

15-826: Multimedia (Databases) and Data Mining

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

C. Faloutsos



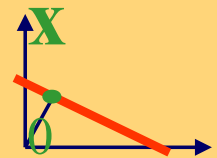
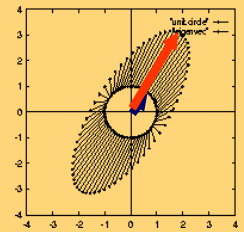
Problem

- 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 ($\leftrightarrow$ 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 $\mathbf{A}$ is a $(n \times n)$ square matrix
 $(\lambda, \mathbf{x})$ is an **eigenvalue/eigenvector** pair
of $\mathbf{A}$ if

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

CLOSELY related to singular values:

Property #1: Eigen- vs singular-values

if $\begin{bmatrix} \square & \parallel & \text{---} \end{bmatrix}$

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

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

$$\mathbf{B}^T \mathbf{B} \mathbf{v}_i = \lambda_i^2 \mathbf{v}_i$$

ie, $\mathbf{v}_1, \mathbf{v}_2, \dots$: eigenvectors of $\mathbf{A} = (\mathbf{B}^T \mathbf{B})$

Property #2

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

(if $\mathbf{A}$ is not symmetric, some eigenvalues may be complex)

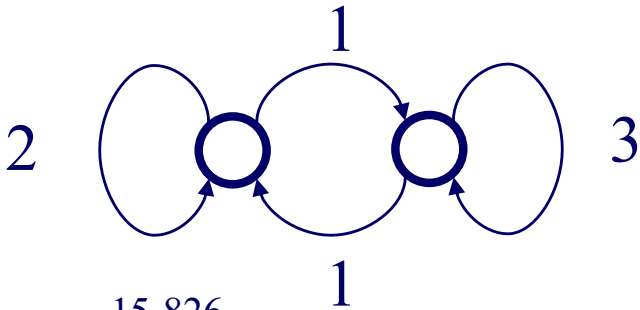
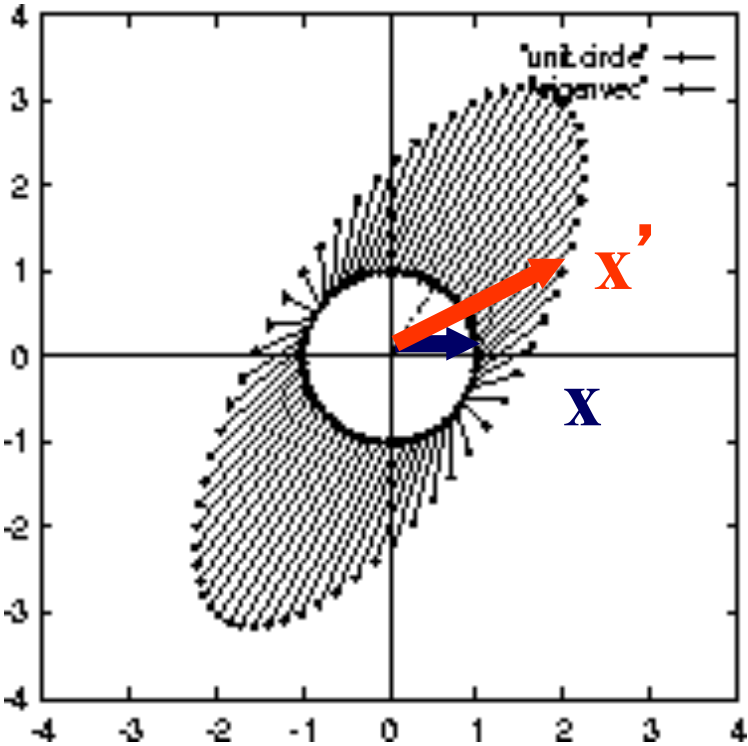
Parenthesis: intuition behind eigenvectors

- Definition
- 2 properties
- **intuition**

Intuition

- A** as vector transformation

$$\begin{matrix} \textcolor{red}{\mathbf{x}'} & & \mathbf{A} & & \mathbf{x} \\ \begin{bmatrix} 2 \\ 1 \end{bmatrix} & = & \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{matrix}$$

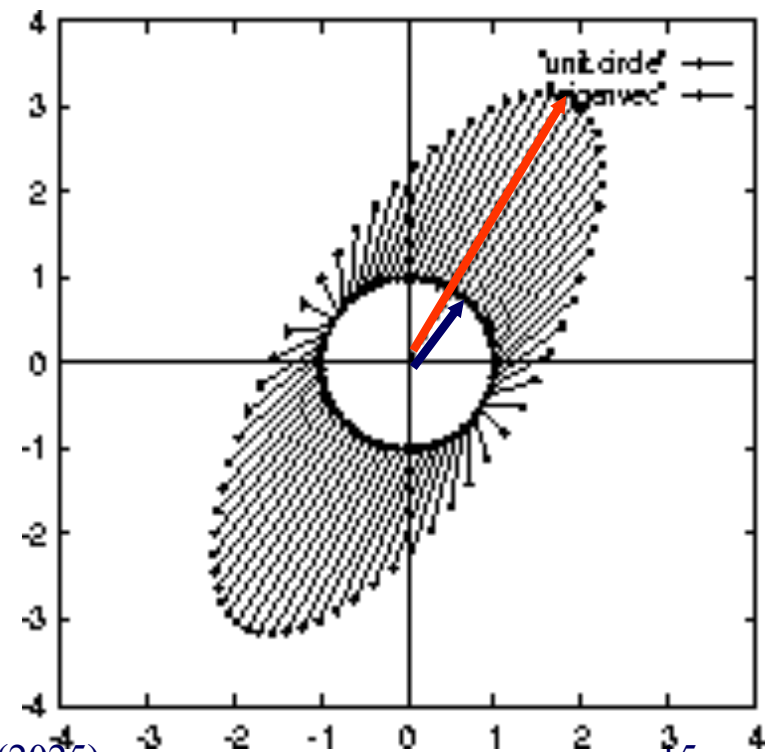


Intuition

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

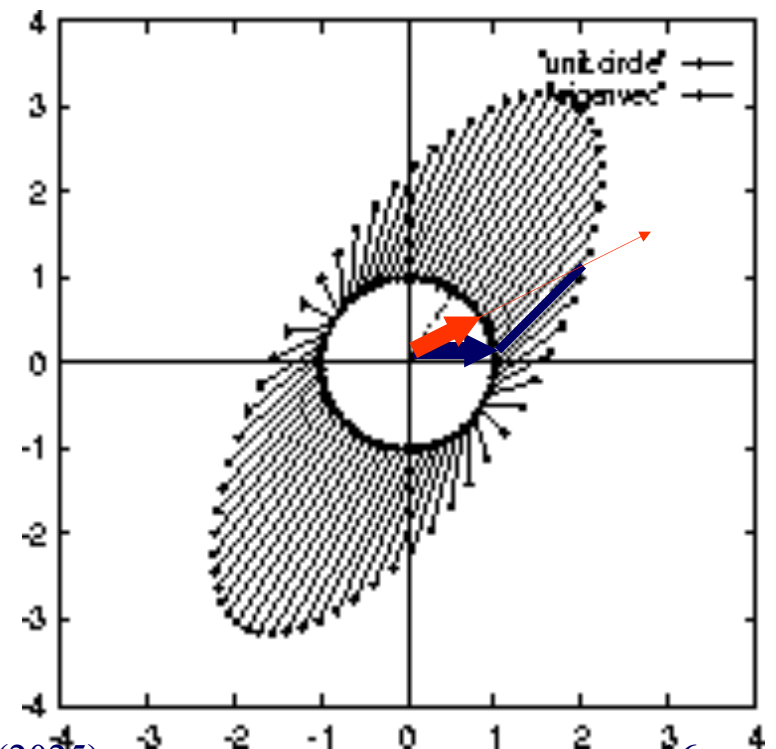
$$\lambda_1 \mathbf{v}_1 = \mathbf{A} \mathbf{v}_1$$

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



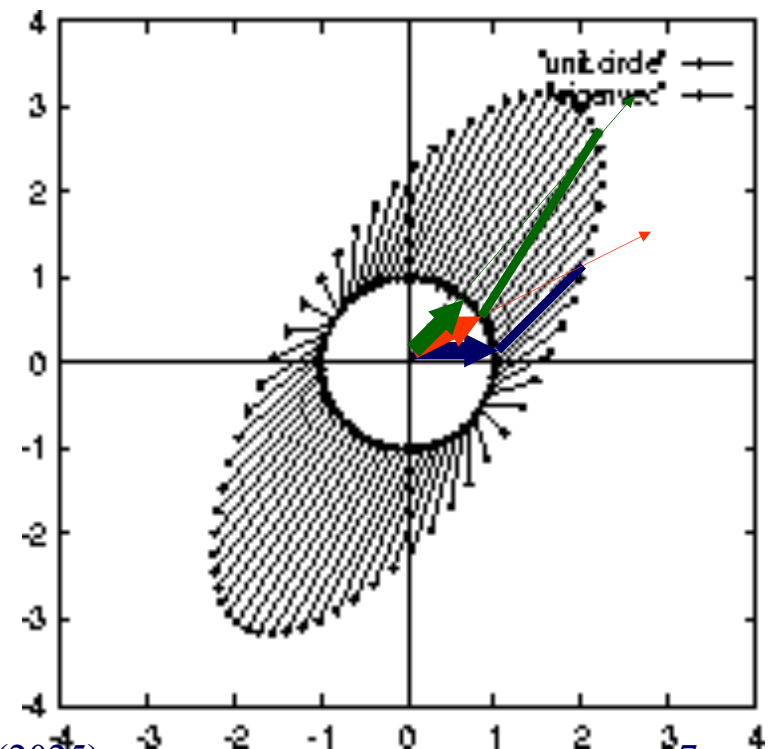
Convergence

- Usually, fast:



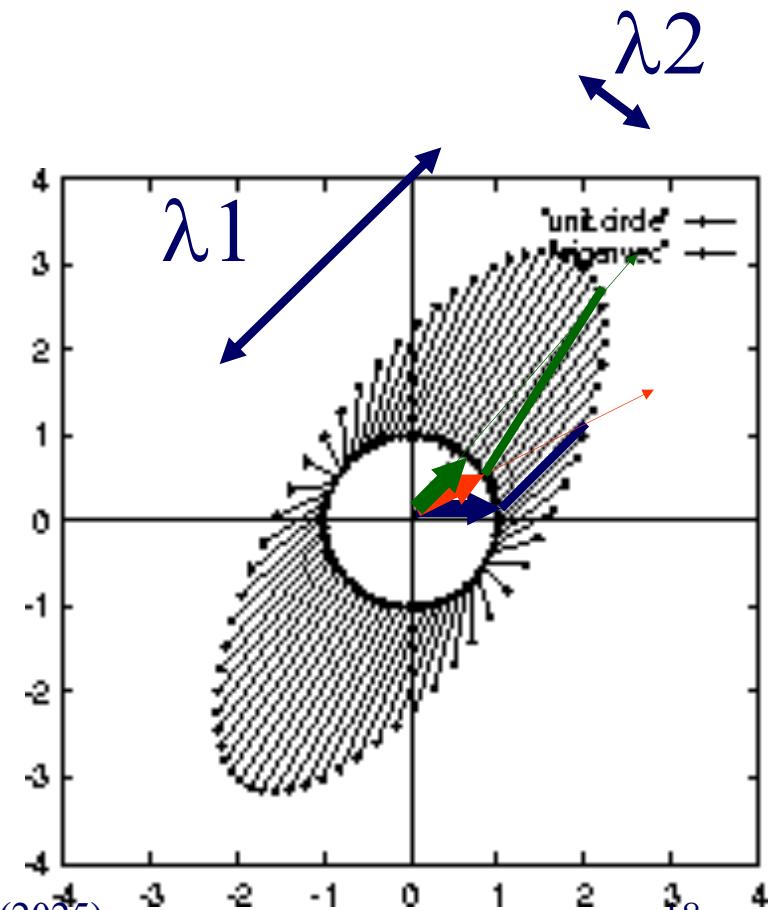
Convergence

- Usually, fast:



Convergence

- Usually, fast:
- depends on ratio
 $\lambda_1 : \lambda_2$



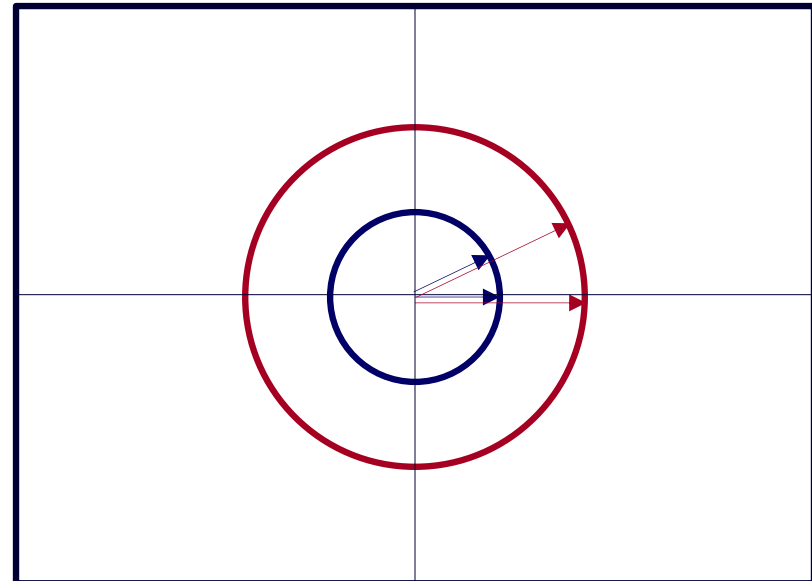
What happens if $\lambda_1 = \lambda_2$?

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

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

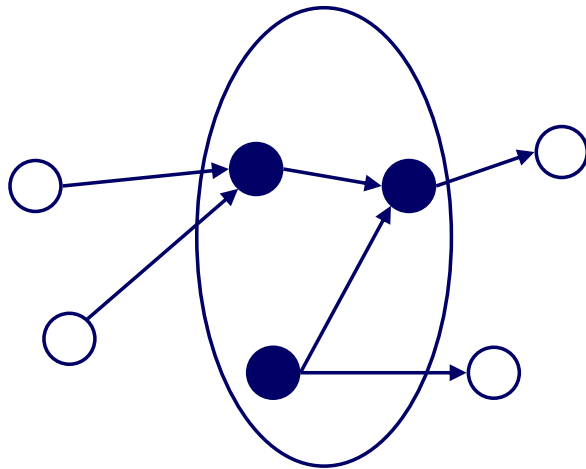
- Problem dfn: given the web and a query
- find the most 'authoritative' web pages for this query

Step 0: find all pages containing the query terms

Step 1: expand by one move forward and backward

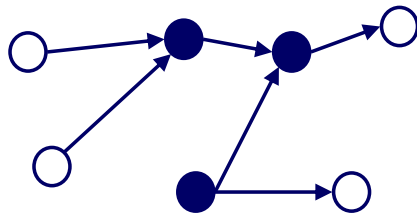
Kleinberg's algorithm

- Step 0: find nodes with query word(s)
- Step 1: expand by one move forward and backward



Kleinberg's algorithm

- on the resulting graph, give high score (= ‘authorities’) to nodes that many ‘wise’ nodes point to
- give high wisdom score (‘hubs’) to nodes that point to good ‘authorities’



Kleinberg's algorithm

Then:

$$a_i = h_k + h_l + h_m$$

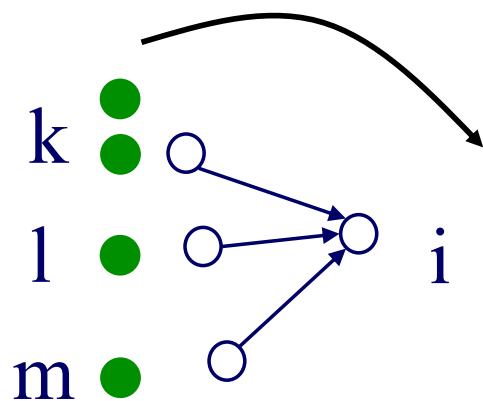
that is

$$a_i = \text{Sum } (h_j) \quad \text{over all } j \text{ that } (j,i) \text{ edge exists}$$

or

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

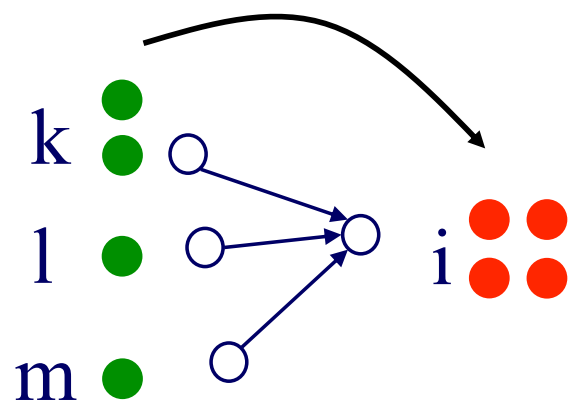
$$\begin{array}{|c} \text{orange} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{box with double arrow} \\ \hline \end{array} \begin{array}{|c} \text{green} \\ \hline \end{array}$$



Kleinberg's algorithm

Then:

$$a_i = h_k + h_l + h_m$$



that is

$$a_i = \text{Sum } (h_j) \quad \text{over all } j \text{ that } (j,i) \text{ edge exists}$$

or

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

$$\begin{array}{|c} \text{red} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{box with double arrow} \\ \hline \end{array} \begin{array}{|c} \text{green} \\ \hline \end{array}$$

