15-826: Multimedia Databases and Data Mining

Lecture #20: SVD - part III (more case studies)

C. Faloutsos

Must-read Material

• Textbook Appendix 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

SVD - detailed outline

- ...
- Case studies
- SVD properties
  - more case studies
    - google/Kleinberg algorithms
    - query feedbacks
  - Conclusions
SVD - Other properties - summary
• can produce orthogonal basis (obvious) (who cares?)
• can solve over- and under-determined linear problems (see C(1) property)
• can compute ‘fixed points’ (= ‘steady state prob. in Markov chains’) (see C(4) property)

SVD - outline of properties
• (A): obvious
• (B): less obvious
• (C): least obvious (and most powerful!)

Properties - by defn.:
A(0): \( A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]} \)

A(1): \( U^T_{[r \times n]} U_{[n \times r]} = I_{[r \times r]} \) (identity matrix)
A(2): \( V^T_{[r \times n]} V_{[n \times r]} = I_{[r \times r]} \)
A(3): \( \Lambda^k = \text{diag}(\lambda_1^k, \lambda_2^k, \ldots, \lambda_r^k) \) (k: ANY real number)
A(4): \( A^T = V \Lambda U^T \)
Less obvious properties

A(0): \( A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]} \)

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

symmetric; Intuition?

B(2): symmetrically, for ‘\( V \)’

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

Intuition?
Less obvious properties

A: term-to-term similarity matrix

B(3): \(( A^T A )^{[m \times n]} \) for k to infinity

and

B(4): \(( A^T A )^k \) for k to infinity

where

\( v_1 \): first column (singular-vector) of V

\( \lambda_1 \): strongest singular value

Proof of (B4)?

Less obvious properties

B(4): \(( A^T A )^k \) for k to infinity

B(5): \(( A^T A )^k \) for k to infinity

ie., for any \( v^i \), it converges to a vector parallel to \( v_1 \)

Thus, useful to compute first singular vector

/value as well as the next ones, too...
Proof of (B5)?

Property (B5)

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


Property (B5)

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

Smith’s preferences

Smith
Property (B5)

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

![Image of users, products, and similarities to Smith]
Property (B5)

- Intuition:
  - \((A^T A) v'\) what Smith’s ‘friends’ like
  - \((A^T A)^k v'\) what k-step-away-friends like

(i.e., after \(k\) steps, we get what everybody likes, and Smith’s initial opinions don’t count)

Less obvious properties - repeated:

\(A(0): A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]}\)

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

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

\(B(3): (A^T A)^k \sim V_1 \lambda_1^{2k} V_1^T\)

\(B(4): (A^T A)^k v' \sim (\text{constant}) v_1\)

Least obvious properties

\(A(0): A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]}\)

\(C(1): A_{[n \times m]} \times_{[m \times 1]} = b_{[n \times 1]}\)

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

if under-specified, \(x_0\) gives ‘shortest’ solution
if over-specified, it gives the ‘solution’ with the smallest least squares error

(see Num. Recipes, p. 62)
Least obvious properties

Illustration: under-specified, eg
\[
\begin{bmatrix}
1 & 2
\end{bmatrix}
\begin{bmatrix}
w \\
z
\end{bmatrix}^T = 4 \quad \text{(ie, } 1 w + 2 z = 4)\]

- Shortest-length solution
- All possible solutions

Verify formula:

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

\[
A = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad b = \begin{bmatrix} 4 \end{bmatrix}
\]

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

\[
U = \begin{bmatrix} 1 \end{bmatrix}
\]

\[
\Lambda = \begin{bmatrix} \sqrt{5} \end{bmatrix}
\]

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

\[
x_0 = V \Lambda^{-1} U^T b = \begin{bmatrix} 1/5 & 2/5 \end{bmatrix}^T \begin{bmatrix} 4 \end{bmatrix} = \begin{bmatrix} 4/5 & 8/5 \end{bmatrix}^T \quad w= 4/5, z = 8/5
\]

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)

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: \( [4/5 \quad 8/5] \) is perpenticular to \( [2 \quad -1] \)
Least obvious properties – cont’d

