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.
Must-read Material, cont’d


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 \((n \times n)\) square matrix,

\((\lambda, \mathbf{x})\) is an **eigenvalue/eigenvector** pair of \( A \) if

\[
A \mathbf{x} = \lambda \mathbf{x}
\]

CLOSELY related to singular values:
Property #1: Eigen- vs singular-values

if

\[ B_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} (V_{[m \times r]})^T \]

then \( A = (B^T B) \) is symmetric and

\[ B^T B \ v_i = \lambda_i^2 \ v_i \]

ie, \( v_1, v_2, \ldots \): eigenvectors of \( A = (B^T B) \)

Property #2

- If \( A_{[nxn]} \) 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

\[
\begin{bmatrix}
2 \\
1
\end{bmatrix} = \begin{bmatrix}
2 & 1 \\
1 & 3
\end{bmatrix} \begin{bmatrix}
1 \\
0
\end{bmatrix}
\]
Intuition

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

\[
\lambda_1 \begin{bmatrix} 0.52 \\ 0.85 \end{bmatrix} = A \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 0.52 \\ 0.85 \end{bmatrix}
\]

Convergence

• Usually, fast:
Convergence

• Usually, fast:

\[ \lambda_1 : \lambda_2 \]
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’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’)

observations
• recursive definition!
• each node (say, ‘i’-th node) has both an authoritativenss 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 $[n \times 1]$ vectors with the ‘hubness’ and ‘authoritativiness’ scores.
Then:

$$a_i = h_k + h_l + h_m$$

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

or
$$a = A^T h$$
Kleinberg’s algorithm

\[ h_i = a_n + a_p + a_q \]

that is
\[ h_i = \text{Sum} (q_j) \over all\ j\ that\ (i,j)\ edge\ exists \]
or
\[ h = A a \]

In conclusion, we want vectors \( h \) and \( a \) such that:
\[ h = A a \]
\[ a = A^T h \]

Properties:
- \( A [n \times m] v_1 [m \times 1] = \lambda_1 u_1 [n \times 1] \)
- \( u_1^T A = \lambda_1 v_1^T \)
In short, the solutions to
\[ h = A \ a \]
\[ a = A^T \ h \]
are the **left- and right- singular-vectors** of the adjacency matrix \( A \).
Starting from random \( a' \) and iterating, we’ll eventually converge
(Q: to which of all the singular-vectors? why?)

---

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

A: to the ones of the strongest singular-value, because of property:
\[ (A^T A)^k \ v' \sim (\text{constant}) \ v_1 \]
Proof, and intuition

Proof (sketch)

\[ A = U \Lambda V^T \quad U^T U = I \quad V^T V = I \]

1) \[ A^T A = V \Lambda U^T U \Lambda V^T = V \Lambda^2 V^T \]
Proof (sketch)

\[ A = U \Lambda V^T \quad U^T U = I \quad V^T V = I \]

1) \( A^T A = V \Lambda U^T U \Lambda V^T = V \Lambda^2 V^T \)

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

3) \( (A^T A)^k v' \approx \lambda_1^{2k} v_1 v_1^T v' \)
Proof (sketch) - pictorial

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

More intuition

- \((A^T A)^k v' \sim (\text{constant}) v_1\)
- \(= (A^T A) \cdots (A^T A) v' \sim (\text{constant}) v_1\) \(k\) times
More intuition

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

```
A^T  A
```

users  products

Smith's preferences

```
A  A^T
```

users  products

(libraries)  (books)
More intuition

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

users\[\]
products\[\]

More intuition

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

similarities to Smith

users\[\]
products\[\]
More intuition

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

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

(i.e., 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)
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 (\(<\rightarrow\) 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)


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 \(A\) be the adjacency matrix;
- let \(B\) be the transition matrix: transpose, column-normalized - then

\[
B = \begin{bmatrix}
1 & 1 \\
1/2 & 1/2 \\
1/2 & 1/2 \\
1/2 & 1/2
\end{bmatrix}
\]

\[
p = \begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5
\end{bmatrix}
\]

\[
p = \begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5
\end{bmatrix}
\]
(Simplified) PageRank algorithm

- \( B \cdot p = p \)

\[
\begin{bmatrix}
1 & 1 \\
1/2 & 1/2 \\
1/2 &
\end{bmatrix}
\begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
\end{bmatrix}
= 
\begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
\end{bmatrix}
\]