Kleinberg's algorithm

symmetrically, for the 'hubness':

$$h_i = a_n + a_p + a_q$$

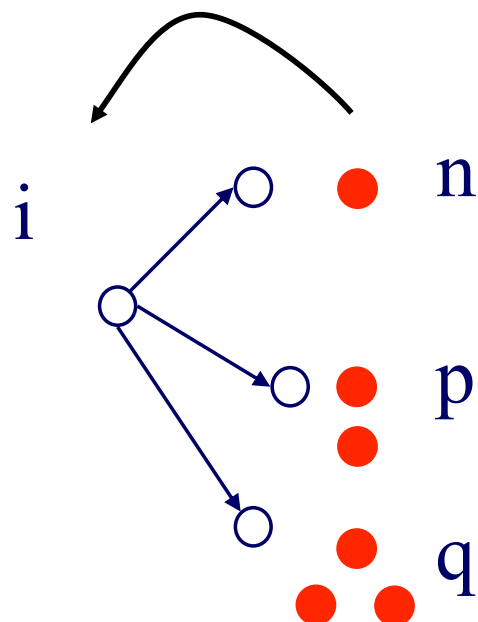
that is

$$h_i = \text{Sum } (q_j) \quad \text{over all } j \text{ that } (i,j) \text{ edge exists}$$

or

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

$$\begin{array}{|c|} \hline \\ \hline \end{array} = \begin{array}{|c|c|} \hline & \\ \hline \end{array} \begin{array}{|c|} \hline \\ \hline \end{array}$$



Kleinberg's algorithm

symmetrically, for the 'hubness':

$$h_i = a_n + a_p + a_q$$

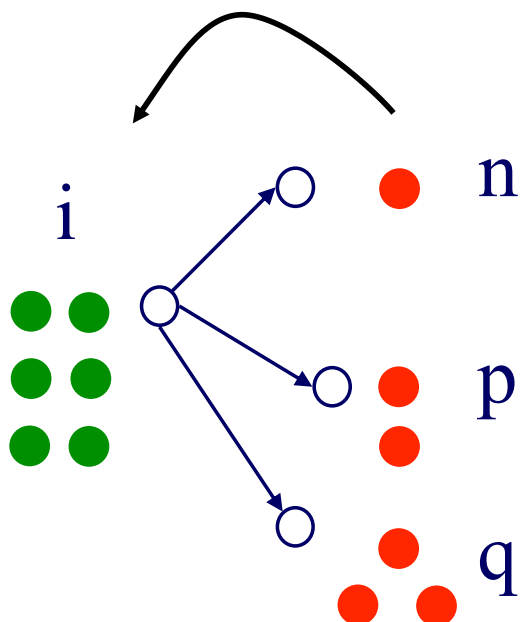
that is

$$h_i = \text{Sum } (q_j) \quad \text{over all } j \text{ that } (i,j) \text{ edge exists}$$

or

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

$$\begin{array}{|c|} \hline \\ \hline \end{array} = \begin{array}{|c|c|} \hline & \\ \hline \end{array} \begin{array}{|c|} \hline \\ \hline \end{array}$$



Kleinberg's algorithm

In conclusion, we want vectors $\mathbf{h}$ and $\mathbf{a}$ such that:

$$\begin{aligned} \mathbf{h} &= \mathbf{A} \mathbf{a} \\ \mathbf{a} &= \mathbf{A}^T \mathbf{h} \end{aligned} \quad \left| \quad \left\| \right\| = \square \right\|$$

Kleinberg's algorithm

In conclusion, we want vectors $\mathbf{h}$ and $\mathbf{a}$ such that:

$$\begin{array}{|l} \mathbf{h} = \mathbf{A} \mathbf{a} \\ \mathbf{a} = \mathbf{A}^T \mathbf{h} \end{array} \quad \|\cdot\| = \square \|\cdot\|$$

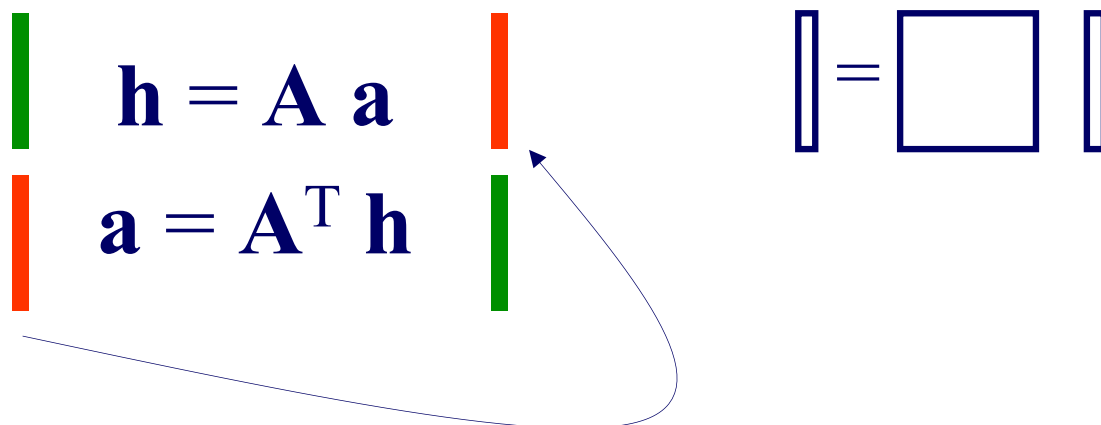
Kleinberg's algorithm

In conclusion, we want vectors $\mathbf{h}$ and $\mathbf{a}$ such that:

$$\begin{array}{|c} \mathbf{h} = \mathbf{A} \mathbf{a} \\ \mathbf{a} = \mathbf{A}^T \mathbf{h} \end{array} \quad \|\cdot\| = \square \|\cdot\|$$

Kleinberg's algorithm

In conclusion, we want vectors $\mathbf{h}$ and $\mathbf{a}$ such that:

$$\begin{array}{|c|} \hline \mathbf{h} = \mathbf{A} \mathbf{a} \\ \hline \mathbf{a} = \mathbf{A}^T \mathbf{h} \\ \hline \end{array}$$


$$\boxed{} = \boxed{} \boxed{}$$

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 $\mathbf{A}$.

Starting from random $\mathbf{a}'$ and iterating, we' ll eventually converge

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

Kleinberg' s algorithm

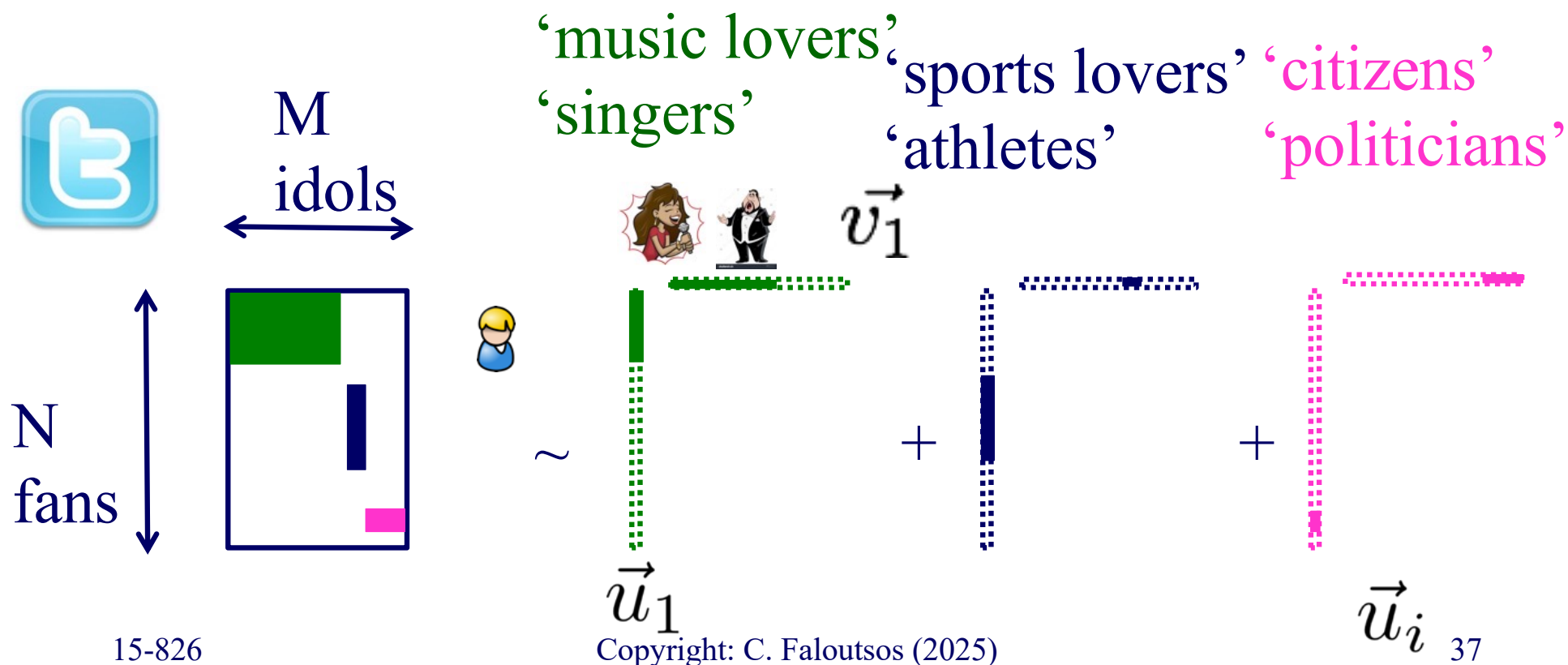
(Q: to which of all the singular-vectors?
why?)

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

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

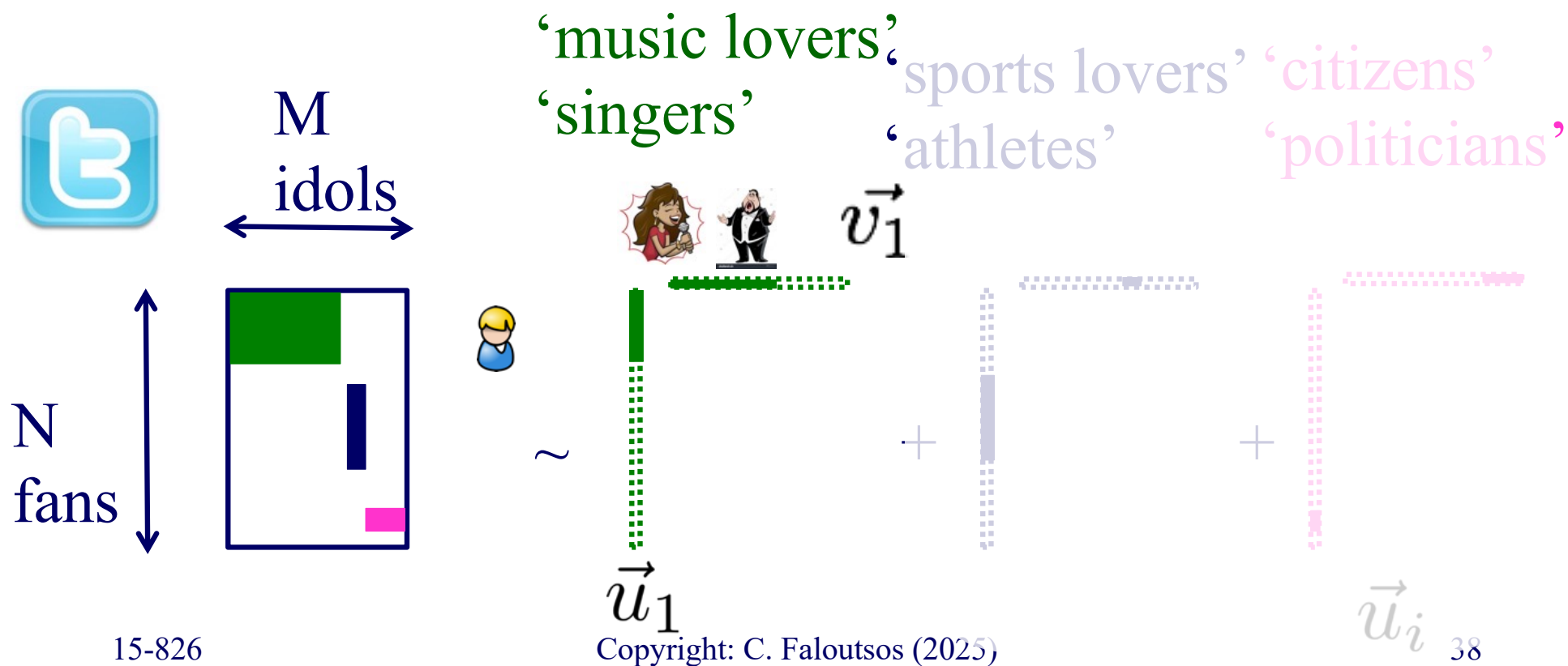
Pictorially:

- (SVD) matrix factorization: finds blocks



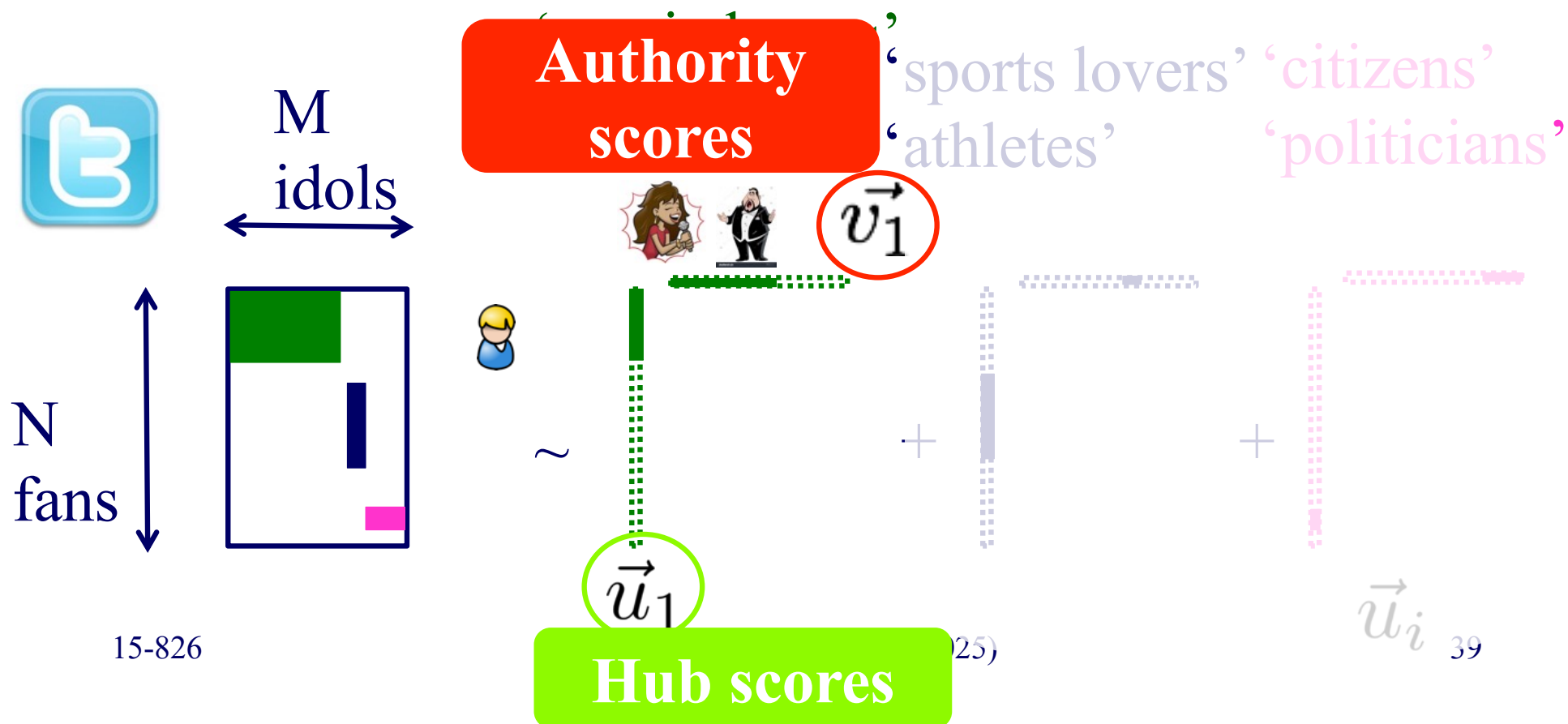
Pictorially:

- (SVD) matrix factorization: finds blocks



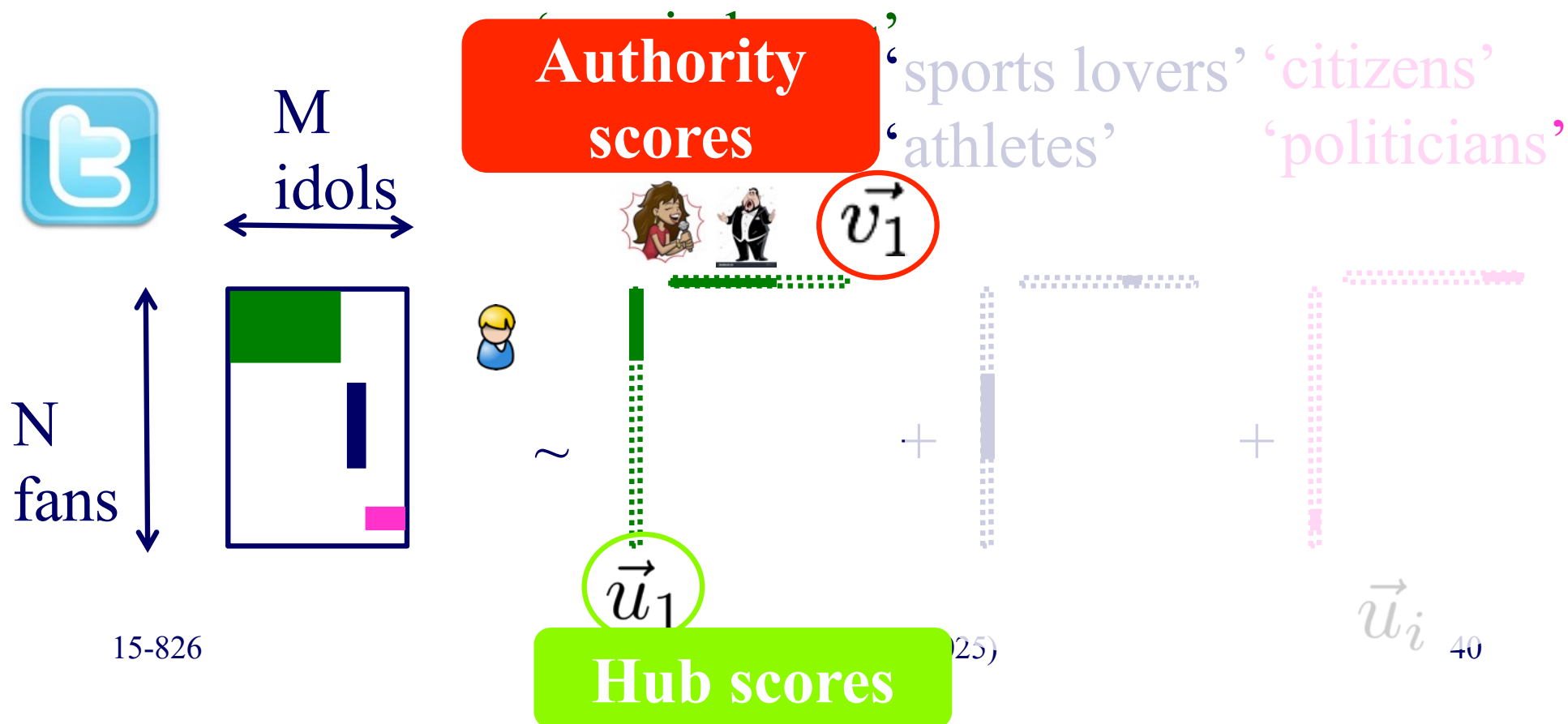
Pictorially:

- (SVD) matrix factorization: finds blocks



Pictorially:

HITS: first singular vector, ie, fixates on largest group



Proof, and intuition

Proof (sketch)

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


$$1) \quad A^T A = V \Lambda \overbrace{U^T U}^{\text{red arrow}} \Lambda V^T = V \Lambda^2 V^T$$

Proof (sketch)

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


$$1) \quad A^T A = V \Lambda \overbrace{U^T U}^{\text{red arrow}} \Lambda V^T = V \Lambda^2 V^T$$

$$2) \quad (A^T A)^k = V \Lambda^{2k} V^T \quad \text{Spectral form}$$

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

$$\approx \lambda_1^{2k} v_1 v_1^T \quad \lambda_1 > \lambda_2 > \dots$$

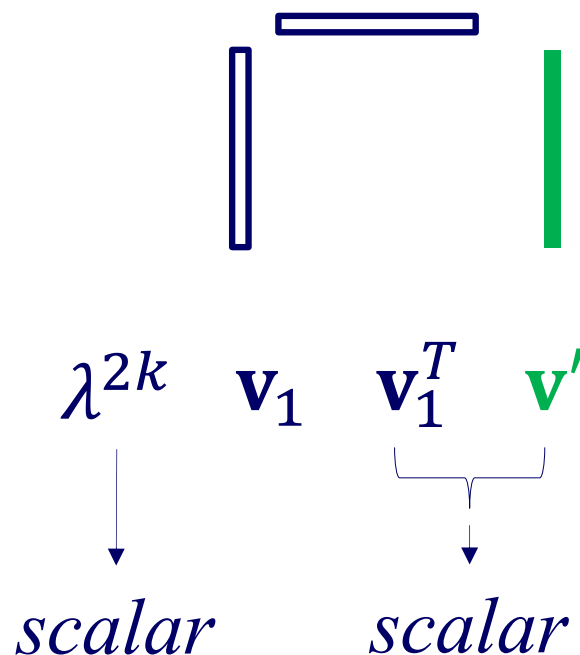
Proof (sketch)

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

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

Proof (sketch) - pictorial

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



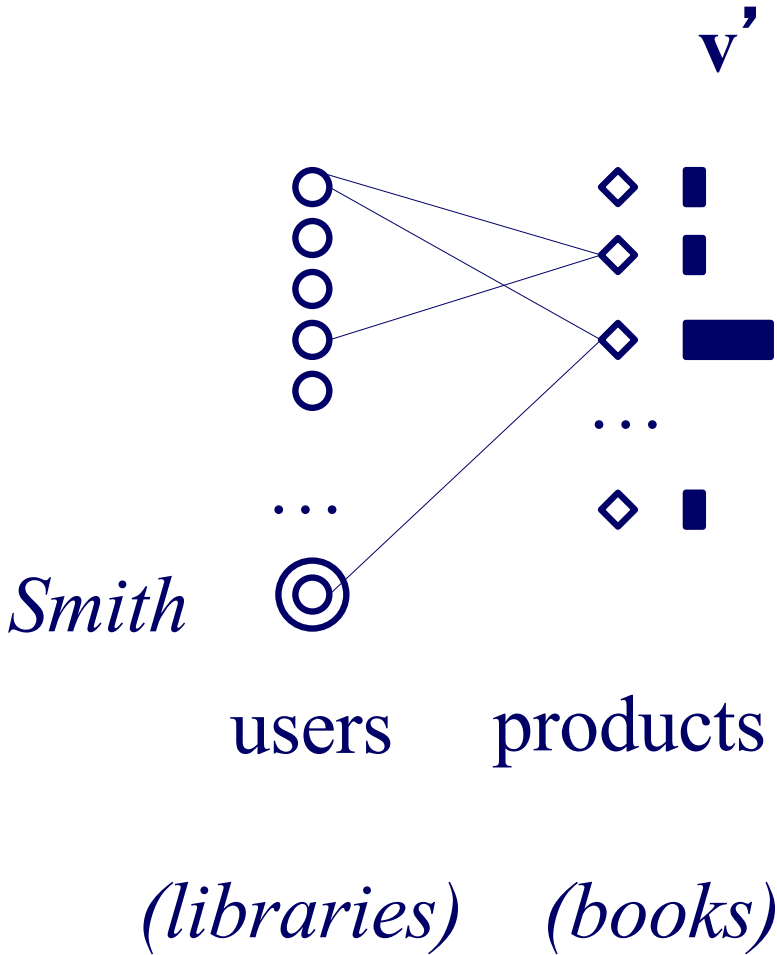
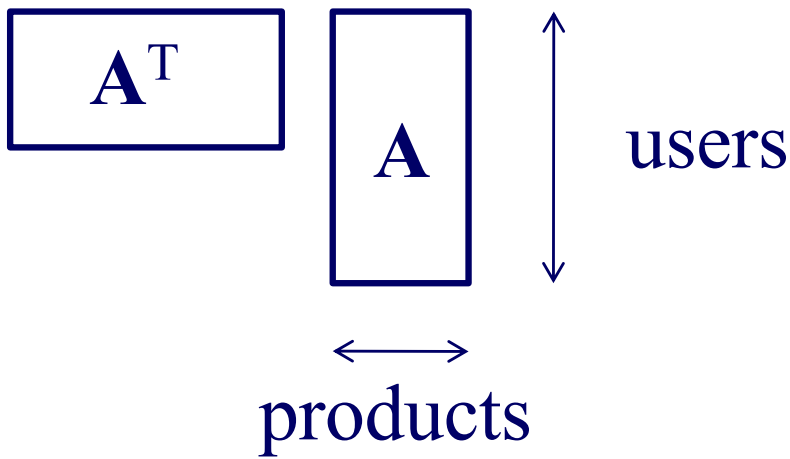
More intuition

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

k times

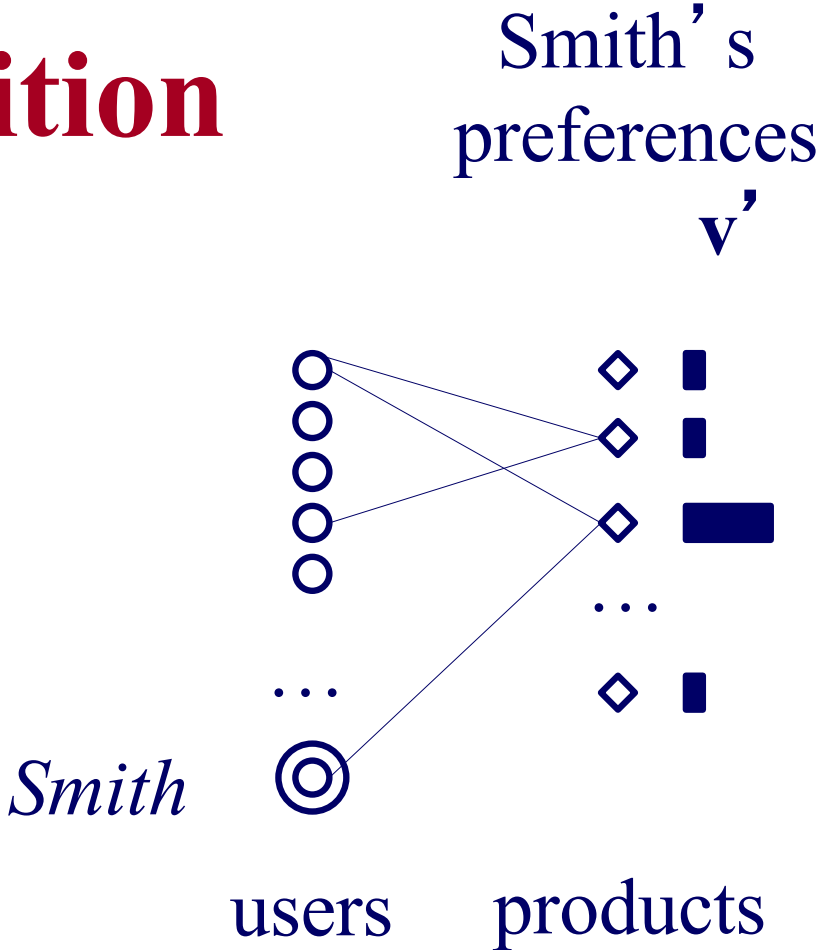
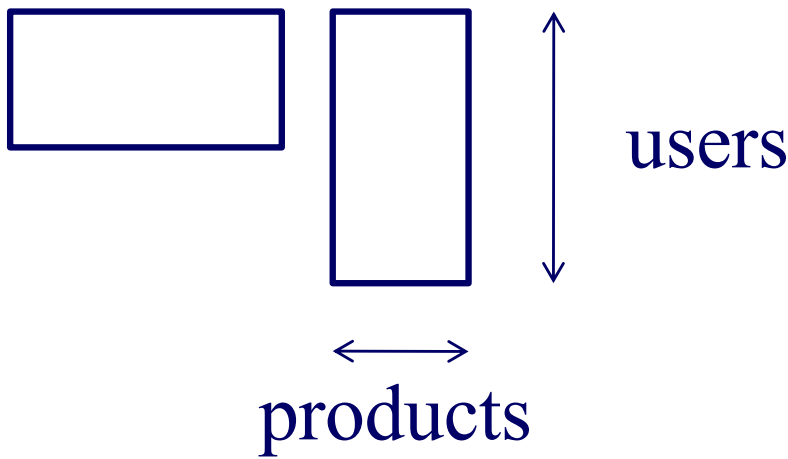
More intuition

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



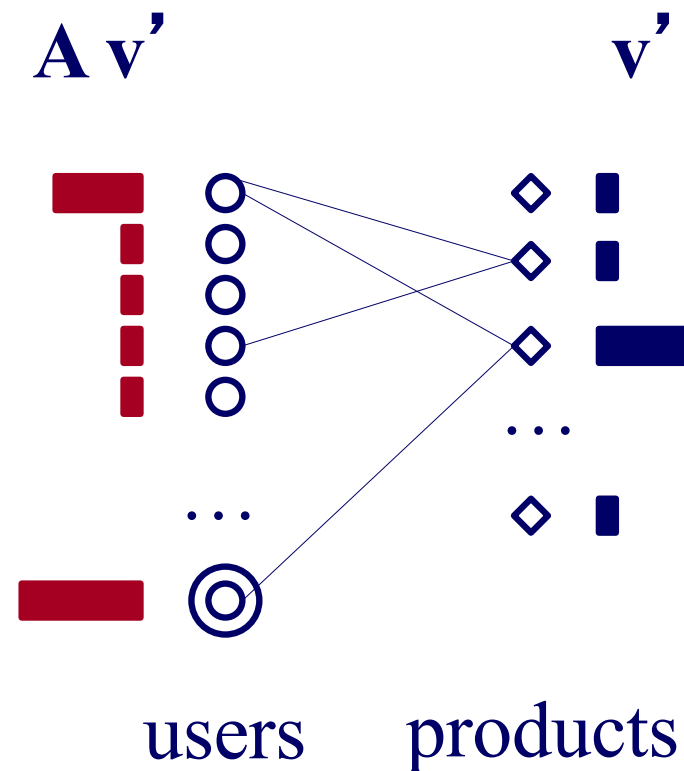
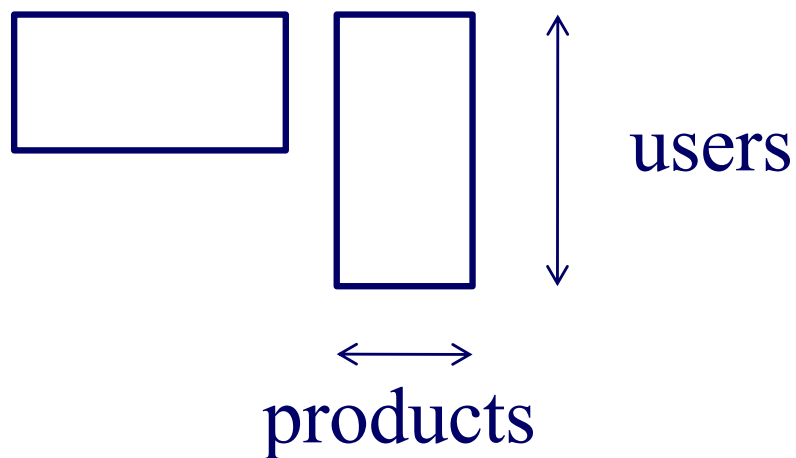
More intuition

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



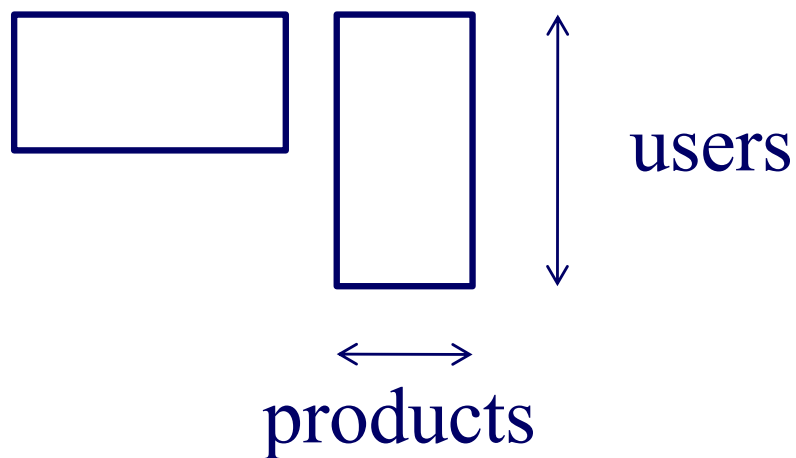
More intuition

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

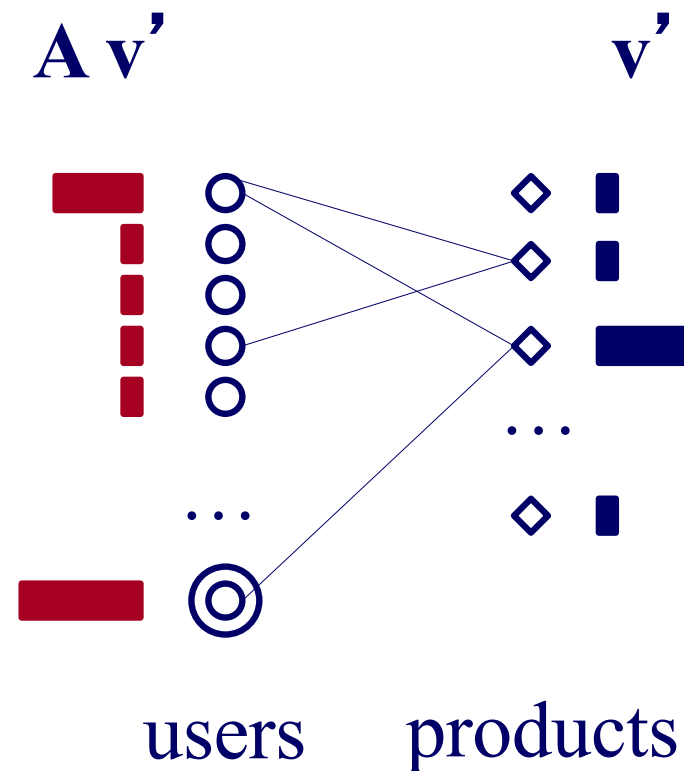


More intuition

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

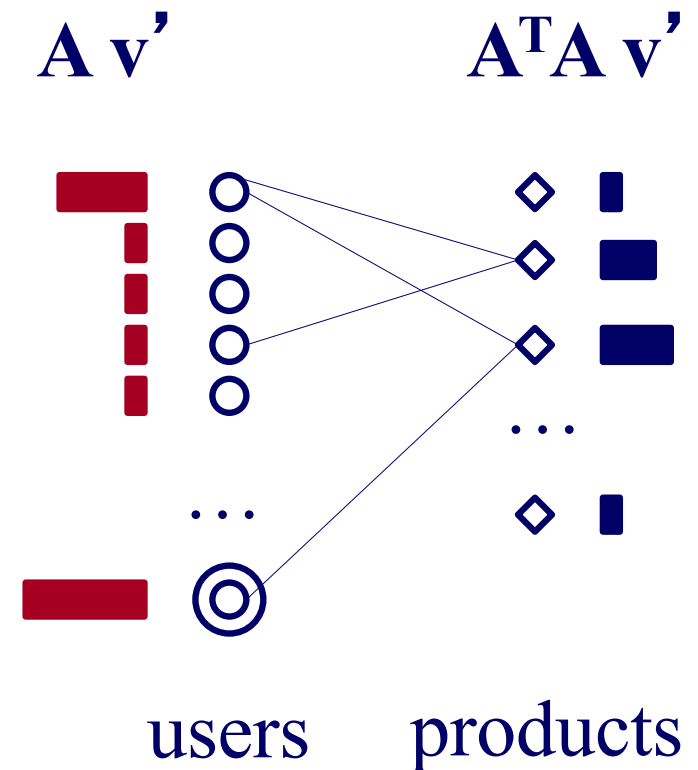
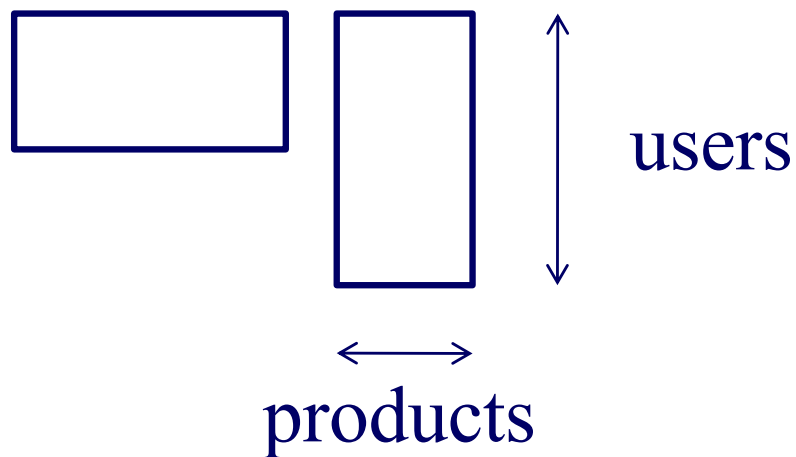


similarities
to Smith



More intuition

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

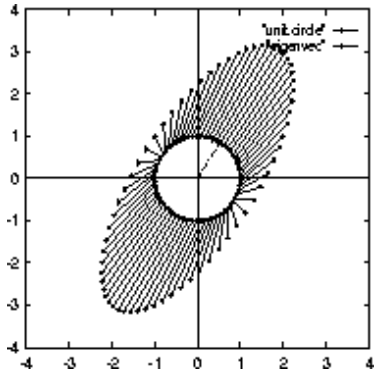


More intuition

- Intuition:
 - $(\mathbf{A}^T \mathbf{A}) \mathbf{v}'$ 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. $\rightarrow$ 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”)

