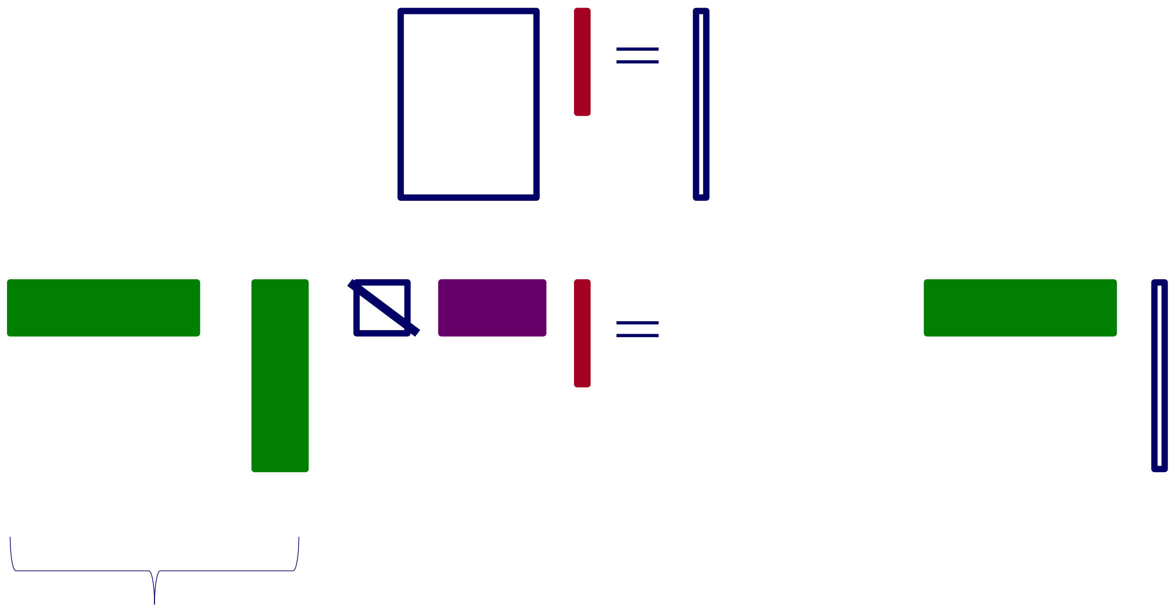
NOT proof

Slowly:



NOT proof

Slowly:

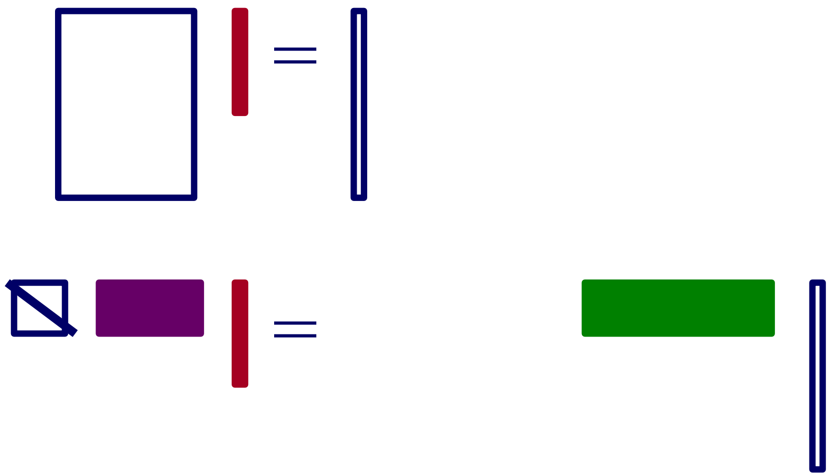


Identity
U: column-
orthonormal

Carnegie Mellon

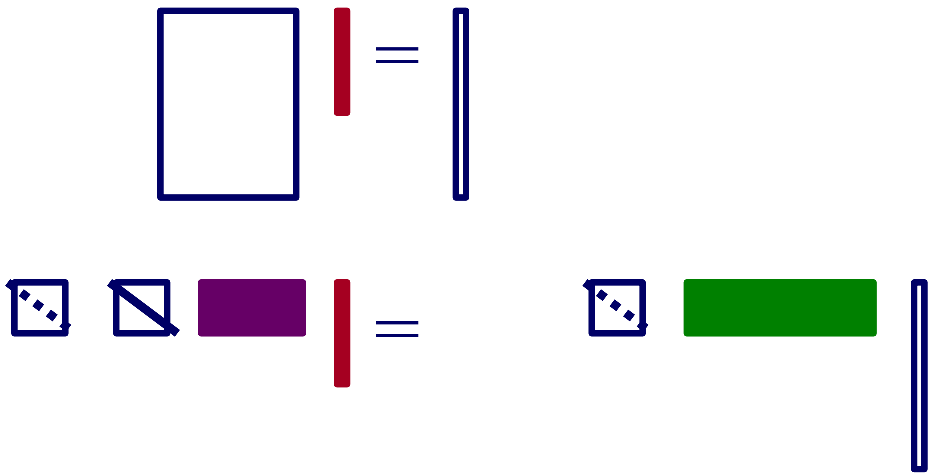
NOT proof

Slowly:



NOT proof

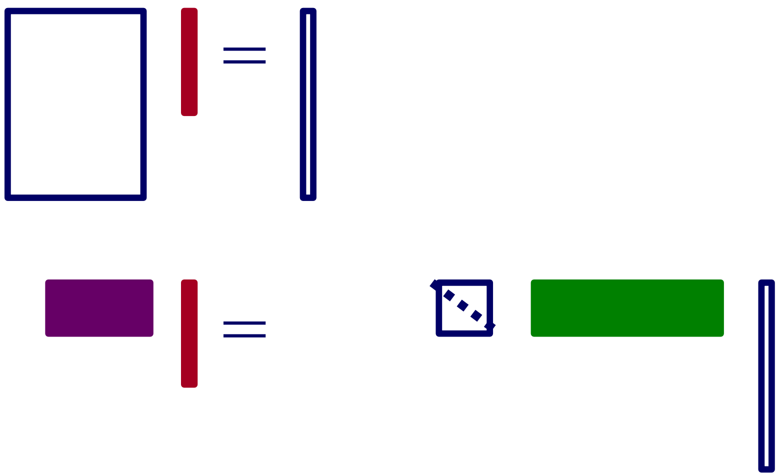
Slowly:



Carnegie Mellon

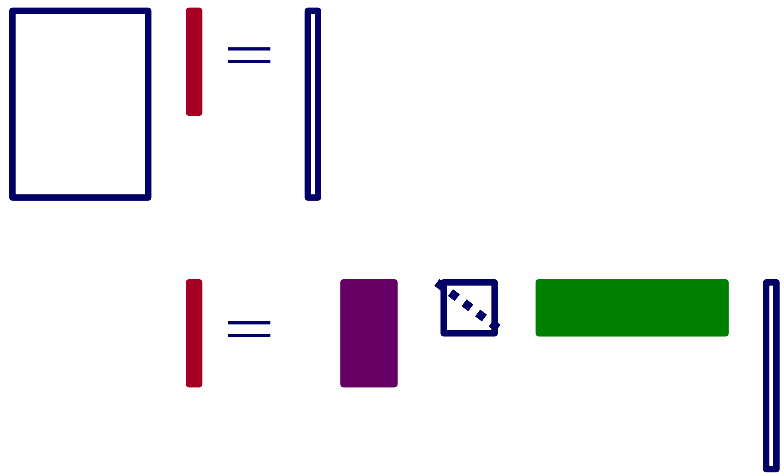
NOT proof

Slowly:



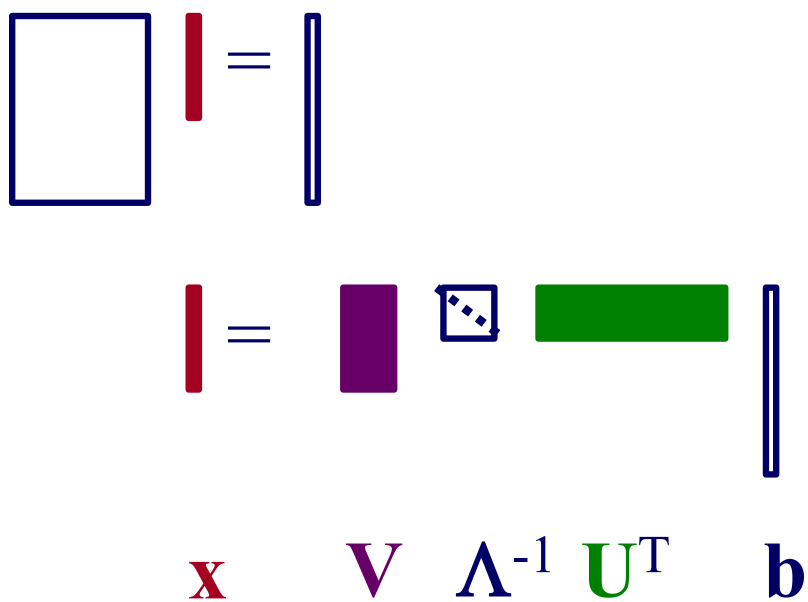
NOT proof

Slowly:



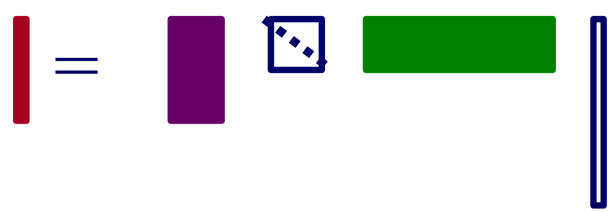
NOT proof

Slowly:



Slowly:

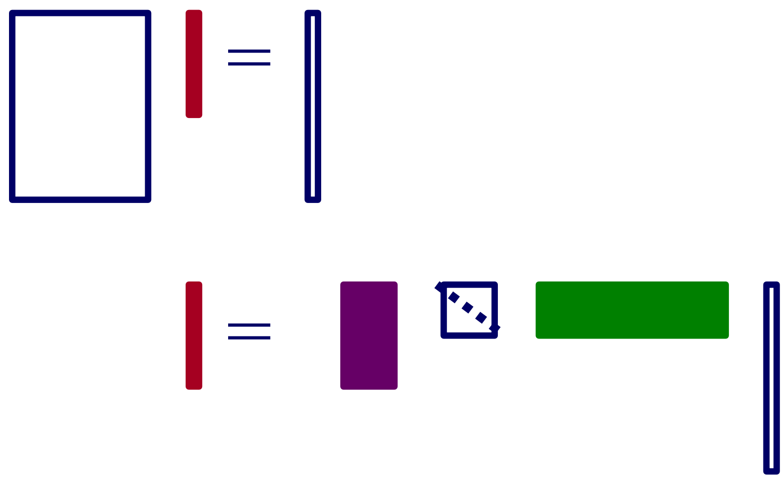
Important: **DROP** small values of Λ
(say, $< 10^{-6} * \lambda_1$)



$\mathbf{x} = \mathbf{V} \; \Lambda^{-1} \; \mathbf{U}^T \; \mathbf{b}$

Recursive Least Squares (RLS)

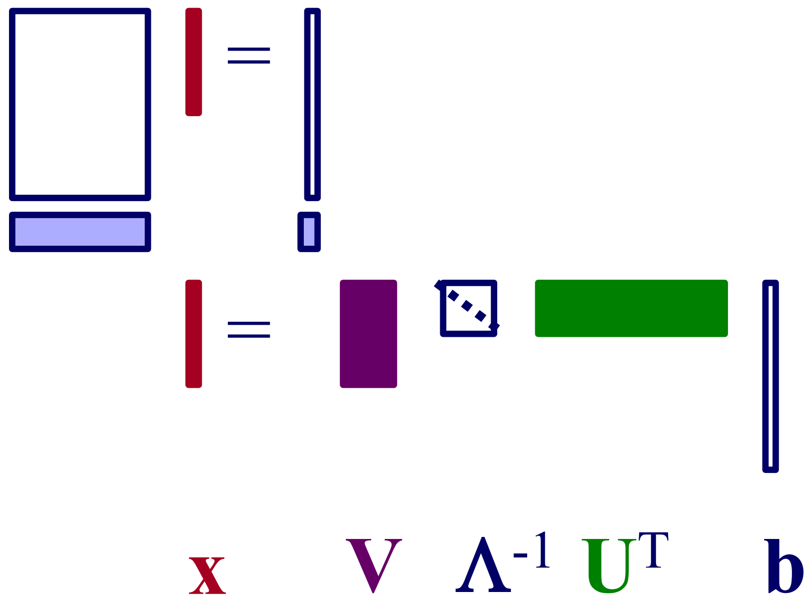
Can add to **A** and **b**;



$\mathbf{x} \qquad \mathbf{V} \quad \mathbf{\Lambda}^{-1} \; \mathbf{U}^{T} \qquad \mathbf{b}$

Recursive Least Squares (RLS)

Can add to **A** and **b**; and incrementally update **x**



Drills:

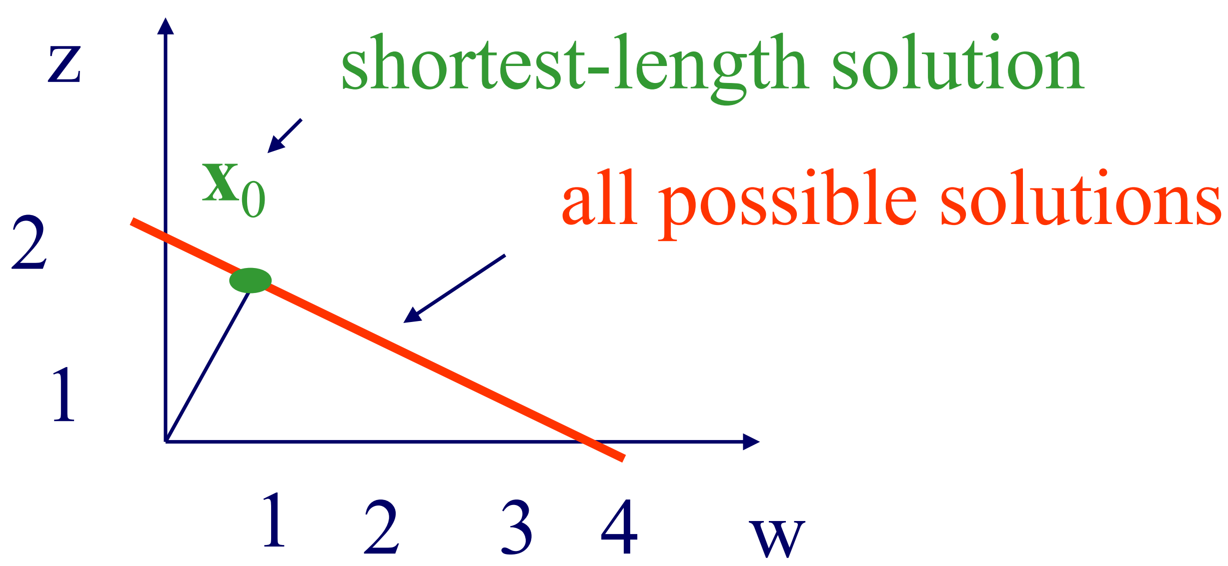
Least obvious properties

Illustration: under-specified, eg

$[1\ 2]\ [w\ z]^T = 4$ (ie, $1\,w + 2\,z = 4$)

A=??

b=??