Illustration: over-specified, eg
\[ [3 \ 2]^T \ [w] = [1 \ 2]^T \] (ie, 3 \ w = 1; 2 \ w = 2 )

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

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

\[
\begin{bmatrix} 3 & 2 \end{bmatrix}^T \begin{bmatrix} 7/13 \\ 21/13 & 14/13 \end{bmatrix}^T \rightarrow \text{‘red point’}
\]

Reachable points (3w, 2w)

Verify formula:

\[
\begin{bmatrix} 3 & 2 \end{bmatrix}^T \begin{bmatrix} 7/13 \\ 21/13 & 14/13 \end{bmatrix}^T \rightarrow \text{‘red point’ - perpendicular?}
\]

Reachable points (3w, 2w)

Verify formula:

A: \[
\begin{bmatrix} 3 & 2 \end{bmatrix} \cdot (\begin{bmatrix} 1 & 2 \\ 21/13 & 14/13 \end{bmatrix} - \begin{bmatrix} -8/13 & 12/13 \end{bmatrix}) = \]
\[
\begin{bmatrix} 3 & 2 \end{bmatrix} \cdot \begin{bmatrix} -2 & 3 \end{bmatrix} = 0
\]
Least obvious properties - cont’d

A(0): $A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]}$

C(2): $A_{[n \times m]} v_1_{[m \times 1]} = \lambda_1 u_1_{[n \times 1]}$
where $v_1$, $u_1$ the first (column) vectors of $V$, $U$. ($v_1$ == right-singular-vector)
C(3): symmetrically: $u_1^T A = \lambda_1 v_1^T$
$u_1$ == left-singular-vector
Therefore:

C(4): $A^T A v_1 = \lambda_1^2 v_1$

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

Least obvious properties - altogether

A(0): $A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]}$

C(1): $A_{[n \times m]} x_{[m \times 1]} = b_{[n \times 1]}$
then, $x_0 = V \Lambda^{-1} U^T b$: shortest, actual or least-squares solution
C(2): $A_{[n \times m]} v_1_{[m \times 1]} = \lambda_1 u_1_{[n \times 1]}$
C(3): $u_1^T A = \lambda_1 v_1^T$
C(4): $A^T A v_1 = \lambda_1^2 v_1$
Properties - conclusions

A(0): \( A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} V^T_{[r \times m]} \)

B(5): \( (A^T A)^k \mathbf{v} \sim \text{(constant)} \mathbf{v}_1 \)

C(1): \( A_{[n \times m]} x_{[m \times 1]} = b_{[n \times 1]} \)
then, \( x_0 = V \Lambda^{-1} U^T b \): shortest, actual or least-
squares solution

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

SVD - detailed outline

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

Kleinberg’s algo (HITS)

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’
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 $[n \times 1]$ vectors with the
‘hubness’ and ‘authoritativeness’ scores.
Then:

Kleinberg’s algorithm

Then:

$$a_i = h_k + h_l + h_m$$

that is

$$a_i = \text{Sum}(h_j) \text{ over all } j \text{ that}$$

$$(j,i) \text{ edge exists}$$
or

$$a = A^T h$$
Kleinberg’s algorithm

**Symmetry:** for the ‘hubness’:

\[ h_i = a_n + a_p + a_q \]

that is

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

or

\[ h = A a \]

---

Kleinberg’s algorithm

In conclusion, we want vectors \( h \) and \( a \) such that:

\[ h = A a \]

\[ a = A^T h \]

Recall properties:

\[ C(2): A \begin{bmatrix} v_1 \end{bmatrix} = \lambda_1 u_1 \begin{bmatrix} 1 \end{bmatrix} \]

\[ C(3): u_1^T A = \lambda_1 v_1^T \]

---

Kleinberg’s algorithm

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?)
Kleinberg’s algorithm

(Q: to which of all the singular-vectors? why?)
A: to the ones of the strongest singular-value, because of property B(5):
B(5): \((A^T A)^k v^* \sim \text{(constant)} v_1\)