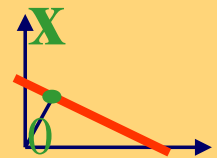
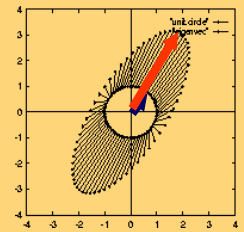
Kleinberg' s algorithm - discussion

- ‘authority’ score can be used to find ‘similar pages’ (how?)
- closely related to ‘citation analysis’, social networks / ‘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 ($\leq$ - $\geq$ 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)



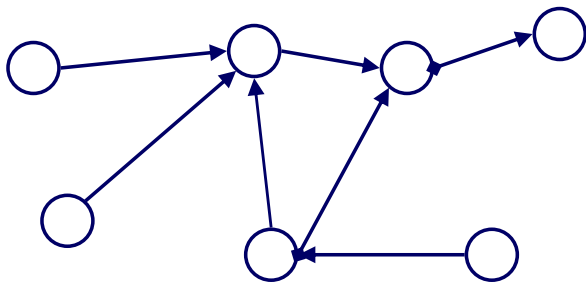
• Brin, Sergey and Lawrence Page (1998). *Anatomy of a Large-Scale Hypertextual Web Search Engine*. 7th Intl World Wide Web Conf.

Larry
Page

Sergey
Brin

Problem: PageRank

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

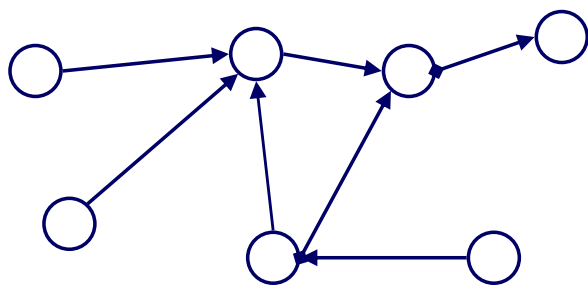


A node is important, if it is connected with important nodes (recursive, but OK!)

Problem: PageRank - solution

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

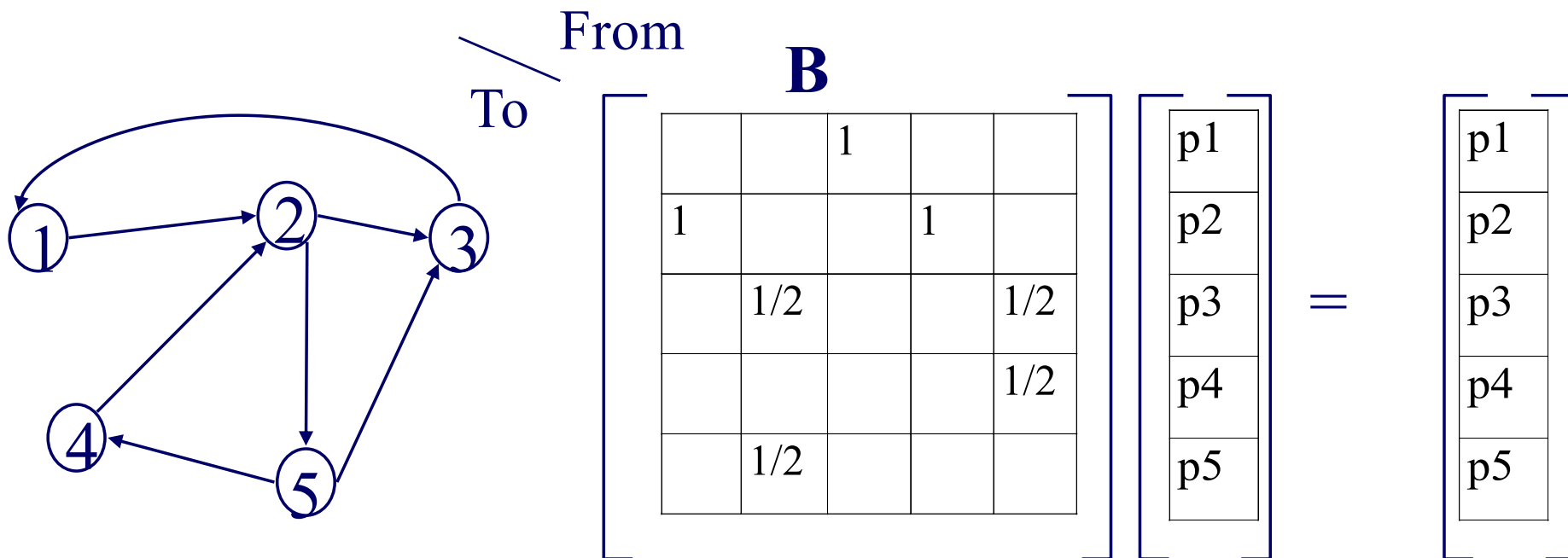
Proposed solution: Random walk; spot most 'popular' node ($\rightarrow$ steady state prob. (ssp))



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

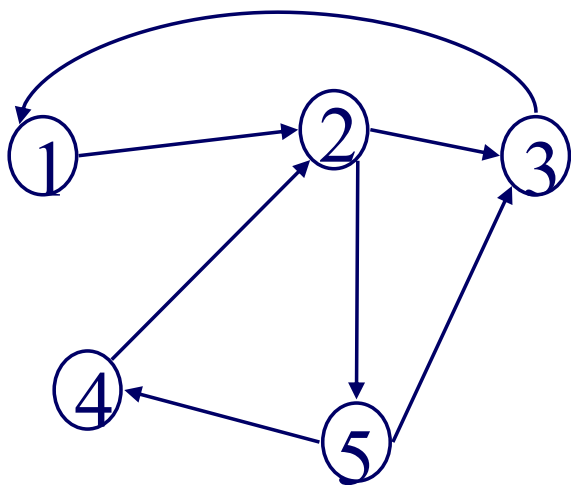
(Simplified) PageRank algorithm

- Let $\mathbf{A}$ be the adjacency matrix;
- let $\mathbf{B}$ be the transition matrix: transpose, column-normalized - then



(Simplified) PageRank algorithm

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



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

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

p1
p2
p3
p4
p5

$$=$$

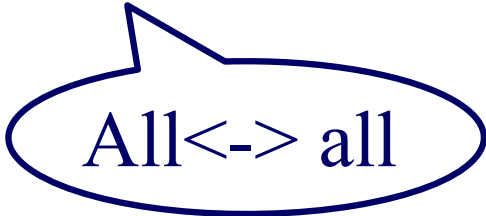
p1
p2
p3
p4
p5

(Simplified) PageRank algorithm

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

(Simplified) PageRank algorithm

- $\mathbf{B} \mathbf{p} = 1 * \mathbf{p}$
- thus, $\mathbf{p}$ is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a $\mathbf{p}$ exist?
 - $\mathbf{p}$ exists if $\mathbf{B}$ is $n \times n$, 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

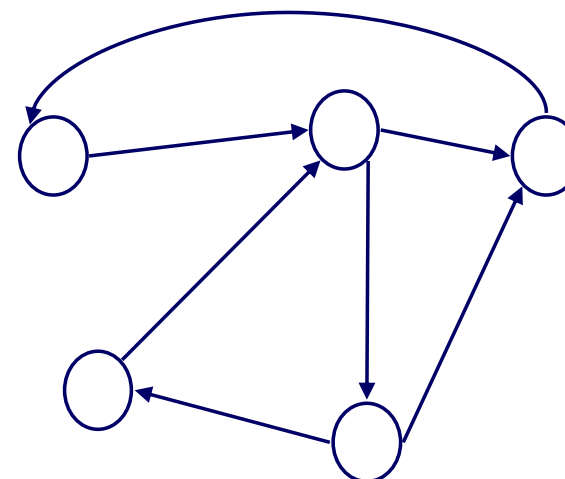
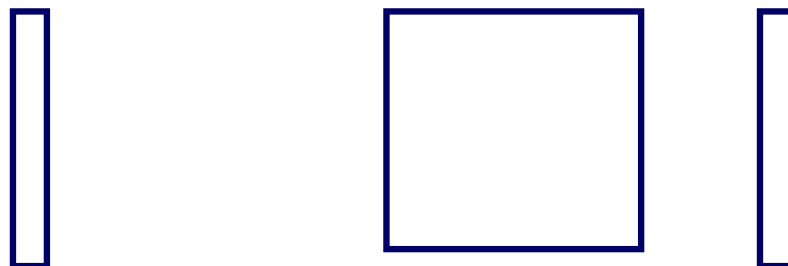
Full Algorithm



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

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

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



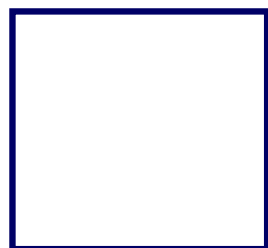
Full Algorithm



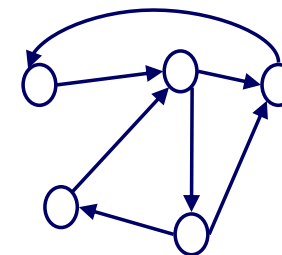
- With probability $1-c$, fly-out to a random node
- Then, we have

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

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



$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$



Alternative notation – eigenvector viewpoint

M Modified transition matrix

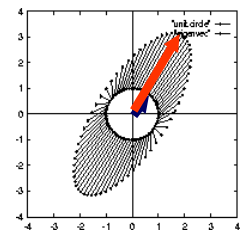
$$\mathbf{M} = c \mathbf{B} + (1-c)/n \mathbf{1} \mathbf{1}^T$$

Then



CLIQUE

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



That is: the steady state probabilities =

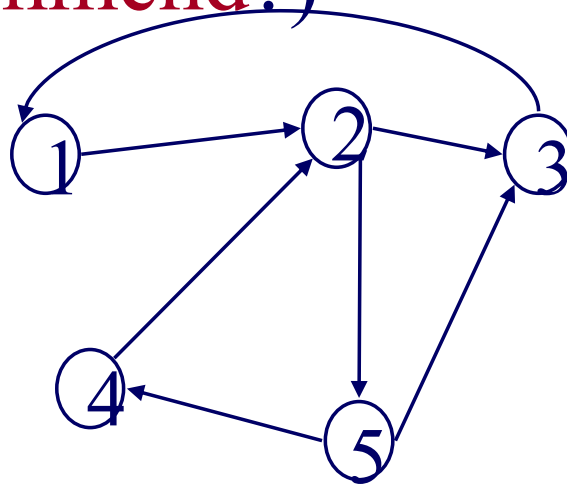
PageRank scores form the *first eigenvector* of the ‘modified transition matrix’

Personalized P.R.

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

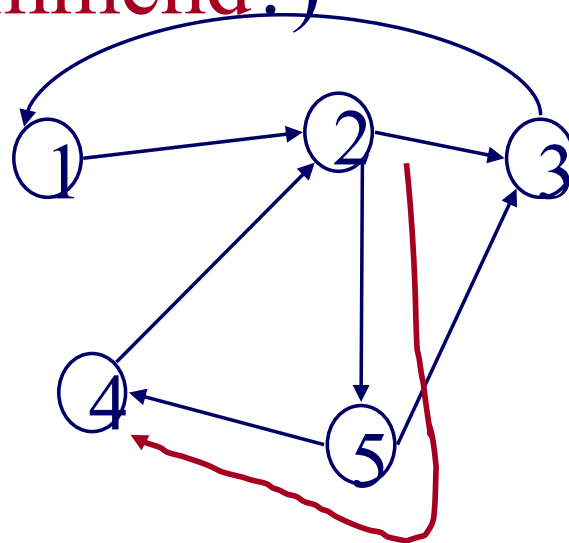
Extension: Personalized P.R.

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



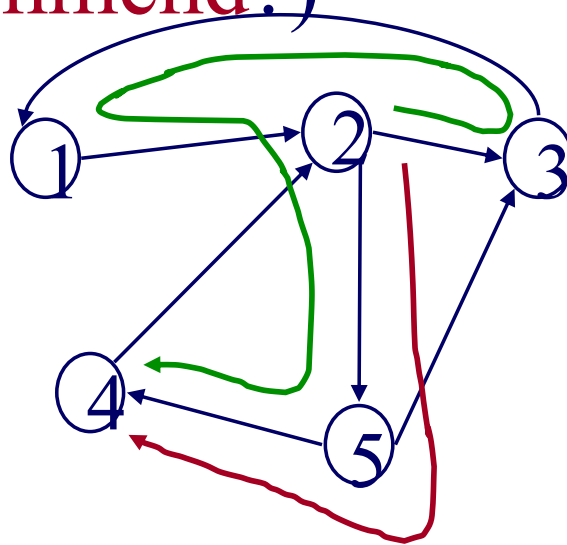
Extension: Personalized P.R.

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



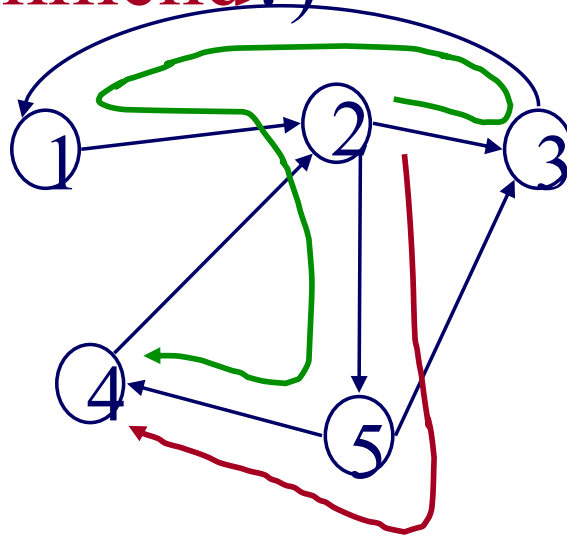
Extension: Personalized P.R.

- 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 ($A \rightarrow B$) if

- Many
 - Short
 - Heavy
- paths $A \rightarrow B$

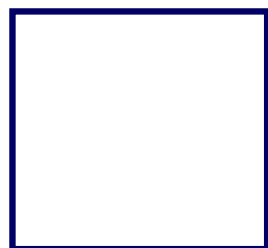
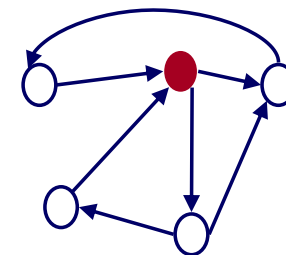
Extension: Personalized P.R.

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

- Then, we have

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

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

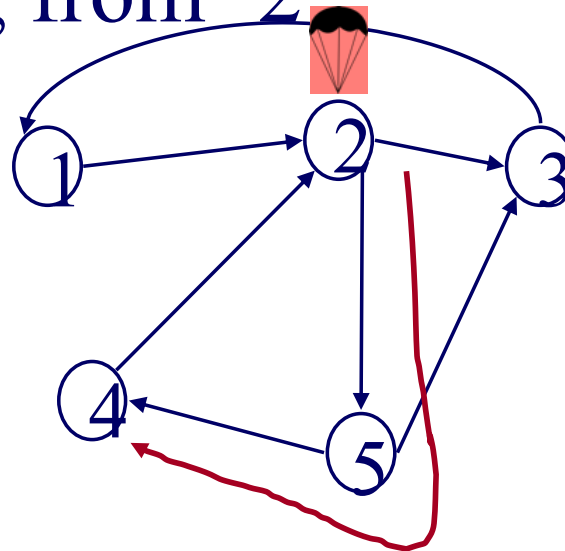


$$\vec{e} \begin{bmatrix} \textcircled{\times} \\ 1 \\ \vdots \\ \textcircled{\times} \end{bmatrix}$$

Extension: Personalized P.R.