Exercise

Verify formula:

$\mathbf{A} = [1\ 2] \qquad \mathbf{b} = [4]$

$\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{V}^T$

$\mathbf{U} = ??$

$\mathbf{\Lambda} = ??$

$\mathbf{V} = ??$

$\mathbf{x_0} = \mathbf{V}\ \mathbf{\Lambda}^{(-1)}\ \mathbf{U}^T\ \mathbf{b}$

Carnegie Mellon

Exercise

Verify formula:

$\mathbf{A} = [1\ 2] \qquad \mathbf{b} = [4]$

$\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{V}^T$

$\mathbf{U} = [1]$

$\mathbf{\Lambda} = [\ \mathrm{sqrt}(5)\ ]$

$\mathbf{V} = [\ 1/\mathrm{sqrt}(5) \qquad 2/\mathrm{sqrt}(5)\ ]^T$

$\mathbf{x_0} = \mathbf{V}\ \mathbf{\Lambda}^{(-1)}\ \mathbf{U}^T\ \mathbf{b}$

Exercise

Verify formula:

$\mathbf{A} = [1\ 2] \qquad \mathbf{b} = [4]$

$\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{V}^T$

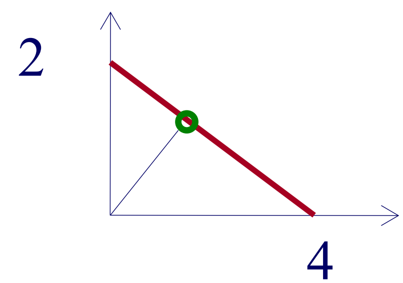
$\mathbf{U} = [1]$

$\mathbf{\Lambda} = [\ \mathrm{sqrt}(5)\ ]$

$\mathbf{V} = [\ 1/\mathrm{sqrt}(5) \quad 2/\mathrm{sqrt}(5)\ ]^T$

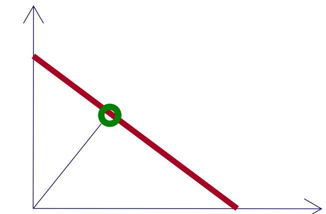
$\mathbf{x_0} = \mathbf{V}\ \mathbf{\Lambda}^{(-1)}\ \mathbf{U}^T\ \mathbf{b} = [\ 1/5 \quad 2/5]^T\ [4]$

$= [4/5 \quad 8/5]^T: \quad w = 4/5,\ z = 8/5$



Exercise

Verify formula:



Show that w= 4/5, z = 8/5 is

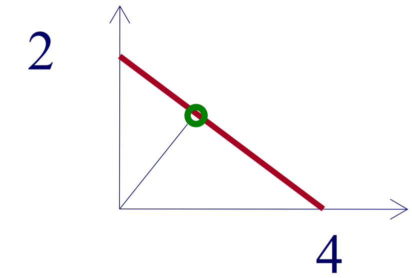
(a) A solution to 1*w + 2*z = 4 and

(b) Minimal (wrt Euclidean norm)

Carnegie Mellon

Exercise

Verify formula:



Show that w= 4/5, z = 8/5 is

(a) A solution to 1*w + 2*z = 4 and

A: easy

(b) Minimal (wrt Euclidean norm)

A: [4/5   8/5] is perpenticular to [2   -1]

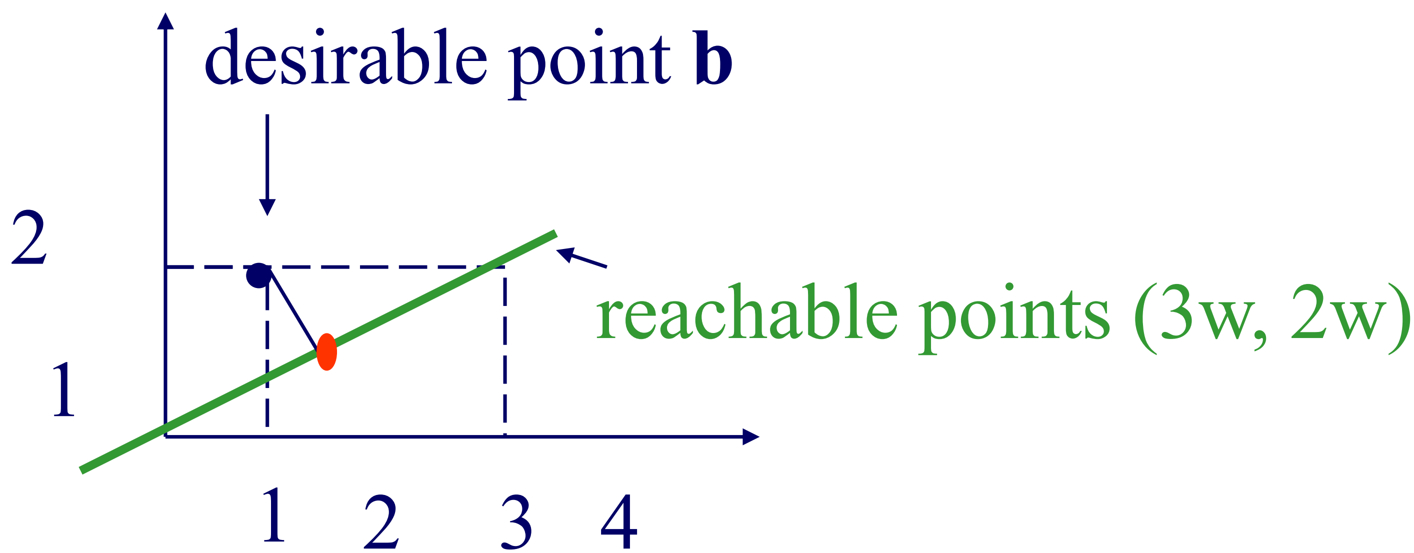
Least obvious properties – cont' d

Illustration: over-specified, eg

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

A=??

b=??



Exercise

Verify formula:

$\mathbf{A} = [3\ 2]^T \qquad \mathbf{b} = [\ 1\ \ 2]^T$

$\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{V}^T$

$\mathbf{U} = ??$

$\mathbf{\Lambda} = ??$

$\mathbf{V} = ??$

$\mathbf{x_0} = \mathbf{V}\ \mathbf{\Lambda}^{(-1)}\ \ \mathbf{U}^T\ \mathbf{b}$

Exercise

Verify formula:

$\mathbf{A} = [3\ 2]^T \quad \mathbf{b} = [\ 1\ \ 2]^T$

$\mathbf{A} = \mathbf{U}\ \mathbf{\Lambda}\ \mathbf{V}^T$

$\mathbf{U} = [\ 3/\text{sqrt}(13) \quad 2/\text{sqrt}(13)\ ]^T$

$\mathbf{\Lambda} = [\ \text{sqrt}(13)\ ]$

$\mathbf{V} = [\ 1\ ]$

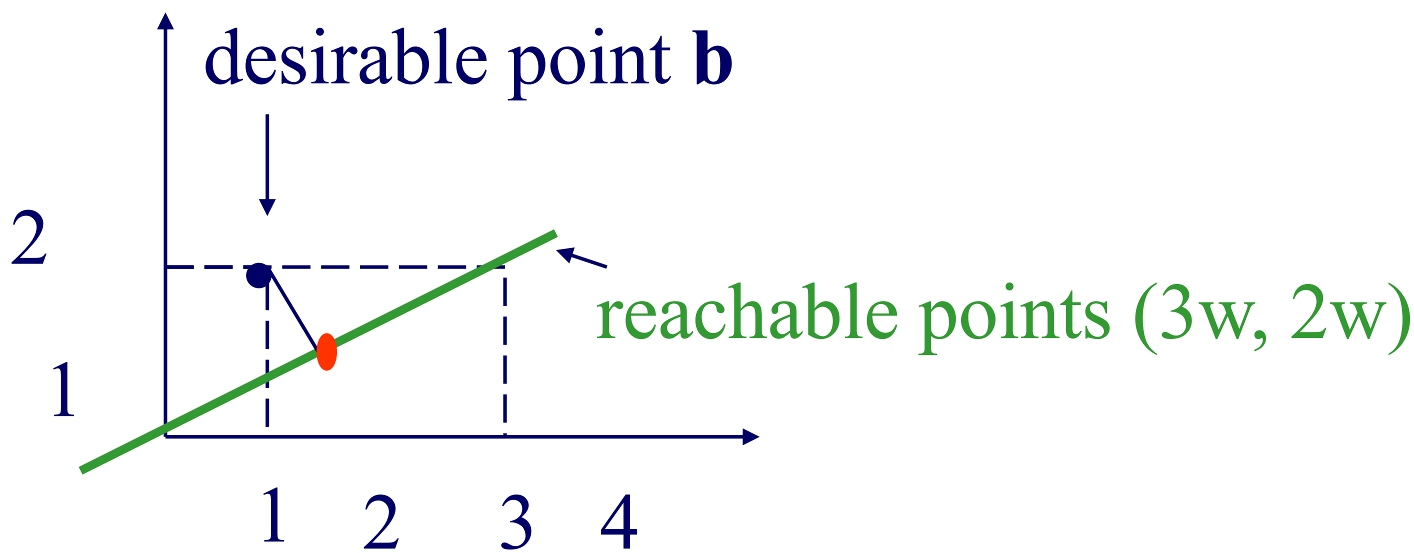
$\mathbf{x_0} = \mathbf{V}\ \mathbf{\Lambda}^{(-1)}\ \mathbf{U}^T\ \mathbf{b} = [\ 7/13\ ]$

Exercise

Verify formula:

$[3\ 2]^T\ [7/13] = [1\ 2]^T$

$[21/13\ \ 14/13\ ]^T$ -> 'red point'

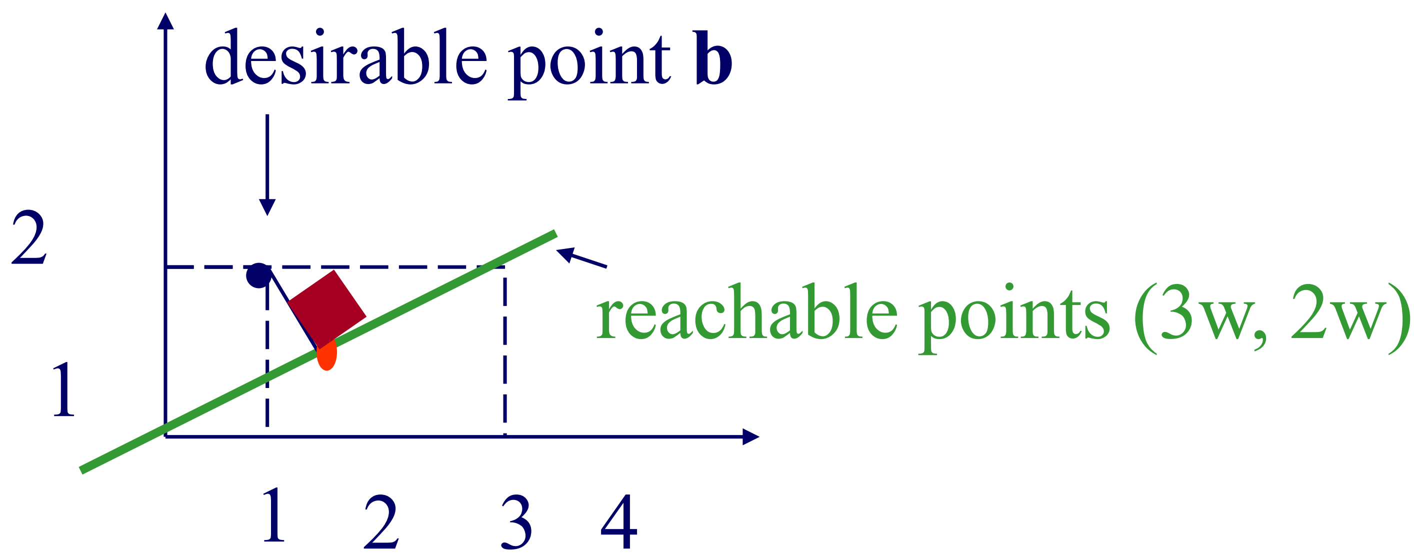


Exercise

Verify formula:

$[3\ 2]^T\ [7/13] = [1\ 2]^T$

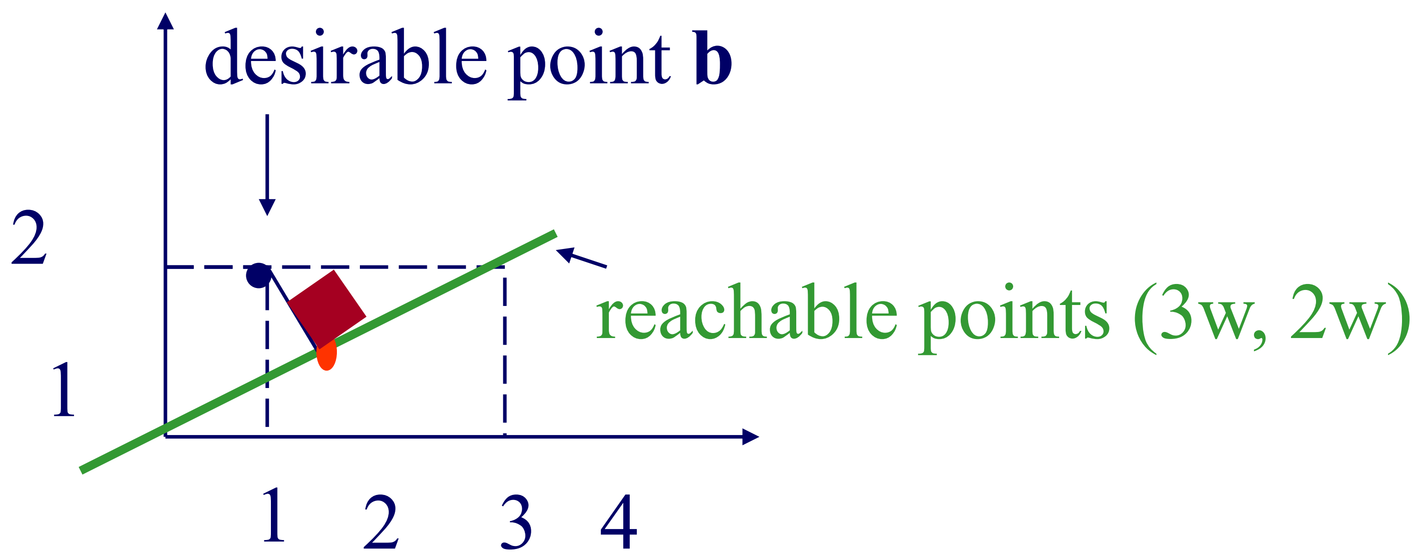
$[21/13\ \ 14/13\ ]^T$ -> 'red point' - <u>perpenticular</u>?



Verify formula:

A: [3 2] . ([1 2] – [21/13   14/13]) =

[3 2] . [-8/13   12/13] = [3 2] . [-2   3] = 0



Carnegie Mellon

SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks – theory, and application
- Conclusions

Query feedbacks

[Chen & Roussopoulos, sigmod 94]

Sample problem:

estimate selectivities (e.g., '*how many movies were made between 1940 and 1945*?'

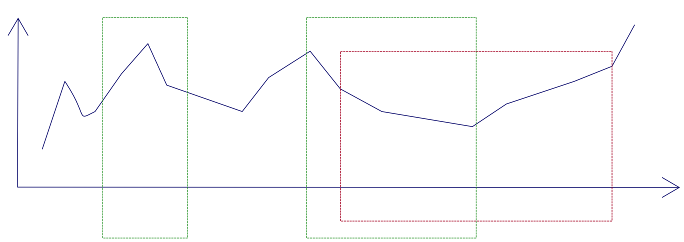
for query optimization,

LEARNING from the query results so far!!