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
SVD - detailed outline

- ...
- Case studies
- SVD properties
- more case studies
  - Kleinberg/google algorithms
  - 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 (→ 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

\[
B p = p
\]
(Simplified) PageRank algorithm

- \( B \mathbf{p} = \mathbf{1} \times \mathbf{p} \)
- 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 \( B \) is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

In short: imagine a particle randomly moving along the edges

- compute its steady-state probabilities (ssp)

Full Algorithm

- With probability \( 1-c \), fly-out to a random node
- Then, we have
  \[ \mathbf{p} = c B \mathbf{p} + (1-c)/n \mathbf{1} \Rightarrow \mathbf{p} = (1-c)/n \left[ I - c B \right]^{-1} \mathbf{1} \]
Full Algorithm

• With probability $1-c$, fly-out to a random node
• Then, we have
  \[ p = c \mathbf{B} p + \frac{(1-c)}{n} 1 \Rightarrow \]
  \[ p = \frac{(1-c)}{n} \left( I - c \mathbf{B} \right)^{-1} 1 \]

Alternative notation – eigenvector viewpoint

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

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

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

Parenthesis: intuition behind eigenvectors

• Definition
• 3 properties
• intuition
Formal definition

If \( A \) is a \((n \times n)\) square matrix, 
\((\lambda, x)\) is an eigenvalue/eigenvector pair of \( A \) if

\[ A \times x = \lambda \times x \]

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

C(4): \( B^T B \times v_i = \lambda_i^2 \times v_i \)

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

Property #2

- If \( A_{[n \times n]} \) is a real, symmetric matrix
- Then it has \( n \) real eigenvalues

(if \( A \) is not symmetric, some eigenvalues may be complex)
Property #3

- If $A_{nxn}$ is a real, symmetric matrix
- Then it has $n$ real eigenvalues
- And they agree with its $n$ singular values, except possibly for the sign

Parenthesis: intuition behind eigenvectors

- Definition
- 3 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")

\[ \lambda_1 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = A \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \]

**Convergence**

- Usually, fast:
Convergence

- Usually, fast:
- depends on ratio \( \lambda_1 : \lambda_2 \)

Closing the parenthesis wrt intuition behind eigenvectors

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
SVD - detailed outline

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

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

year
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

- original estimate
- actual
- count, so far
- year
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 $n$ queries and 6-th degree polynomials:

$$\begin{bmatrix}
  x_{i1} & x_{i2} & x_{i3} \\
  x_{j1} & x_{j2} & x_{j3} \\
  x_{k1} & x_{k2} & x_{k3}
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}$$

Query feedbacks

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

$$\begin{bmatrix}
  x_{i1} & x_{i2} & x_{i3} \\
  x_{j1} & x_{j2} & x_{j3} \\
  x_{k1} & x_{k2} & x_{k3}
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}$$

Query feedbacks

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

$$a_1 + a_2 \cdot 1932 + a_3 \cdot 1932**2 + …$$

$$= (a_1 + a_2 \cdot 1920 + a_3 \cdot 1920**2 + …)$$

$$\begin{bmatrix}
  x_{i1} & x_{i2} & x_{i3} \\
  x_{j1} & x_{j2} & x_{j3} \\
  x_{k1} & x_{k2} & x_{k3}
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}$$
Query feedbacks

And thus $X_{11} = 0; X_{12} = 1932-1920$, etc

$$a_1 + a_2 * 1932 + a_3 * 1932^2 + \ldots$$

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

In matrix form:

$$X a = b$$

1st query

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$n$-th query

In matrix form:

$$X a = b$$

and the least-squares estimate for $a$ is

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

according to property C(1)

(let $X = U \Lambda V^T$)
Query feedbacks - enhancements

The solution
\[
a = V \Lambda^{(-1)} U^T b
\]
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:

Intuition:

least squares fit
Query feedbacks - enhancements

Intuition:

least squares fit

\[ a_1 x + a_2 \]
\[ a_1' x + a_2' \]

new query

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)

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

\[ a_1 x + a_2 \]

\[ a_1' x + a_2' \]

\[ a_1'' x + a_2'' \]

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)

SVD - detailed outline

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

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

Conclusions cont’d

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

References

References cont’d