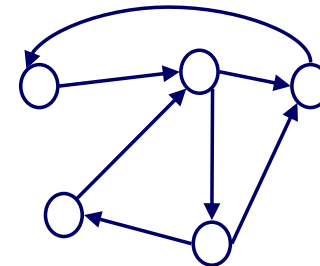


- How close is '4' to '2'?
- A: compute Personalized P.R. of '4', restarting from '2'



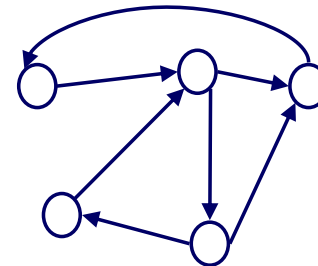
Extension: Personalized P.R.

- 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.

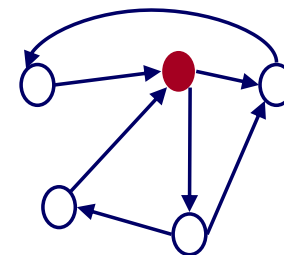
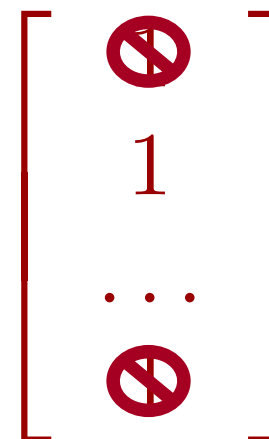
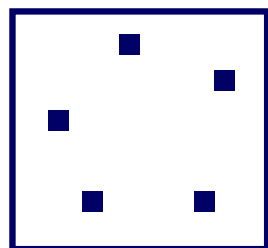
- How close is '4' to '2'?
- A: compute Personalized P.R. of '4', restarting from '2'
- How to compute it quickly?
- A: 'Pixie' algorithm



Extension: Personalized P.R.

- Q: Faster computation than:

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



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

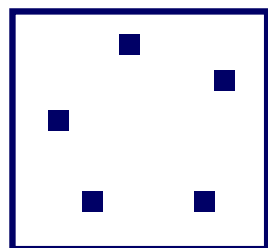
- Q: Faster computation than:

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

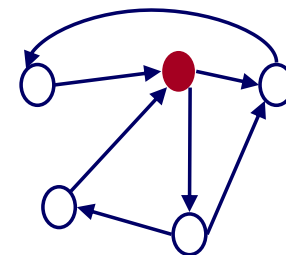
- A: **simulate** a few R.W.

- keep visit counts C_i

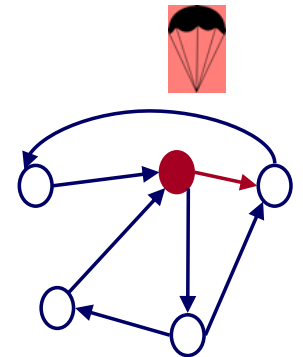
- fast and nimble



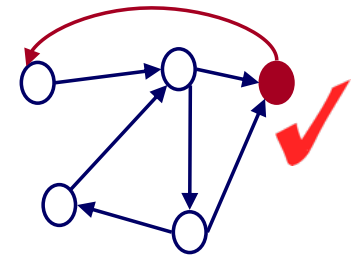
$$\begin{bmatrix} \text{no} \\ 1 \\ \dots \\ \text{no} \end{bmatrix}$$



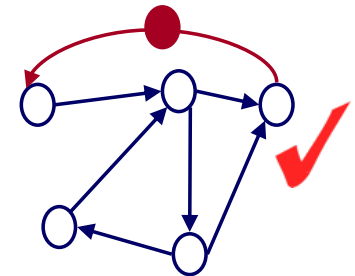
Personalized PageRank algorithm



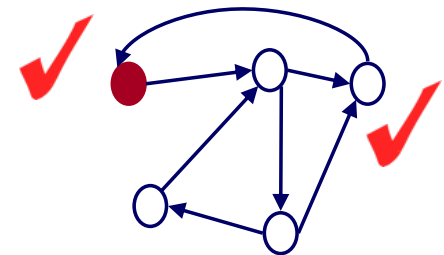
Personalized PageRank algorithm



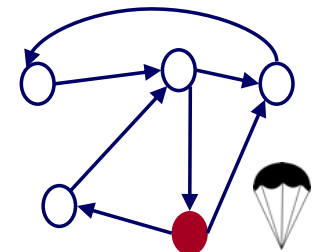
Personalized PageRank algorithm



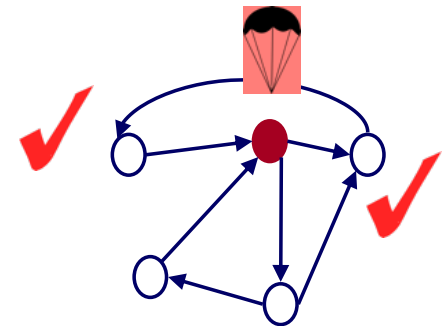
Personalized PageRank algorithm



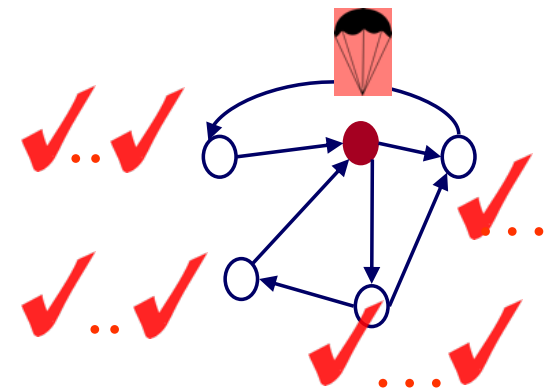
Personalized PageRank algorithm



Personalized PageRank algorithm



Personalized PageRank algorithm



Kleinberg/PageRank - conclusions

SVD helps in graph analysis:

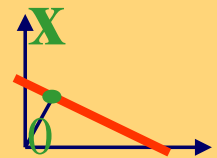
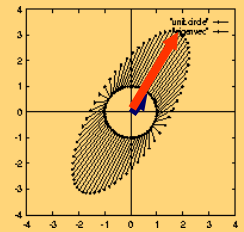
hub/authority scores: strongest left- and right-
singular-vectors of the adjacency matrix

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



Conclusions

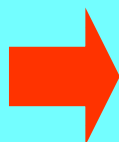
- Q1: most important node(s) in a graph?
 - ✓ A1.1: HITS (= SVD)
 - ✓ A1.2: PageRank (= fixed point)
- Q2: how to solve *any* linear system (over, under-, exactly-specified)?
 - A2: SVD ($\leftrightarrow$ Moore-Penrose pseudo-inverse)





SVD - detailed outline

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



Least obvious properties

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

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

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

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

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

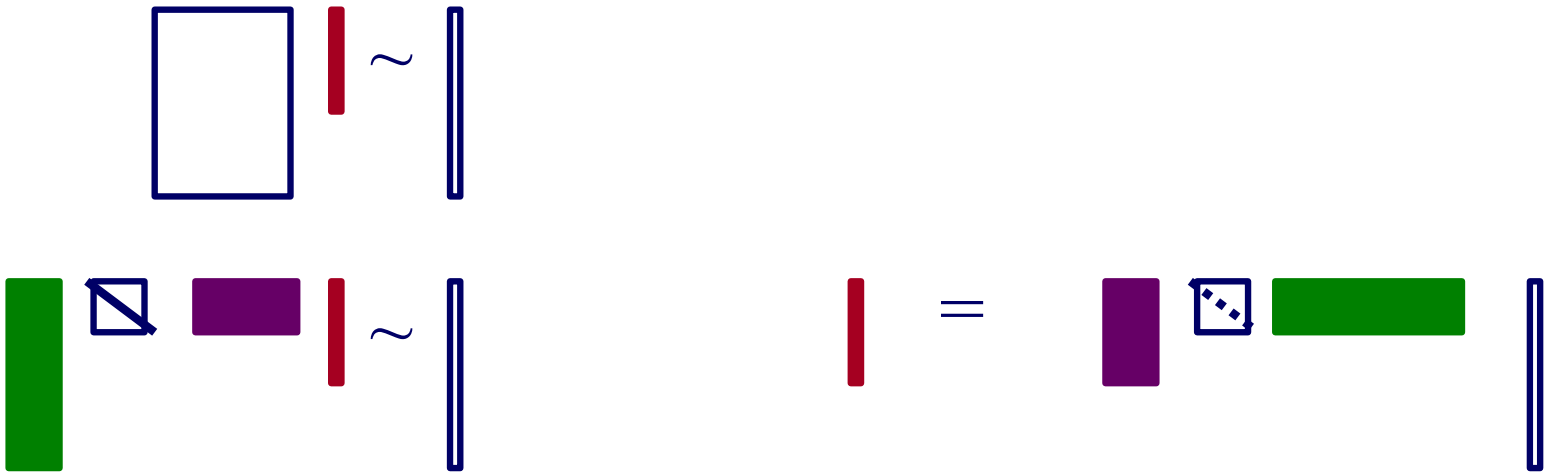
(see Num. Recipes, p. 62)

Least obvious properties

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

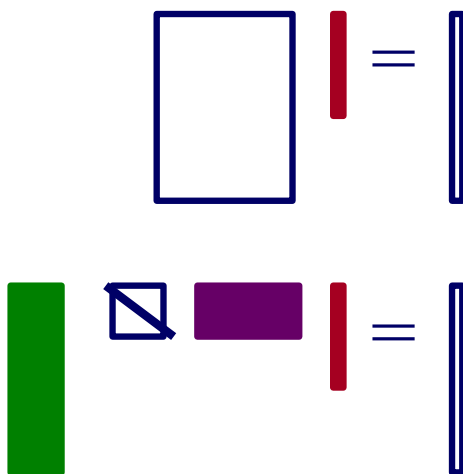
$$C(1): \mathbf{A}_{[n \times m]} \mathbf{x}_{[m \times 1]} = \mathbf{b}_{[n \times 1]}$$

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



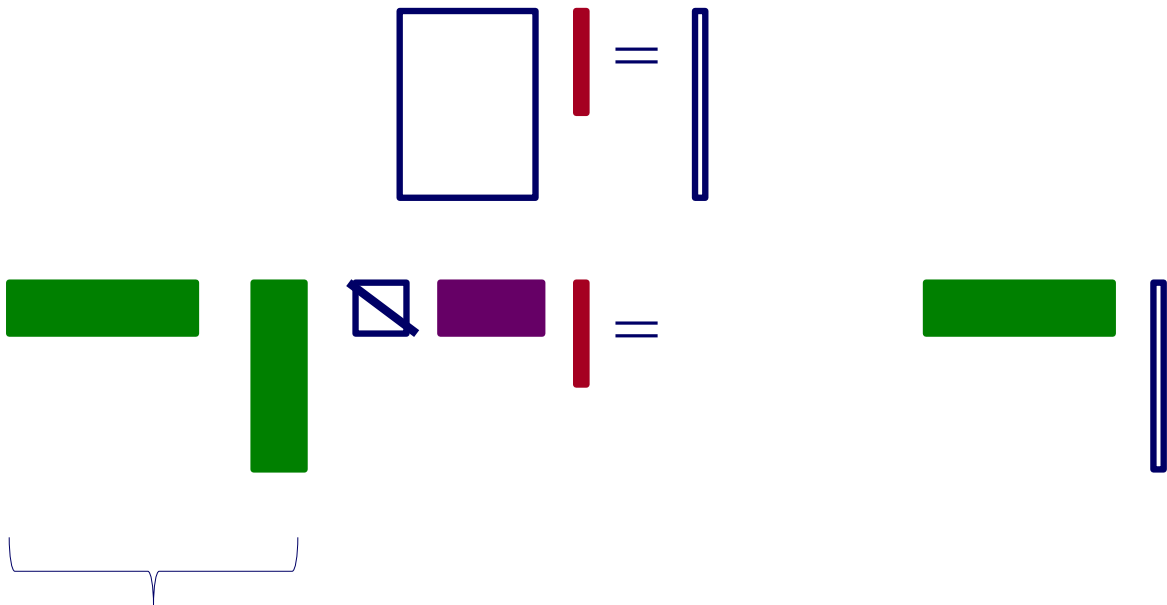
NOT proof

Slowly:



NOT proof

Slowly:

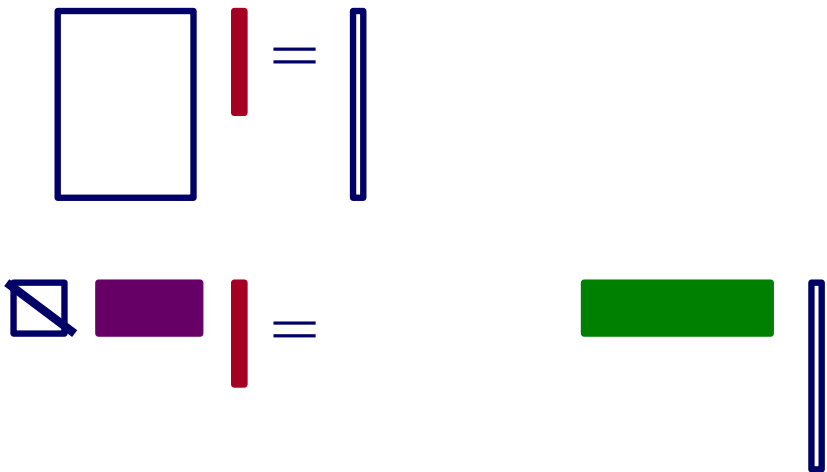


Identity

U: column-orthonormal

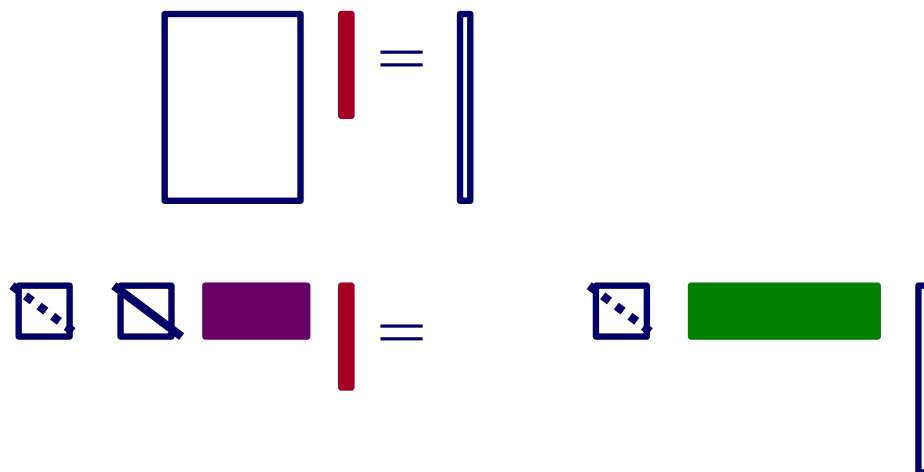
NOT proof

Slowly:



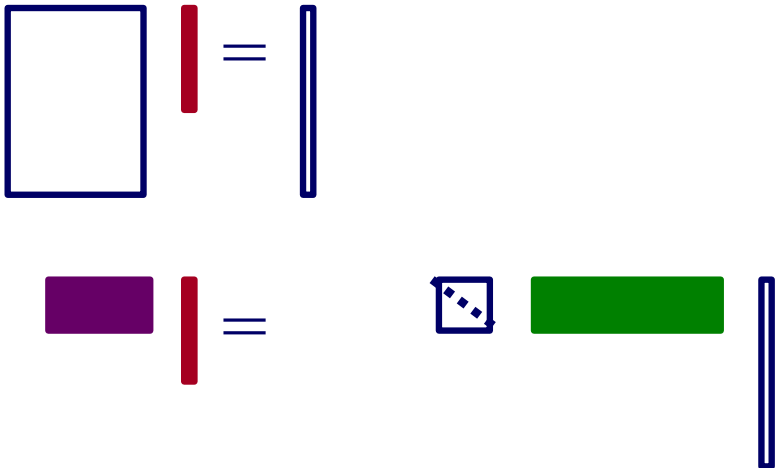
NOT proof

Slowly:



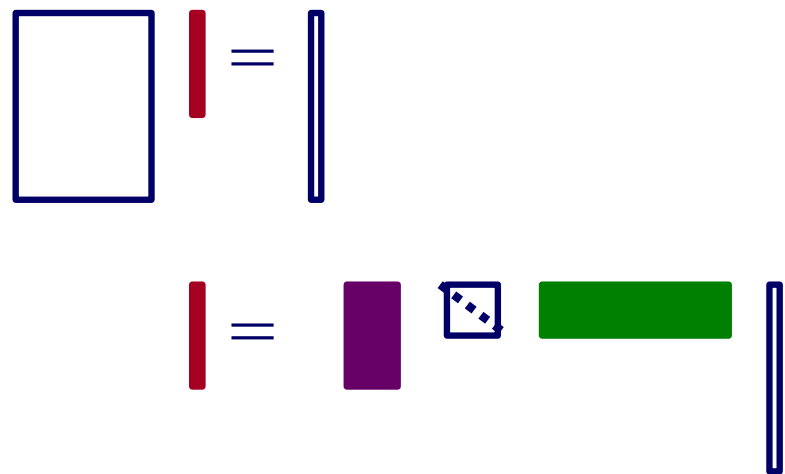
NOT proof

Slowly:



NOT proof

Slowly:



NOT proof

Slowly:

$$\begin{array}{c}
 \boxed{} \text{ (blue outline)} \quad \text{red bar} = \text{blue double bar} \\
 \\
 \text{red bar} = \text{purple bar} \quad \boxed{\text{dotted}} \quad \text{green bar} \quad \text{blue double bar} \\
 \\
 \mathbf{x} \quad \mathbf{V} \quad \mathbf{\Lambda}^{-1} \quad \mathbf{U}^T \quad \mathbf{b}
 \end{array}$$

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 $\mathbf{A}$ and $\mathbf{b}$;

$$\begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} \\ \end{bmatrix}$$

$$\begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} \\ \end{bmatrix}$$

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

Recursive Least Squares (RLS)

Can add to $\mathbf{A}$ and $\mathbf{b}$; and incrementally update $\mathbf{x}$

The diagram illustrates the incremental update of the Recursive Least Squares (RLS) algorithm. It shows the relationship between the current state and the updated state when adding a new data point.