Query feedbacks

- Given: past queries and their results
  - #movies(1925,1935) = 52
  - #movies(1948, 1990) = 123
  - …
  - And a new query, say #movies(1979,1980)?
- Give your best estimate

#movies

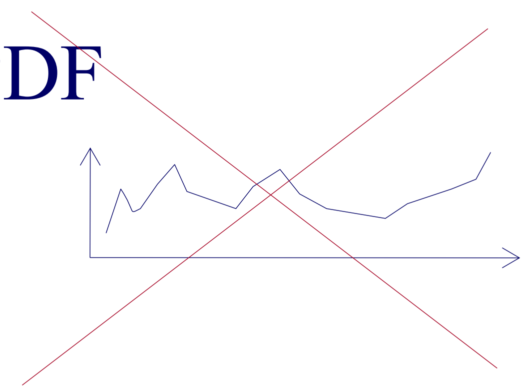


year

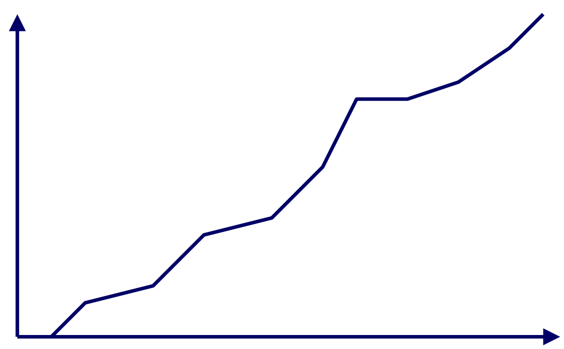
Query feedbacks

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

PDF



count, so far



year

Query feedbacks

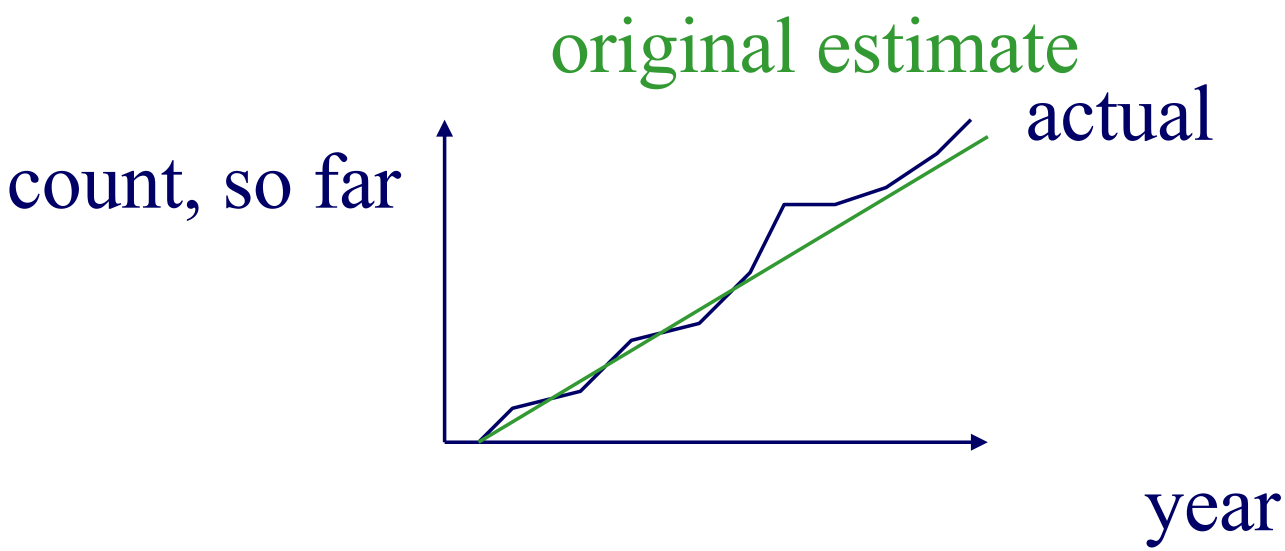
For example

F(x) = # movies made until year 'x'

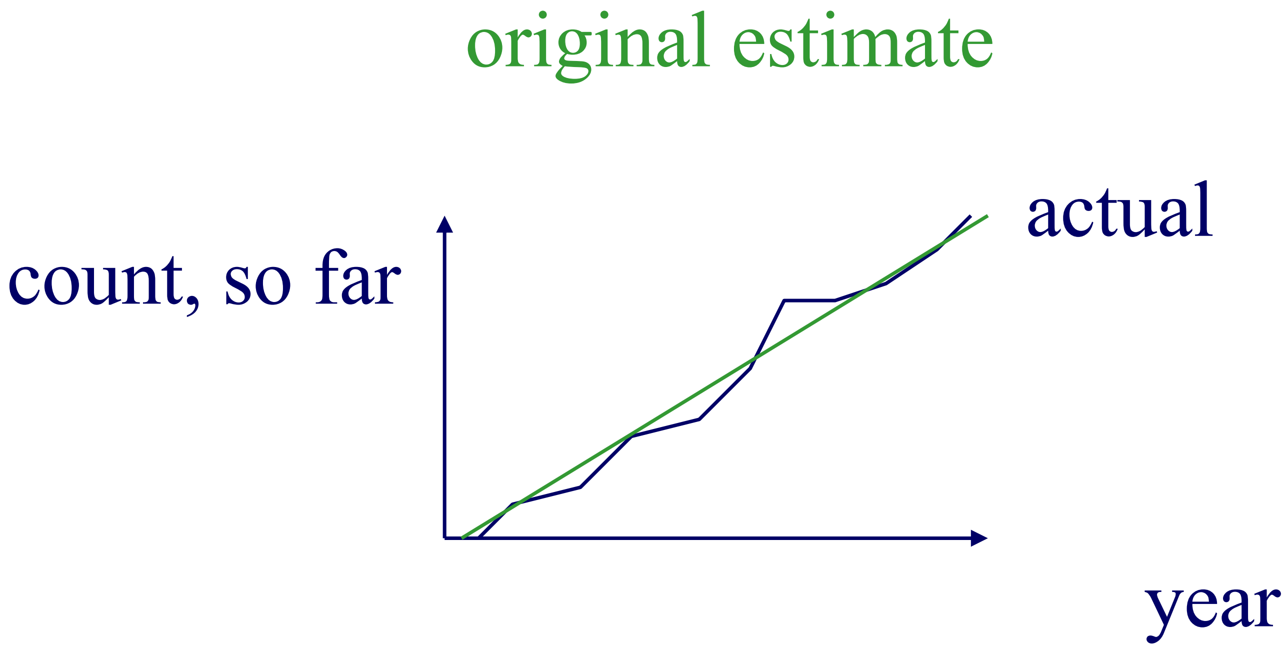
$$= a_1 + a_2 * x + a_3 * x^2 + \ldots a_7 * x^6$$