- \( B \cdot p = 1 \cdot p \)
- 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]

(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
  \[ p = c \mathbf{B} p + \frac{(1-c)}{n} \mathbf{1} \Rightarrow \]
  \[ p = \frac{(1-c)}{n} \left[ \mathbf{I} - c \mathbf{B} \right]^{-1} \mathbf{1} \]
Alternative notation – eigenvector viewpoint

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

Then

\[ \mathbf{p} = \mathbf{M} \mathbf{p} \]

That is: the steady state probabilities = PageRank scores form the *first eigenvector* of the ‘modified transition matrix’

---

Personalized P.R.

  
  [http://dx.doi.org/10.1145/511446.511513](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?)

![Diagram of a network with nodes 1, 2, 3, 4, and 5 connected by arrows.]
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
Extension: Personalized P.R.

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

• Then, we have

$$p = c \mathbf{B} \ p + \frac{1-c}{n} \ e$$

$$p = \frac{1-c}{n} \left[ I - c \mathbf{B} \right]^{-1} \ e$$

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?

- A: ‘Pixie’ algorithm
Extension: Personalized P.R.

- Q: Faster computation than:
  \[ p = \frac{(1-c)}{n} \left[ I - cB \right]^{-1} e \]

Pixie algorithm


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

- Q: Faster computation than:
  \[ p = \frac{1-c}{n} \left( I - c B \right)^{-1} e \]
- A: **simulate** a few R.W.
  - keep visit counts \( C_i \)
  - fast and nimble

**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 (<> 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

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

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

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): \( A_{n \times m} = U_{n \times r} A_{r \times r} V^T_{r \times m} \)

C(1): \( A_{n \times m} x_{m \times 1} = b_{n \times 1} \)

let \( x_0 = V A^{(-1)} U^T b \)

Slowly:

\( \sim \)

\( \sim \)

\( \sim \)

\( \sim \)

\( \sim \)
Slowly:

\[ \begin{align*}
\text{Identity} \\
U: \text{column-orthonormal}
\end{align*} \]
Slowly:

NOT proof

Slowly:

NOT proof
Slowly:

\[ \mathbf{I} = \mathbf{I} \]

\[ \mathbf{I} = \mathbf{I} \]

\[ x \quad V \quad \mathbf{A}^{-1} \quad \mathbf{U}^T \quad \mathbf{b} \]
Slowly:

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

$$| = \begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{V} & \mathbf{\Lambda}^{-1} & \mathbf{U}^T \end{bmatrix} \mathbf{x}$$

---

Recursive Least Squares (RLS)

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

$$| = \begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{V} & \mathbf{\Lambda}^{-1} & \mathbf{U}^T \end{bmatrix} \mathbf{x}$$
Recursive Least Squares (RLS)

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

\[
\begin{align*}
L &= V L^{-1} U^T b \\
&= x
\end{align*}
\]

Drills:
Least obvious properties

Illustration: under-specified, e.g.

\[
\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} w \\ z \end{bmatrix}^T = 4 \quad \text{(i.e., } 1w + 2z = 4) \]

\[ A = \text{??} \]
\[ b = \text{??} \]

shortest-length solution

all possible solutions

Verify formula:

\[ A = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad b = \begin{bmatrix} 4 \end{bmatrix} \]
\[ A = U \Lambda V^T \]
\[ U = \text{??} \]
\[ \Lambda = \text{??} \]
\[ V = \text{??} \]
\[ x_0 = V \Lambda^{-1} U^T b \]
Verify formula:

\[
\begin{align*}
\mathbf{A} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 4 \end{bmatrix} \\
\mathbf{A} &= \mathbf{U} \Lambda \mathbf{V}^T \\
\mathbf{U} &= \begin{bmatrix} 1 \end{bmatrix} \\
\Lambda &= \begin{bmatrix} \sqrt{5} \end{bmatrix} \\
\mathbf{V} &= \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}^T \\
\mathbf{x}_0 &= \mathbf{V} \Lambda^{-1} \mathbf{U}^T \mathbf{b}
\end{align*}
\]
Verify formula:

Show that \( w = \frac{4}{5}, \ z = \frac{8}{5} \) is

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

(b) Minimal (wrt Euclidean norm)