Top equation: A large empty square (representing the current covariance matrix $\mathbf{A}$) with a small blue rectangle at its bottom (representing the current estimate $\mathbf{x}$) is followed by a red vertical bar (representing the current residual $\mathbf{r}$). This is set equal to a tall, thin blue vertical bar (representing the updated covariance matrix $\mathbf{A}^+$) with a small blue rectangle at its bottom (representing the updated estimate $\mathbf{x}^+$).

Bottom equation: A red vertical bar (representing the current residual $\mathbf{r}$) is set equal to a purple rectangle (representing the gain matrix $\mathbf{V}$) multiplied by a small square with a dashed diagonal (representing the innovation covariance matrix $\mathbf{\Lambda}^{-1}$) multiplied by a green rectangle (representing the transpose of the input vector $\mathbf{U}^T$). This result is then added to a tall, thin blue vertical bar (representing the current residual $\mathbf{r}$) to produce the updated residual $\mathbf{b}$.

Below the equations, the variables are labeled: $\mathbf{x}$ (red), $\mathbf{V}$ (purple), $\mathbf{\Lambda}^{-1}$ (blue), $\mathbf{U}^T$ (green), and $\mathbf{b}$ (blue).



Drills:

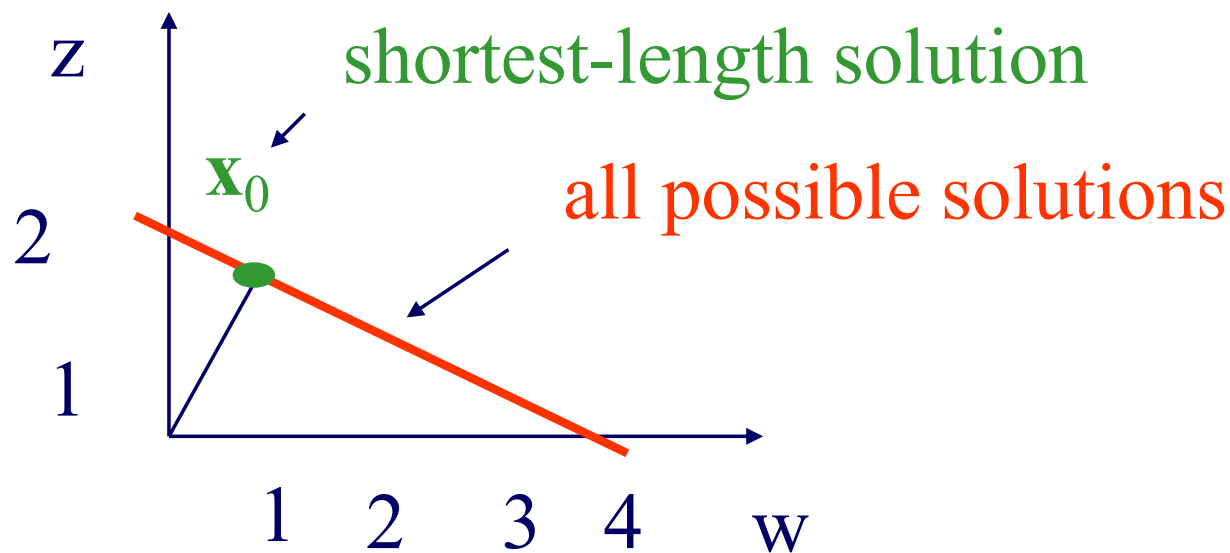
Least obvious properties

Illustration: under-specified, eg

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} w & z \end{bmatrix}^T = 4 \quad (\text{ie, } 1w + 2z = 4)$$

$\mathbf{A} = ??$

$\mathbf{b} = ??$



Verify formula:

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

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

$$\mathbf{U} = ??$$

$$\mathbf{\Lambda} = ??$$

$$\mathbf{V} = ??$$

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

Verify formula:

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

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

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

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

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

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

Verify formula:

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

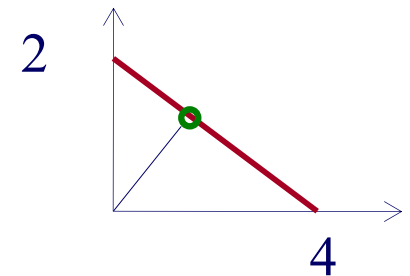
$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^T$$

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

$$\mathbf{\Lambda} = [\sqrt{5}]$$

$$\mathbf{V} = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}^T$$

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b} = \begin{bmatrix} 1/5 & 2/5 \end{bmatrix}^T [4] \\ &= [4/5 \ 8/5]^T : w = 4/5, z = 8/5 \end{aligned}$$

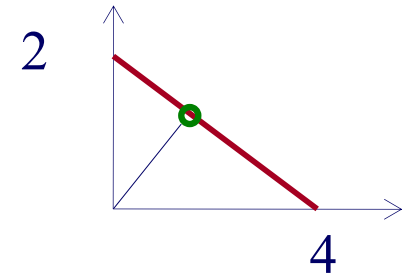


Verify formula:

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

(a) A solution to $1 \cdot w + 2 \cdot z = 4$ and

(b) Minimal (wrt Euclidean norm)



Verify formula:

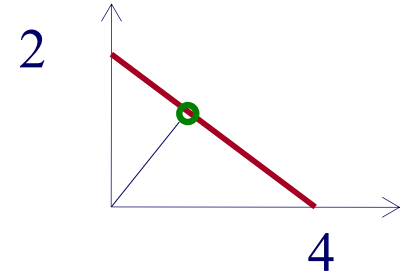
Show that $w = 4/5, z = 8/5$ is

(a) A solution to $1 \cdot w + 2 \cdot z = 4$ and

A: easy

(b) Minimal (wrt Euclidean norm)

A: $[4/5 \ 8/5]$ is perpendicular to $[2 \ -1]$

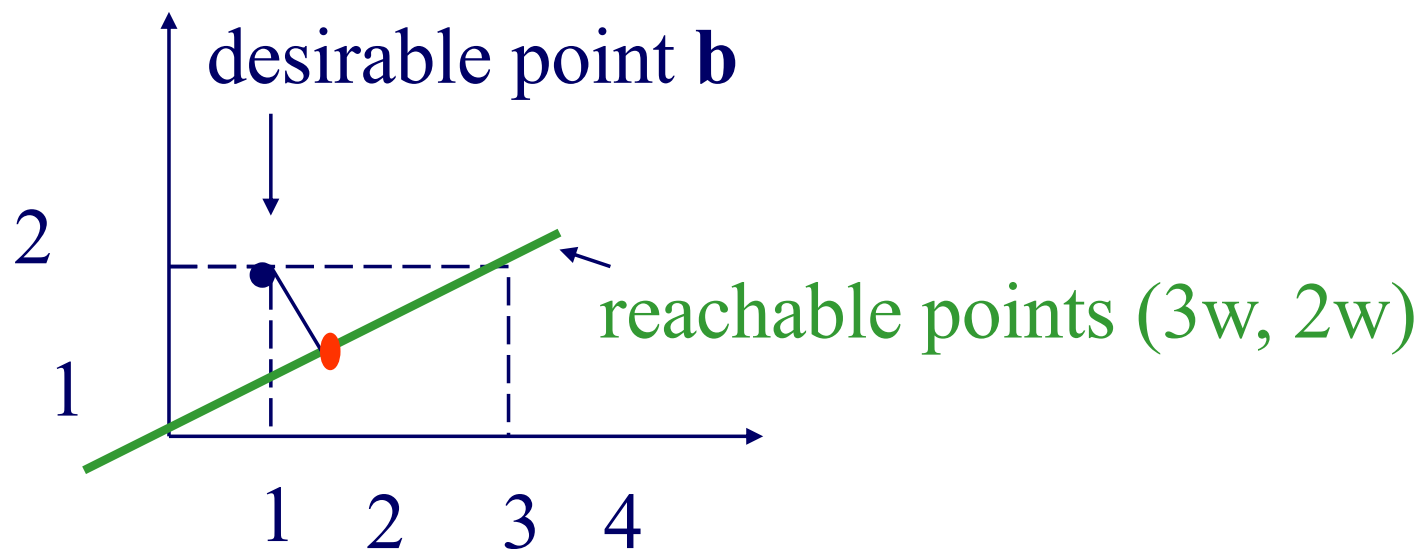


Least obvious properties – cont' d

Illustration: over-specified, eg

$$\begin{bmatrix} 3 & 2 \end{bmatrix}^T [w] = \begin{bmatrix} 1 & 2 \end{bmatrix}^T \quad (\text{ie, } 3w = 1; 2w = 2)$$

$\mathbf{A} = ??$
 $\mathbf{b} = ??$



Verify formula:

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

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

$$\mathbf{U} = ??$$

$$\mathbf{\Lambda} = ??$$

$$\mathbf{V} = ??$$

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

Verify formula:

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

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

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

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

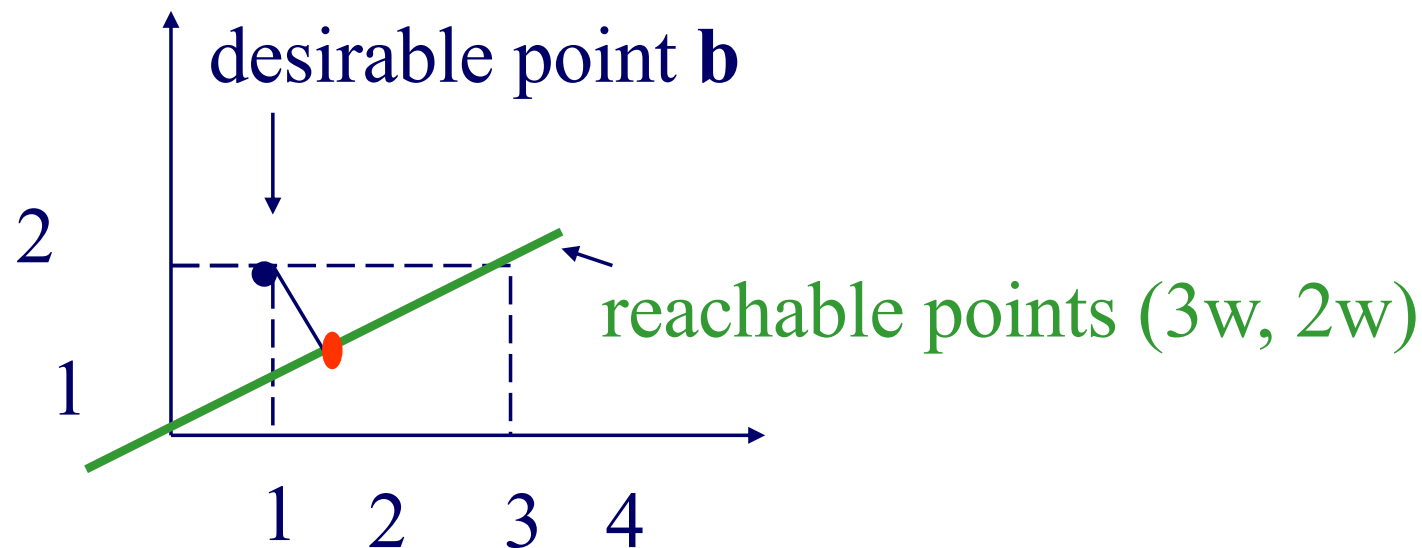
$$\mathbf{V} = [1]$$

$$\mathbf{x}_0 = \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b} = [7/13]$$

Verify formula:

$$\begin{bmatrix} 3 & 2 \end{bmatrix}^T \begin{bmatrix} 7/13 \end{bmatrix} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$$

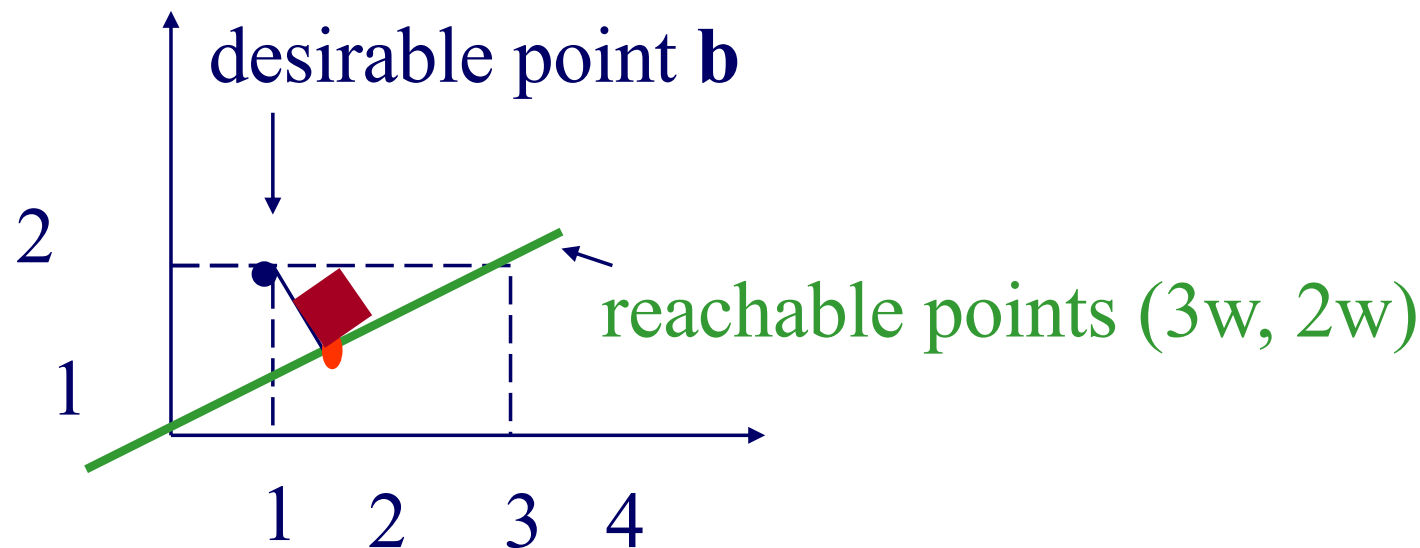
$$\begin{bmatrix} 21/13 & 14/13 \end{bmatrix}^T \rightarrow \text{'red point'}$$



Verify formula:

$$\begin{bmatrix} 3 & 2 \end{bmatrix}^T \begin{bmatrix} 7/13 \end{bmatrix} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$$

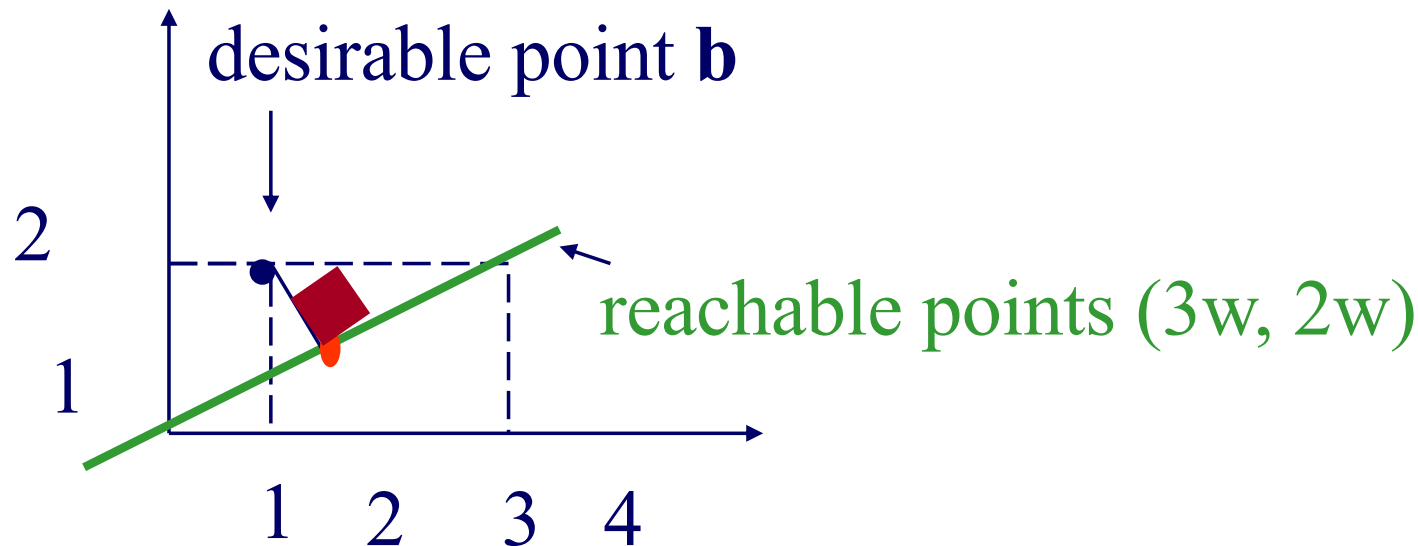
$\begin{bmatrix} 21/13 & 14/13 \end{bmatrix}^T \rightarrow$ 'red point' - perpendicular?