Query feedbacks

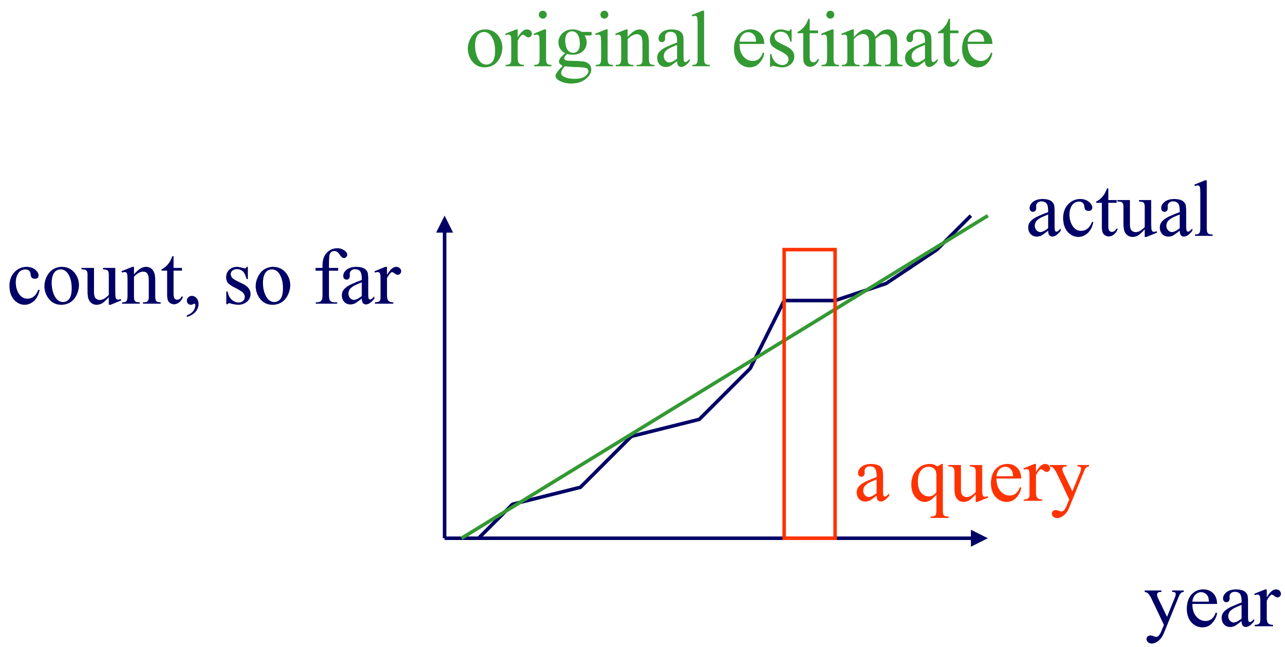
GREAT idea #2: adapt your model, as you see the actual counts of the actual queries



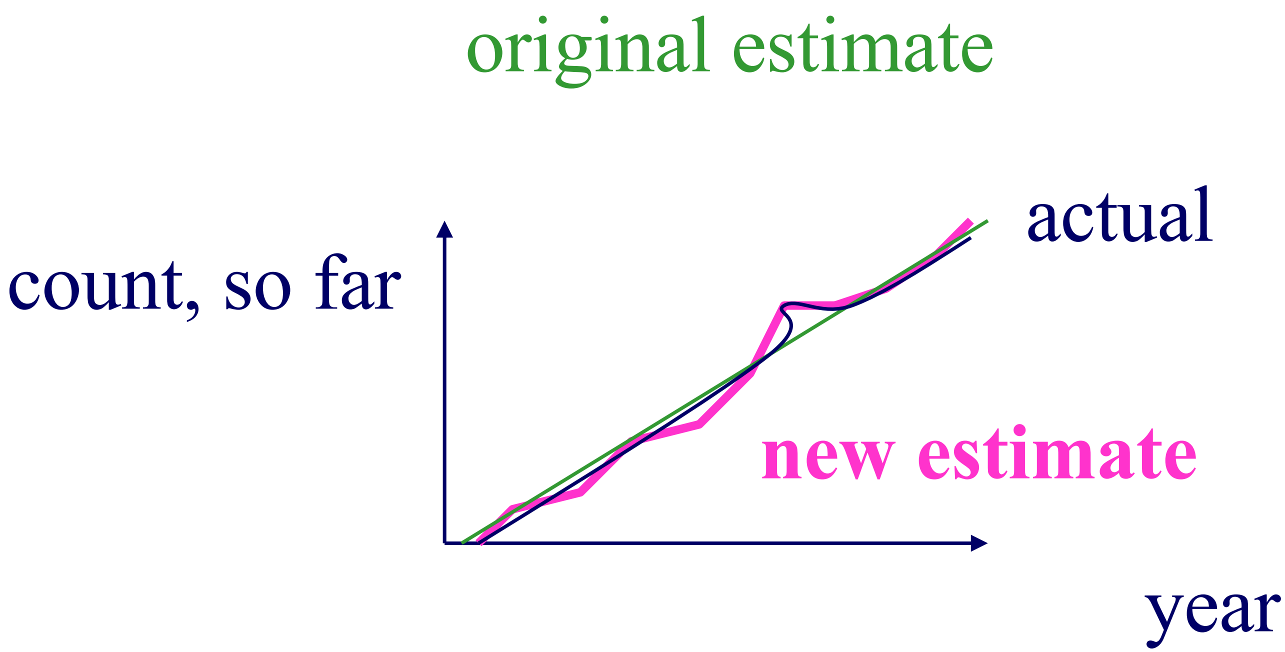
Query feedbacks



Query feedbacks



Query feedbacks



Query feedbacks

Eventually, the problem becomes:

- estimate the parameters a_1, ... a_7 of the model
- to minimize the least squares errors from the real answers so far.

Formally:

Query feedbacks

Formally, with *n* (say, =1000) queries and 6-th degree polynomials:

$$
\underbrace{\begin{bmatrix} X11 & X12 & & & X17 \\ & & & & \\ & & & & \\ & & & & \\ Xn1 & Xn2 & & & Xn7 \end{bmatrix}}_{1{,}000 \times 7}
\underbrace{\begin{bmatrix} a1 \\ a2 \\ \\ \\ a7 \end{bmatrix}}_{7 \times 1}
=
\underbrace{\begin{bmatrix} b1 \\ b2 \\ \\ \\ bn \end{bmatrix}}_{1{,}000 \times 1}
$$

Query feedbacks

What is $X_{i,j}$?

What is b_i ?

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

1,000 x 7 7 x 1 1,000 x 1

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

For example, for query '*find the count of movies during (1920-1932)*' :

$$a_1 + a_2 * 1932 + a_3 * 1932**2 + \dots$$

-

$$(a_1 + a_2 * 1920 + a_3 * 1920**2 + \dots)$$

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

Query feedbacks

And thus X11 = 0; X12 = 1932-1920, etc

$$a_1 + a_2 * 1932 + a_3 * 1932{*}{*}2 + \ldots$$

$$-$$

$$(a_1 + a_2 * 1920 + a_3 * 1920{*}{*}2 + \ldots)$$

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

Query feedbacks

In matrix form:

X **a** = **b**

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

Carnegie Mellon

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

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

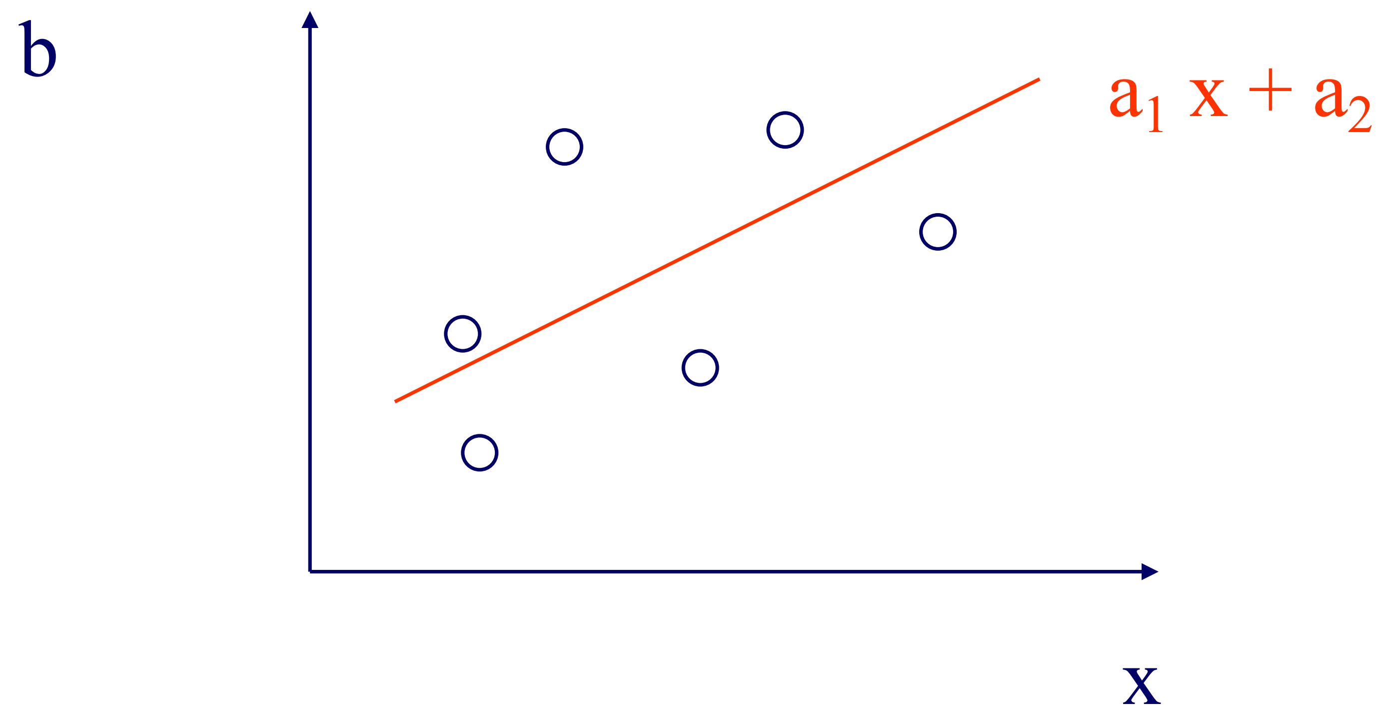
GREAT Idea #3: Use 'Recursive Least Squares', to adapt **a** incrementally.

Details: in paper - intuition:

Query feedbacks - enhancements

Intuition:

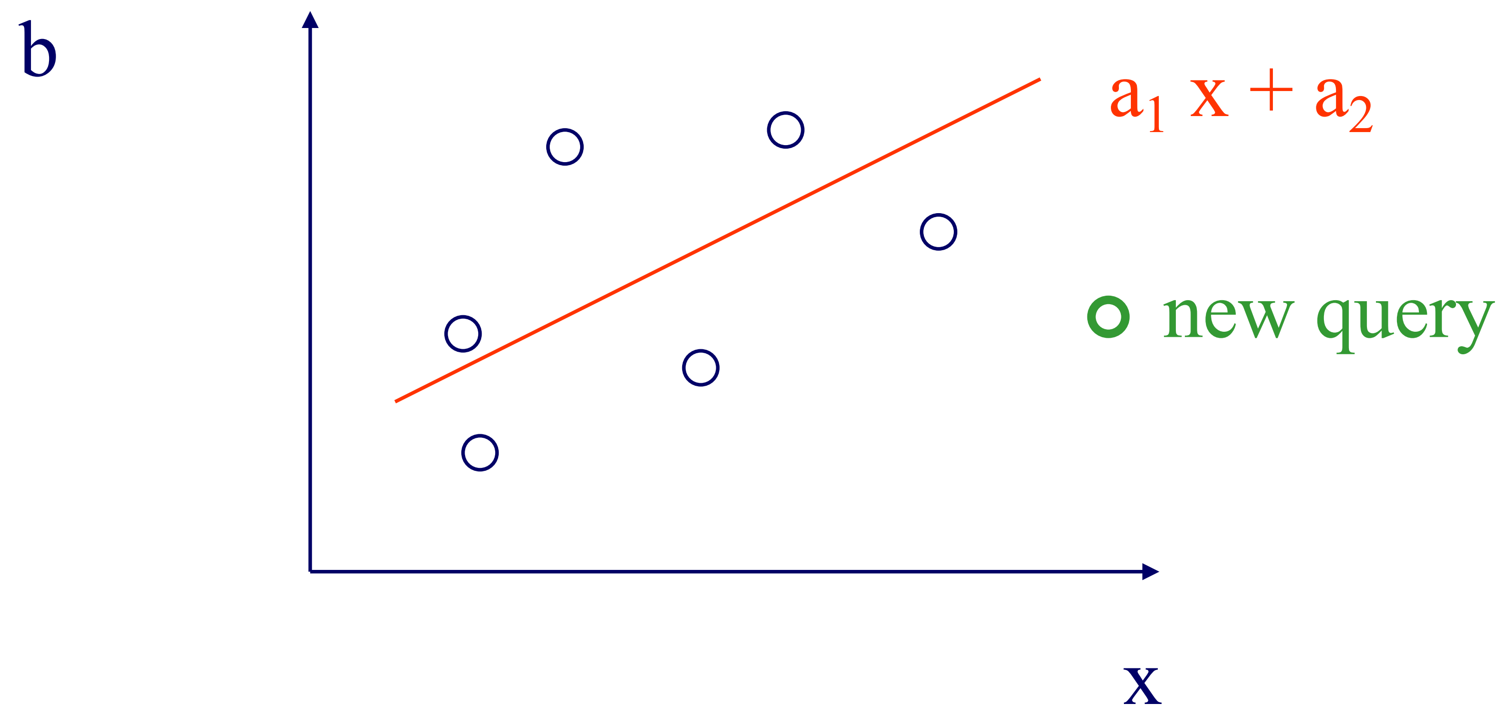
least squares fit



Query feedbacks - enhancements

Intuition:

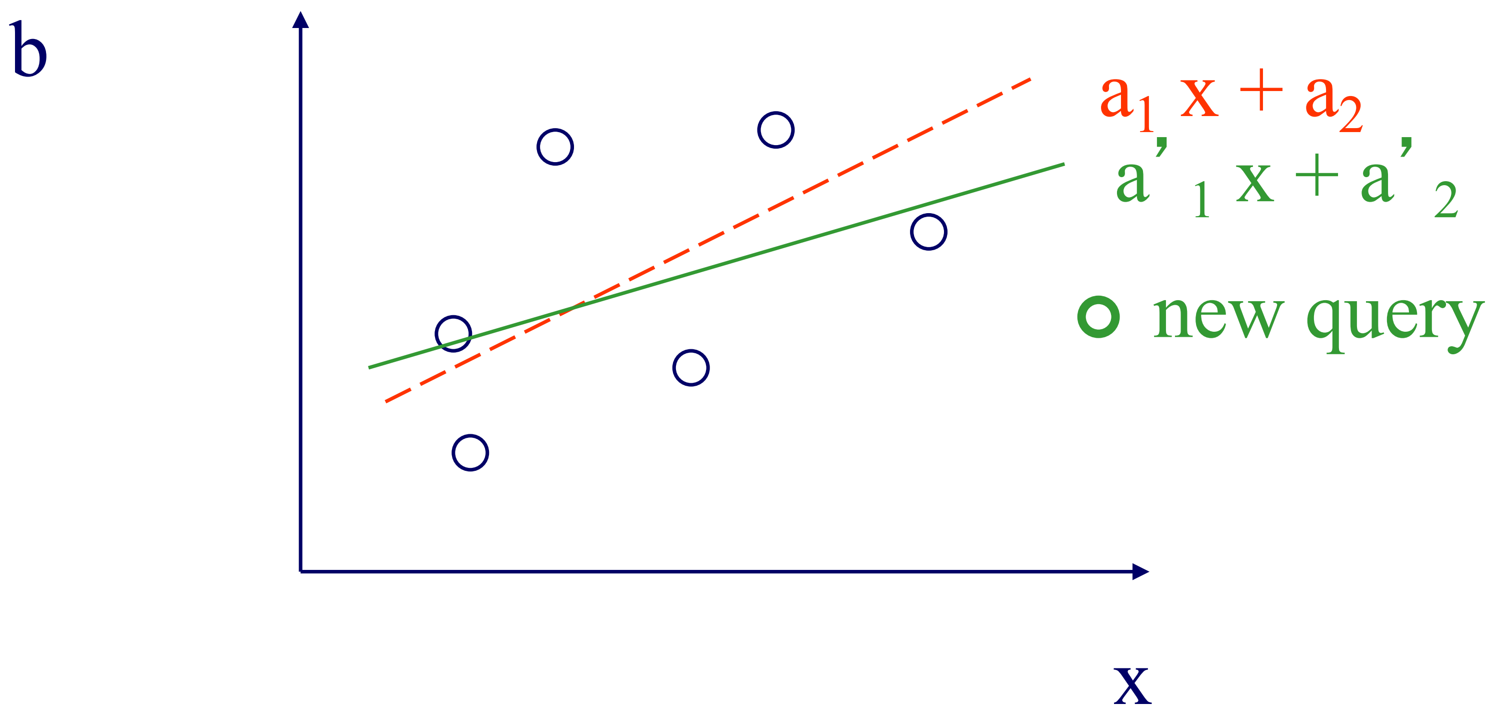
least squares fit



Query feedbacks - enhancements

Intuition:

least squares fit



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)