A: easy

A: \([4/5 \ 8/5]\) is perpendicular 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 )

\[ A=?? \]
\[ b=?? \]

Verify formula:

\[ A = [3 \ 2]^T \quad b = [1 \ 2]^T \]
\[ A = U \Lambda V^T \]
\[ U = ?? \]
\[ \Lambda = ?? \]
\[ V = ?? \]
\[ x_0 = V \Lambda^{-1} U^T b \]
Verify formula:

\[ A = [3 \ 2]^T \quad b = [1 \ 2]^T \]
\[ A = U \Lambda V^T \]
\[ U = \begin{bmatrix} 3/\sqrt{13} & 2/\sqrt{13} \end{bmatrix}^T \]
\[ \Lambda = \begin{bmatrix} \sqrt{13} \end{bmatrix} \]
\[ V = \begin{bmatrix} 1 \end{bmatrix} \]
\[ x_0 = V \Lambda^{-1} U^T b = \begin{bmatrix} 7/13 \end{bmatrix} \]
Verify formula:

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

\[ [21/13 \ 14/13]^T \rightarrow \text{‘red point’ - perpendicular?} \]

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

![Graph showing the number of movies over years](image)

Query feedbacks

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

![Graph showing cumulative distribution function](image)
Query feedbacks

For example

\[ F(x) = \# \text{movies made until year} \ 'x' \]

\[ = a_1 + a_2 \cdot x + a_3 \cdot x^2 + \ldots + a_7 \cdot x^6 \]

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

original estimate

count, so far

actual

year

Query feedbacks

original estimate

count, so far

actual

year

a query
Eventually, the problem becomes:
- estimate the parameters $a_1, \ldots, a_7$ 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:

\[
\begin{bmatrix}
X_{11} & X_{12} & \cdots & X_{17} \\
X_{n1} & X_{n2} & \cdots & X_{n7}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_7
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2 \\
b_n
\end{bmatrix}
\]

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

What is \( X_{ij} \)?
What is \( b_i \)?
Query feedbacks

where \( x_{i,j} \) such that \( \text{Sum}(x_{i,j} \times a_i) = \) our estimate for the # of movies and \( b_j \); the actual

\[
\begin{bmatrix}
X_{11} & X_{12} & X_{17} \\
\vdots & \ddots & \vdots \\
X_{n1} & X_{n2} & X_{n7}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_7
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{bmatrix}
\]

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

\[
a_1 + a_2 \times 1932 + a_3 \times 1932^{**2} + \ldots - (a_1 + a_2 \times 1920 + a_3 \times 1920^{**2} + \ldots)
\]
Query feedbacks

And thus \( X_{11} = 0; X_{12} = 1932-1920, \) etc

\[
\begin{align*}
& (a_1 + a_2 \cdot 1920 + a_3 \cdot 1920^2 + \ldots) \\
& \quad - (a_1 + a_2 \cdot 1932 + a_3 \cdot 1932^2 + \ldots)
\end{align*}
\]

In matrix form:

\[
\begin{bmatrix}
X_{11} & X_{12} & \cdots & X_{17} \\
X_{n1} & X_{n2} & \cdots & X_{n7}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
a_7
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2 \\
b_7 \\n
\end{bmatrix}
\]

1st query

n-th query
Query feedbacks

In matrix form:

\[ \mathbf{X} \mathbf{a} = \mathbf{b} \]

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

\[ \mathbf{a} = \mathbf{V} \Lambda^{-1} \mathbf{U}^T \mathbf{b} \]

according to property C(1)

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

---

Query feedbacks - enhancements

The solution

\[ \mathbf{a} = \mathbf{V} \Lambda^{-1} \mathbf{U}^T \mathbf{b} \]

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

GREAT Idea #3: Use ‘Recursive Least Squares’, to adapt \( \mathbf{a} \) incrementally.

Details: in paper - intuition:
Query feedbacks - enhancements

Intuition:

least squares fit

new query

\[ a_1 x + a_2 \]
Query feedbacks - enhancements

Intuition:

Least squares fit

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...)

Intuition:

least squares fit

\[ a_1 x + a_2 \]
\[ a'_1 x + a'_2 \]
\[ a''_1 x + a''_2 \]

new query
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


References cont’d