Verify formula:

$$A: [3 \ 2] \cdot ([1 \ 2] - [21/13 \ 14/13]) =$$

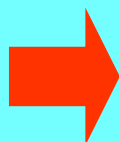
$$[3 \ 2] \cdot [-8/13 \ 12/13] = [3 \ 2] \cdot [-2 \ 3] = 0$$





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

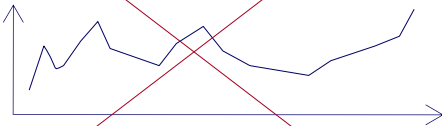
- Given: past queries and their results
 - $\#movies(1925, 1935) = 52$
 - $\#movies(1948, 1990) = 123$
 - ...
 - And a new query, say $\#movies(1979, 1980)?$
- Give your best estimate



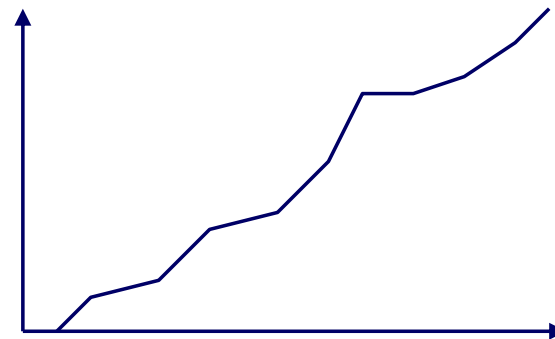
Query feedbacks

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

PDF



count, so far



year

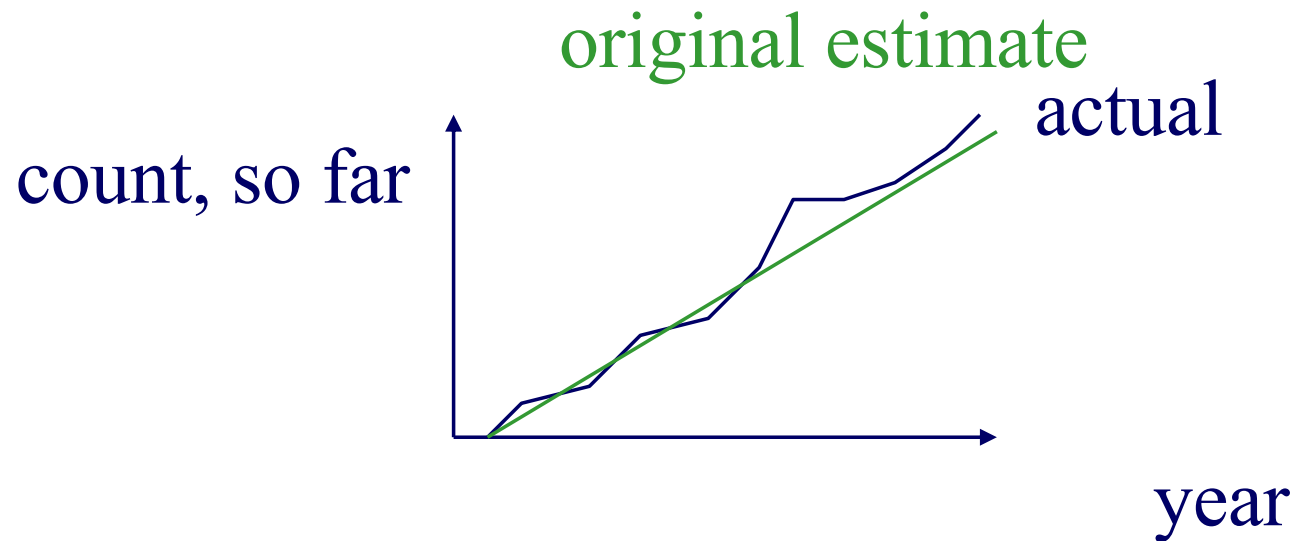
Query feedbacks

For example

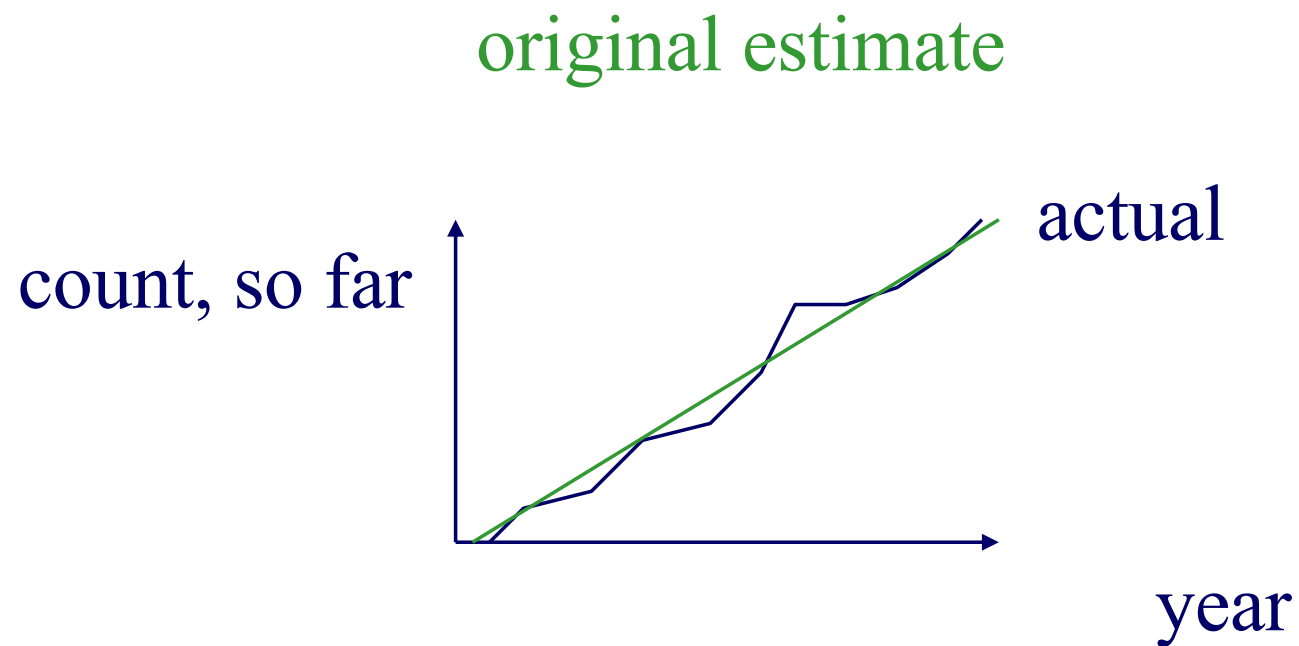
$$\begin{aligned} F(x) &= \# \text{ movies made until year 'x'} \\ &= a_1 + a_2 * x + a_3 * x^2 + \dots a_7 * x^6 \end{aligned}$$

Query feedback

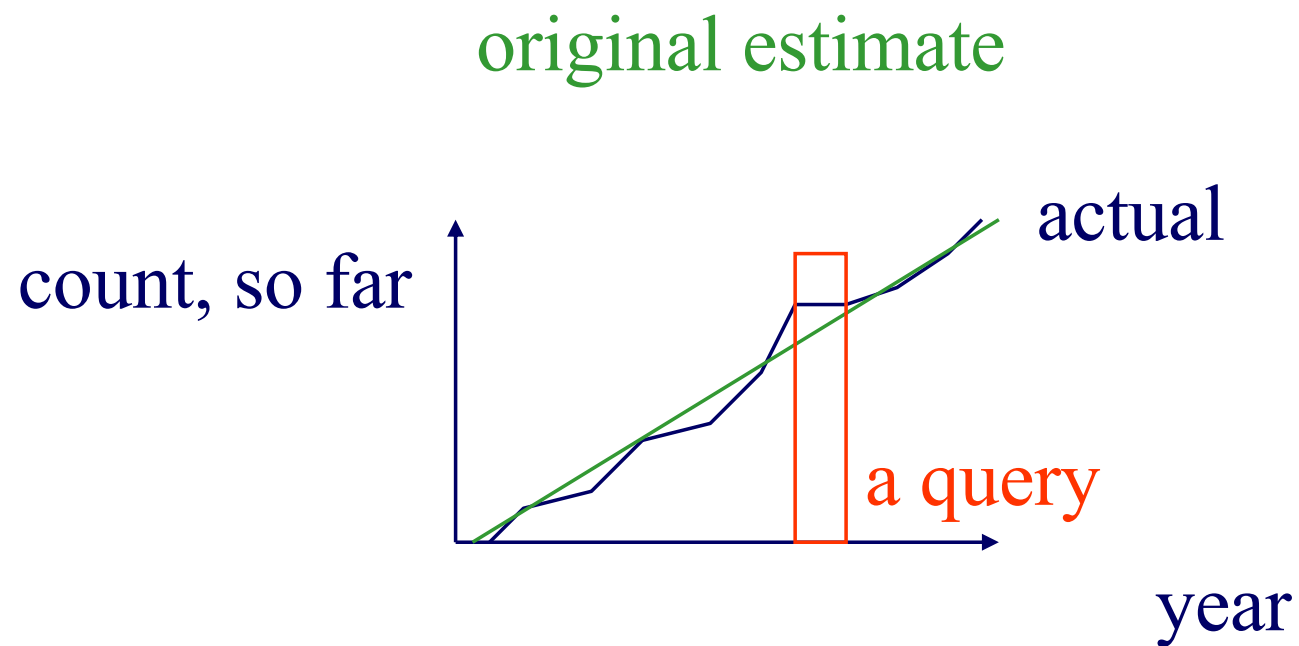
GREAT idea #2: adapt your model, as you see the actual counts of the actual queries



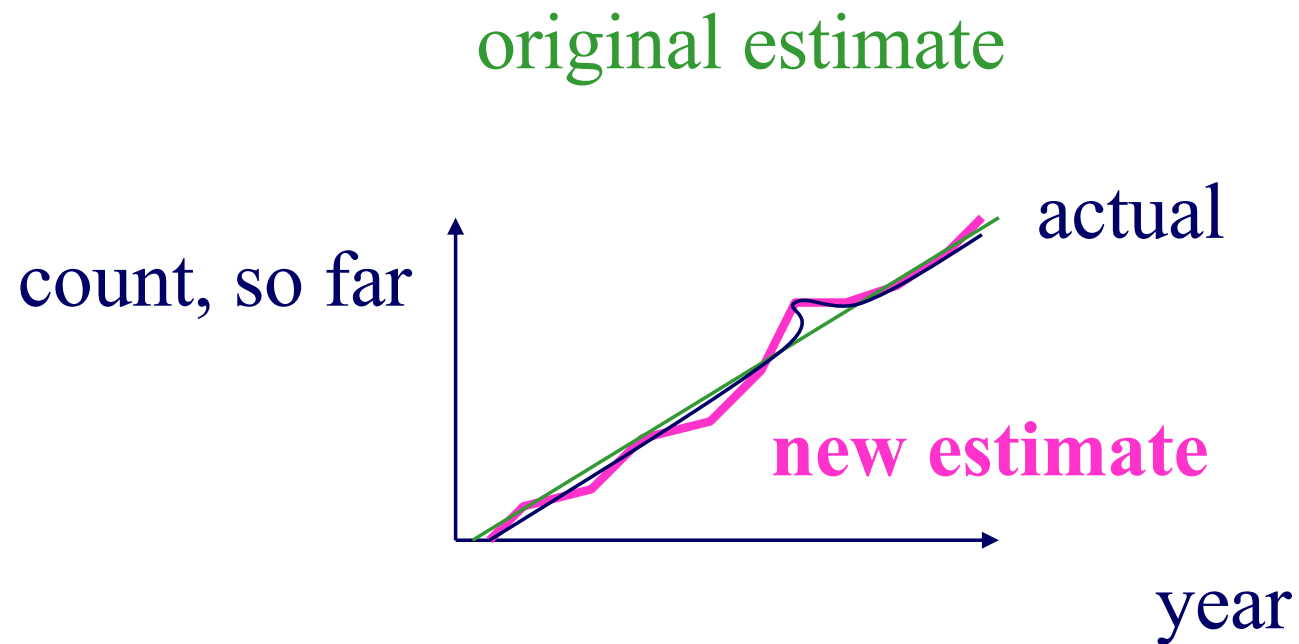
Query feedback



Query feedback



Query feedback



Query feedbacks

Eventually, the problem becomes:

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

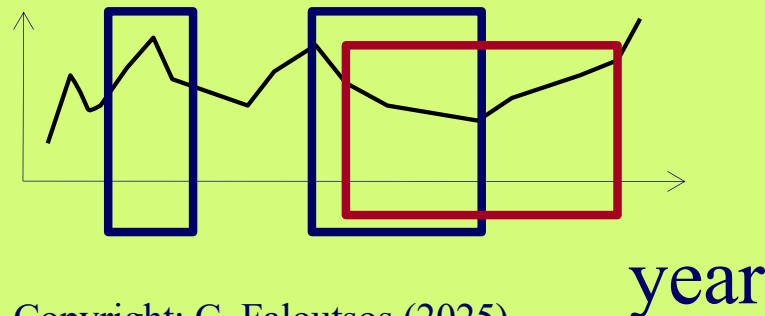
Formally:



Q: How to formulate LSQ?

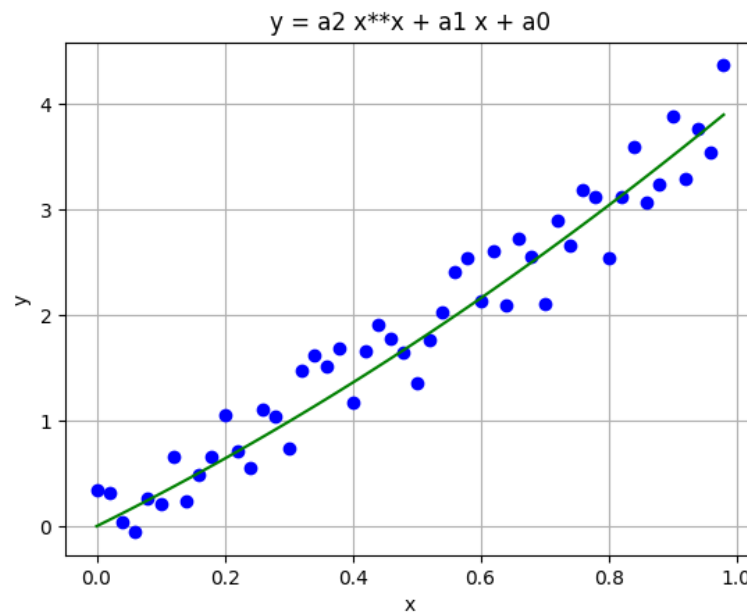
- Given: past queries and their results
 - $\#movies(1925, 1935) = 52$
 - $\#movies(1948, 1990) = 123$
 - ...
 - And a new query, say $\#movies(1979, 1980)?$
- Give your best estimate

$\#movies$



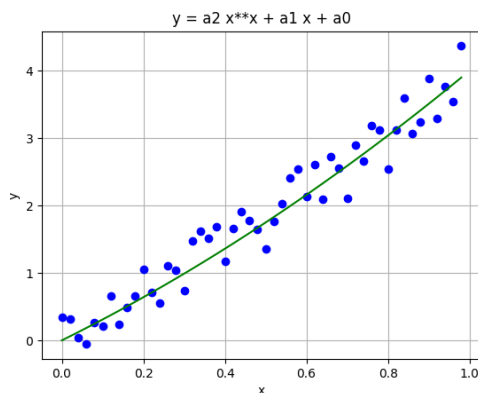
(warm-up - Question)

- How to fit a parabola $y = a_2x^2 + a_1x + a_0$
- To a set of points $(x_1, y_1), \dots (x_n, y_n)$



(warm-up - Answer)

- How to fit a parabola $y = a_2x^2 + a_1x + a_0$
- To a set of points $(x_1, y_1), \dots (x_n, y_n)$



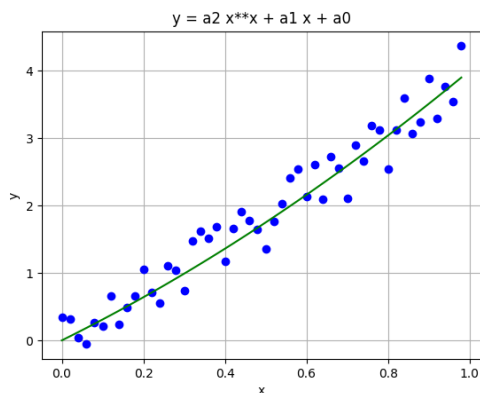
$$a_2x_1^2 + a_1x_1 + a_0 = y_1$$

...

$$a_2x_n^2 + a_1x_n + a_0 = y_n$$

(warm-up - Answer)

- How to fit a parabola $y = a_2x^2 + a_1x + a_0$
- To a set of points $(x_1, y_1), \dots (x_n, y_n)$



$$X \vec{a} = \vec{y}$$

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \dots & \dots & \dots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

Query feedbacks

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

$$\begin{array}{c}
 \left[\begin{array}{ccccc}
 X_{11} & X_{12} & & & X_{17} \\
 & & & & \\
 & & & & \\
 & & & & \\
 & & & & \\
 X_{n1} & X_{n2} & & & X_{n7}
 \end{array} \right]
 \begin{array}{c}
 \left[\begin{array}{c}
 a_1 \\
 a_2 \\
 \\
 \\
 \\
 a_7
 \end{array} \right]
 =
 \begin{array}{c}
 \left[\begin{array}{c}
 b_1 \\
 b_2 \\
 \\
 \\
 \\
 b_n
 \end{array} \right]
 \end{array}
 \end{array}$$

$1,000 \times 7 \qquad 7 \times 1 \qquad 1,000 \times 1$

Query feedback

What is $X_{i,j}$?

What is b_i ?

$$\begin{bmatrix}
 \begin{array}{ccccc}
 X_{11} & X_{12} & & & X_{17} \\
 & & & & \\
 & & & & \\
 & & & & \\
 X_{n1} & X_{n2} & & & X_{n7}
 \end{array} \\
 \end{bmatrix}
 \begin{bmatrix}
 a_1 \\
 a_2 \\
 \\
 \\
 a_7
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 \\
 \\
 b_n
 \end{bmatrix}$$

1,000 x 7

7 x 1

1,000 x 1

Query feedbacks

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

$$\begin{bmatrix}
 \begin{array}{|c|c|c|c|c|}
 \hline
 X_{11} & X_{12} & & & X_{17} \\
 \hline
 & & & & \\
 \hline
 & & & & \\
 \hline
 & & & & \\
 \hline
 X_{n1} & X_{n2} & & & X_{n7} \\
 \hline
 \end{array}
 &
 \begin{bmatrix}
 a_1 \\
 a_2 \\
 \\
 \\
 a_7
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 \\
 \\
 b_n
 \end{bmatrix}
 \end{bmatrix}$$

Query feedbacks

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

$$a_1 + a_2 * 1932 + a_3 * 1932^{**}2 + \dots$$

-

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

$$\begin{bmatrix} \begin{array}{|c|c|c|c|c|} \hline X_{11} & X_{12} & & & X_{17} \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline X_{n1} & X_{n2} & & & X_{n7} \\ \hline \end{array} & \begin{bmatrix} a_1 \\ a_2 \\ \\ \\ \\ a_7 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ \\ \\ b_n \end{bmatrix}$$

Query feedbacks

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

$$\begin{aligned}
 & a_1 + a_2 * 1932 + a_3 * 1932^{**}2 + \dots \\
 - & (a_1 + a_2 * 1920 + a_3 * 1920^{**}2 + \dots)
 \end{aligned}$$

X11	X12			X17
Xn1	Xn2			Xn7

a1
a2
a7

=

b1
b2
bn

Query feedbacks

In matrix form:

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

1st query

n-th query

X11	X12			X17
Xn1	Xn2			Xn7

a1

a2

a7

=

b1

b2

bn

Query feedbacks

In matrix form:

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

and the least-squares estimate for $\mathbf{a}$ is

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

according to property C(1)

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

Query feedbacks - enhancements

The solution

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

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

Q: Can we do it cheaper, incrementally?