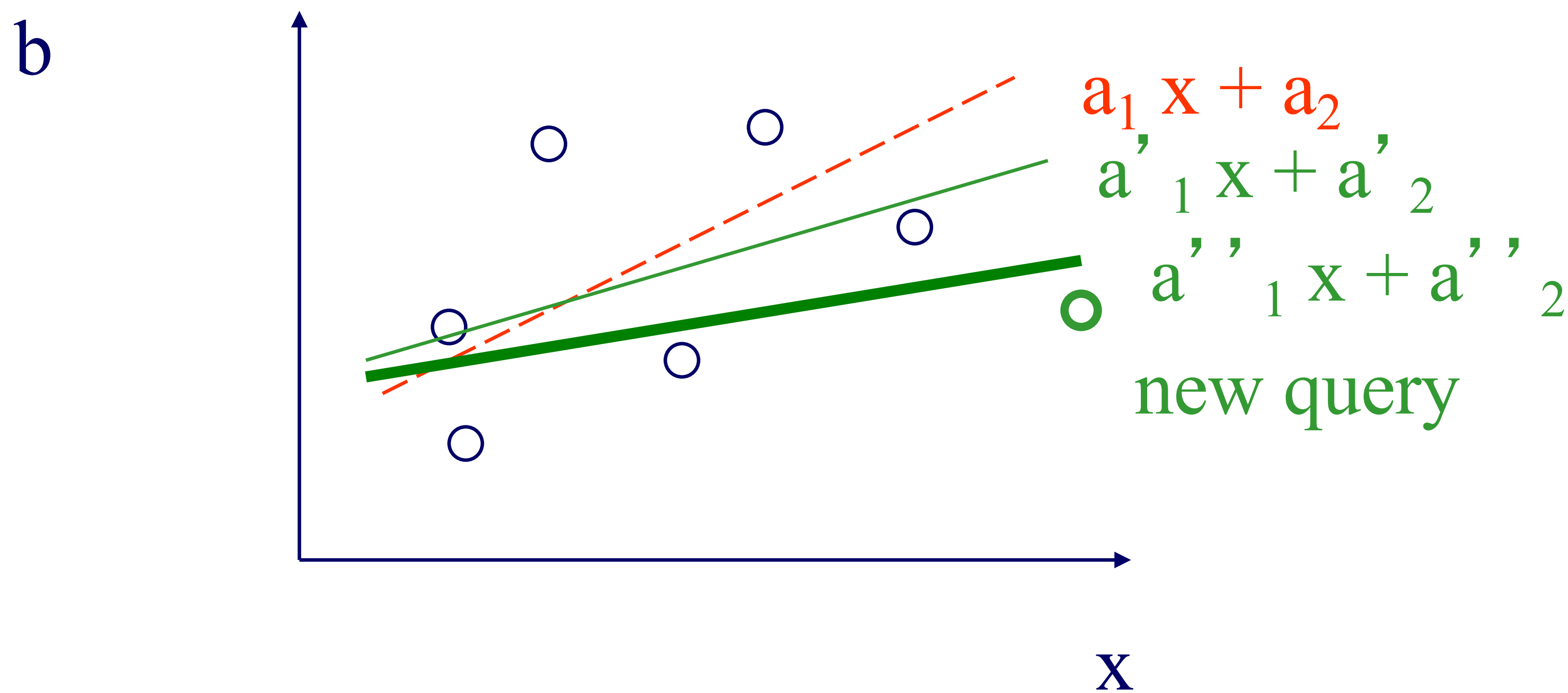
Query feedbacks - enhancements

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...)

Query feedbacks - enhancements

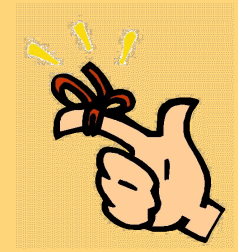
Intuition: least squares fit



Query feedbacks - conclusions

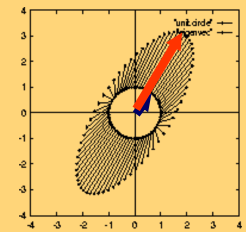
SVD helps find the Least Squares solution, to adapt to query feedbacks

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

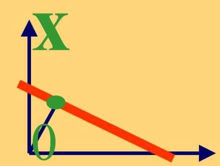


Conclusions

- Q1: most important node(s) in a graph?
  - A1.1: HITS (= SVD)
  - A1.2: PageRank (= fixed point)

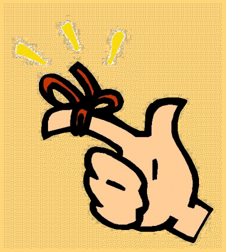


- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
  - A2: SVD (< - > Moore-Penrose pseudo-inverse)



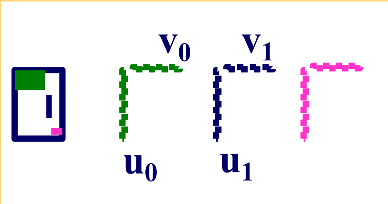
SVD - detailed outline

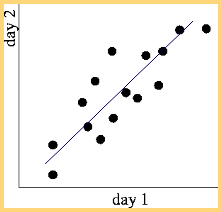
- ...
- Case studies
- SVD properties
- more case studies
  - google/Kleinberg algorithms
  - query feedbacks
- Overall conclusions for SVD

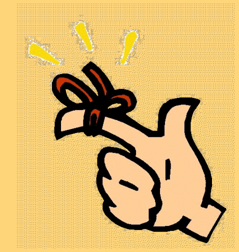


Conclusions (1/3)

- SVD: a **valuable** tool
  - ‘*the importance of SVD can hardly be overstated*’ [Gilbert Strang]
- given a document-term matrix, it finds ‘concepts’ (LSI)
- … and finds blocks (‘EigenSpokes’)
- ... and can reduce dimensionality (KL)

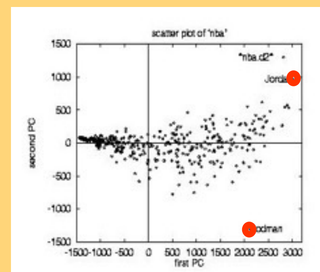




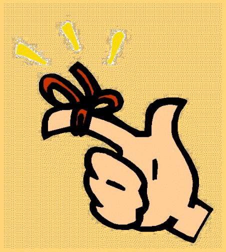


Conclusions (2/3)

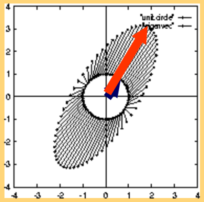
- …
- ... and can find rules (PCA; RatioRules)
- ... and do visualization



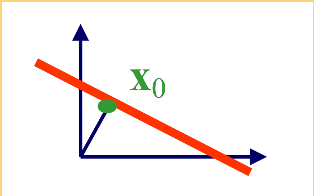
Conclusions (3/3)



- ... 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)



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.

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.