Query feedbacks - enhancements

The solution

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

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

Q: Can we do it cheaper, incrementally?

A: YES - GREAT Idea #3: 'Recursive Least Squares

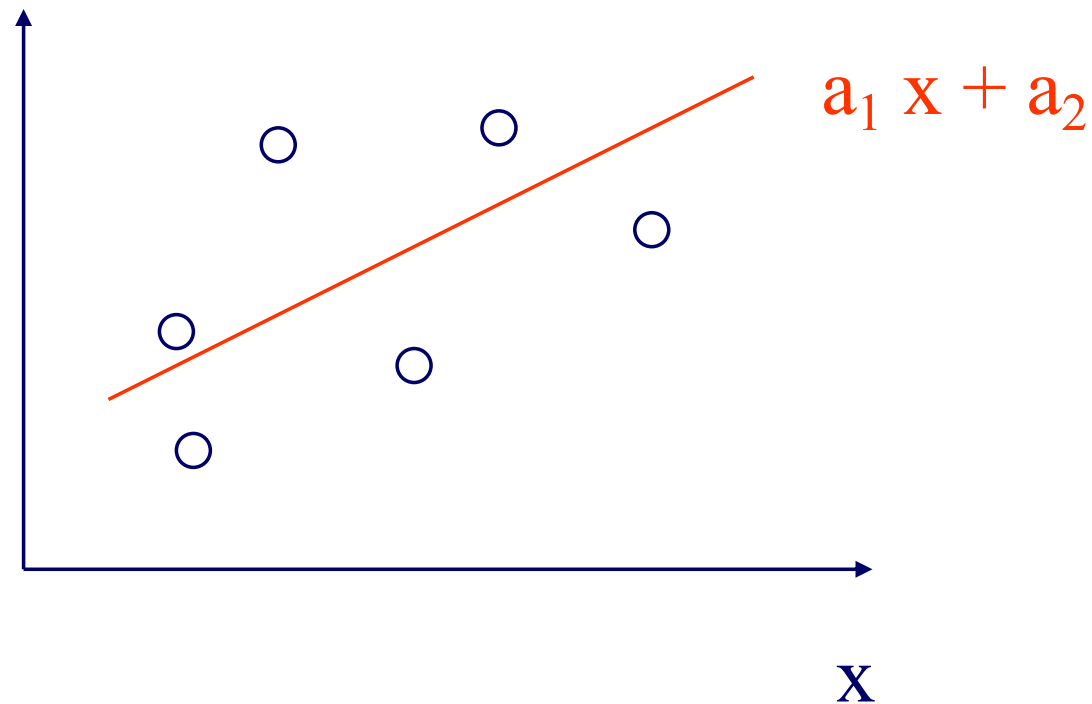
Details: in paper - intuition:

Query feedbacks - enhancements

Intuition:

least squares fit

b

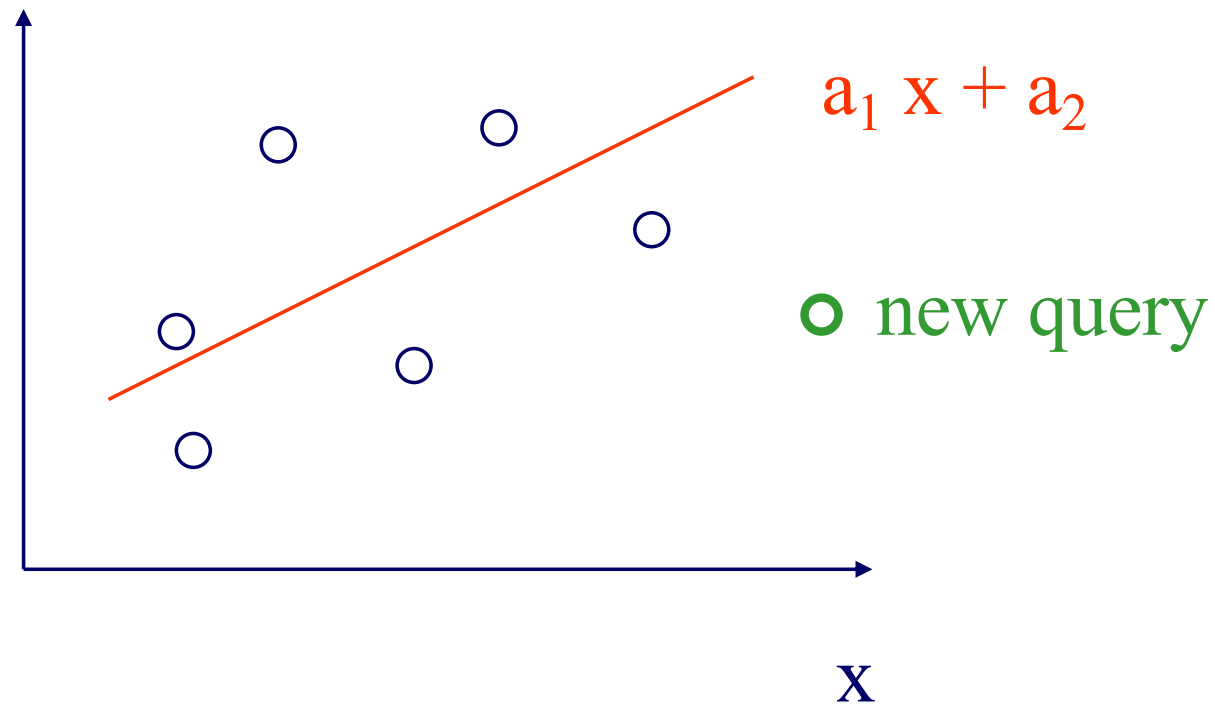


Query feedback - enhancements

Intuition:

least squares fit

b

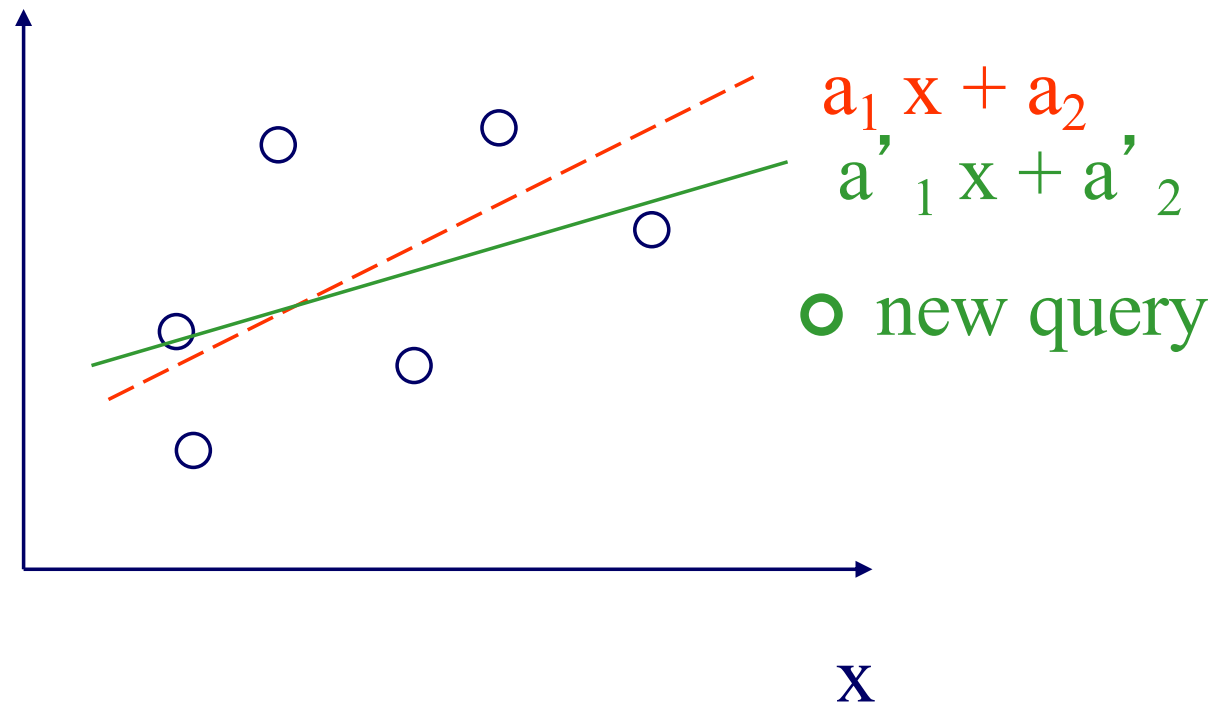


Query feedback - enhancements

Intuition:

least squares fit

b



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

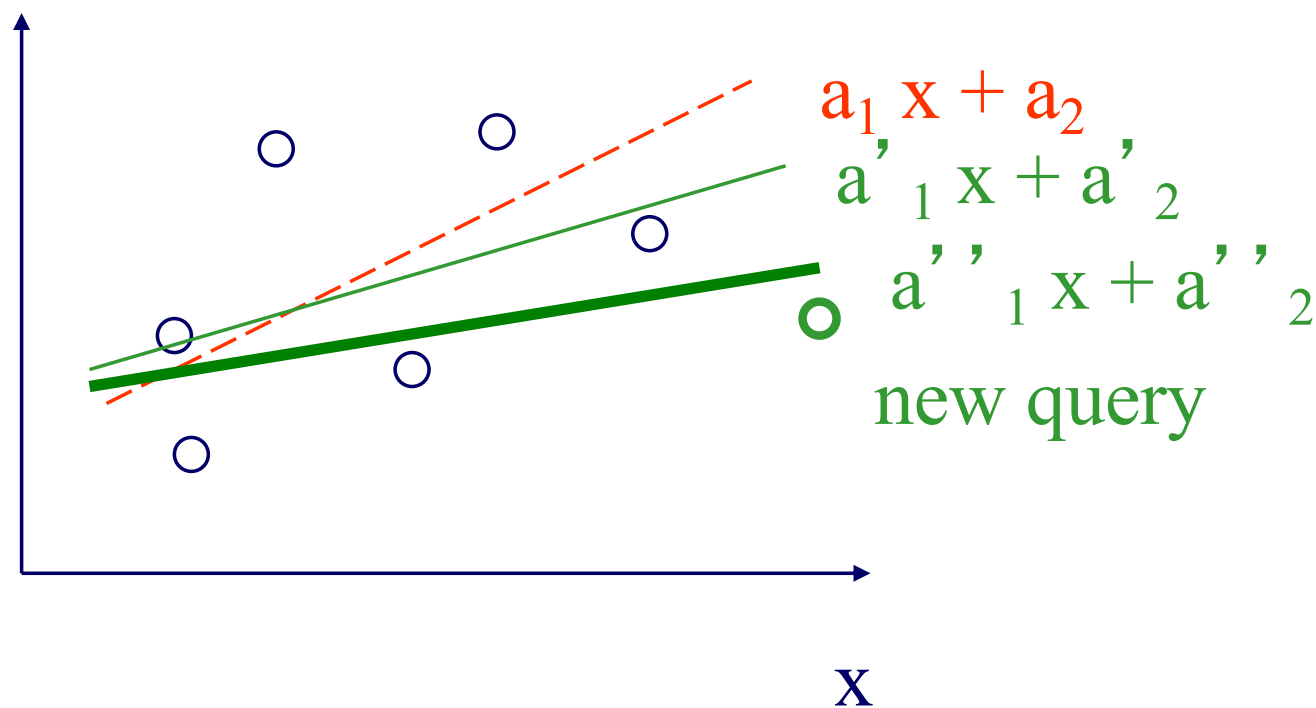
GREAT idea #4: ‘forgetting’ factor - we can even down-play the weight of older queries, since the data distribution might have changed.
(comes for ‘free’ with RLS...)

Query feedback - enhancements

Intuition:

least squares fit

b



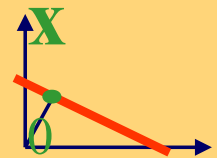
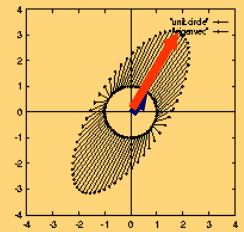
Query feedbacks - conclusions

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 ($\rightarrow$ Moore-Penrose pseudo-inverse)





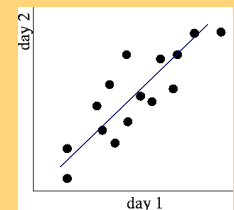
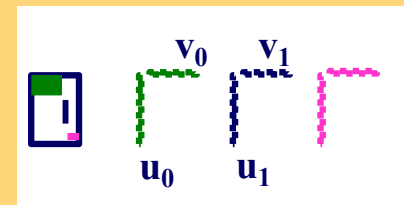
SVD - detailed outline

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



Conclusions (1/4)

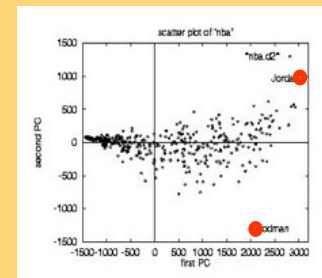
- SVD: a **valuable** tool
 - ‘*the importance of SVD can hardly be overstated*’ [Gilbert Strang]
- given a document-term matrix, it finds ‘concepts’ (LSI)
- ... and finds blocks (‘EigenSpokes’)
- ... and can reduce dimensionality (KL)





Conclusions (2/4)

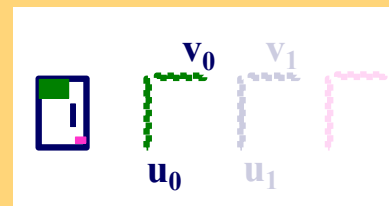
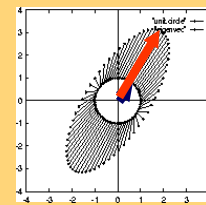
- ...
- ... and can find rules (PCA; RatioRules)
- ... and do visualization





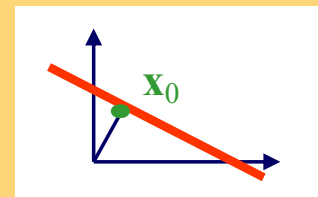
Conclusions (3/4)

- ... 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)

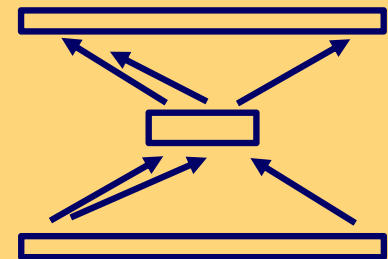
$$\mathbf{x}_0 = \mathbf{V} \mathbf{\Lambda}^{(-1)} \mathbf{U}^T \mathbf{b}$$





Conclusions (4/4)

- Dim. Reduction == embedding; SVD is a special case of neural network:
 - Auto-encoder, 1 hidden layer, L2 loss, linear t.f.



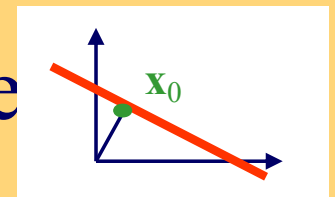
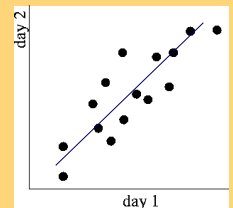
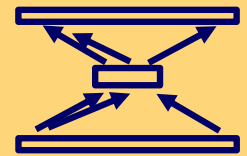
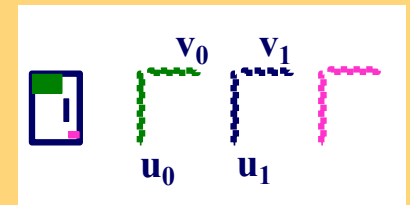
Aapo Hyvärinen, Juha Karhunen, Erkki Oja: *Independent Component Analysis*, John Wiley & Sons, 2001
p. 137



Conclusions – overall summary

SVD:

- Hidden variables
- Blocks
- Dim reduction / embedding
- Visualization
- Most-important node (google/HITS)
- Linear systems (over/exactly/under-spe





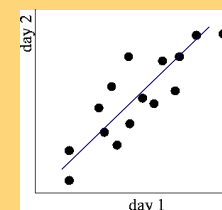
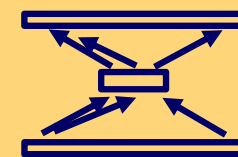
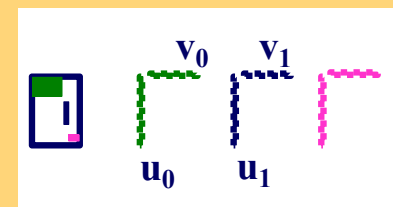
Conclusions – overall summary

SVD:

- Hidden variables

- Blocks

SVD!

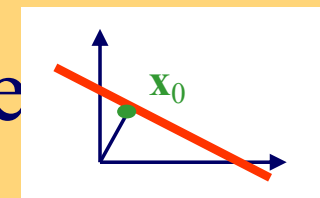


Embedding

ation

Most-important node (google/HITS)

- Linear systems (over/exactly/under-spe



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.