

Latent Semantic Indexing Using SVD and Riemannian SVD

Jennifer Flynn

University of California, Santa Cruz

June 7, 2007

Outline

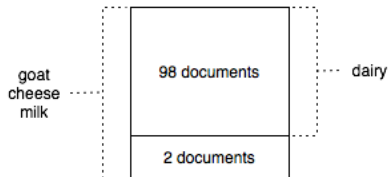
- 1 Latent Semantic Indexing (LSI)
- 2 Riemannian SVD (RSVD)

Outline

1 Latent Semantic Indexing (LSI)

2 Riemannian SVD (RSVD)

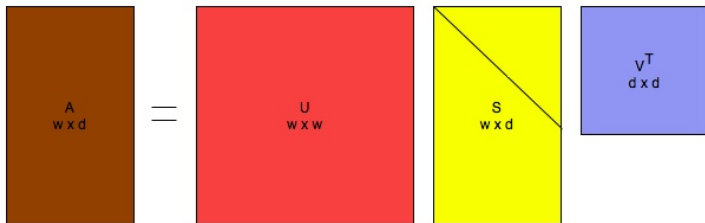
- Capture concepts instead of words
- Why?
 - Polysemy: words have multiple meanings
 - Synonymy: multiple words have the same meaning
- Goal: model reliability of words as a statistical probability



Desired Model

- Represent both words and documents
- Adjustable power or richness
- Feasible complexity
- Doable with truncated SVD

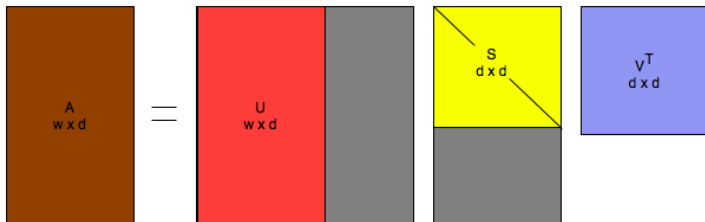
Singular Value Decomposition

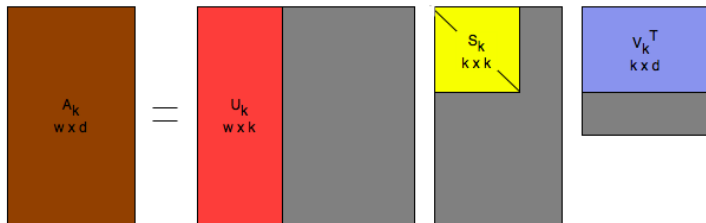


- U and V are unitary, left and right singular vectors
- S is diagonal of singular values, ordered non-decreasingly

Singular Value Decomposition

S is all zeros after row d so:



SVD Rank k Approximation

- Rank k approximation keeps the top k singular values
- A_k now based on major patterns

An Optimization Problem

- A_k is best possible approximation to A (in a least-squares sense)
- As an optimization problem:

$$\min_B \|A - B\|_F^2 \text{ subj to } By = 0 \text{ and } y^T y = 1$$

- When B is lower rank than A we have linear least squares:

$$\min_B \|A - B\|_F^2 = \sigma^2$$

where

$$A = \sum \sigma_i u_i v_i^T$$

Example of SVD for LSI

[DeDuFLH]

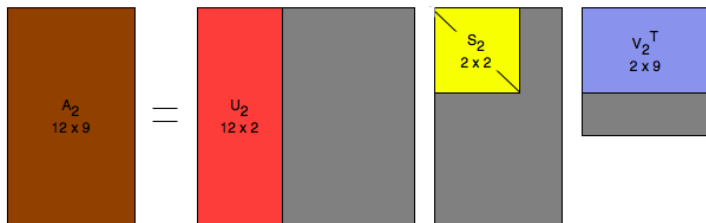
Create a word-by-document matrix

	d1	d2	d3	d4	d5	d6	d7	d8	d9
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	0	0	0	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

d1: **Human** machine **interface** for Lab ABC **computer** applications

The Approximation Matrix

- Get rank 2 approximation



The Approximation Matrix (2)

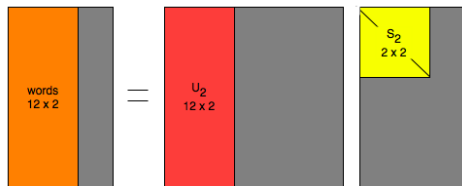
- Goal: Predict what the word-document matrix should be

$$\begin{array}{cccccccc}
 & A & & \text{SVD} \rightarrow & & A_2 & & \\
 \begin{array}{cccccccc}
 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
 \end{array} & & & & & & & \\
 \begin{array}{cccccccc}
 0.16 & 0.40 & 0.38 & 0.47 & 0.18 & -0.05 & -0.12 & -0.16 & -0.09 \\
 0.14 & 0.37 & 0.33 & 0.40 & 0.16 & -0.03 & -0.07 & -0.10 & -0.04 \\
 0.15 & 0.51 & 0.36 & 0.41 & 0.24 & 0.02 & 0.06 & 0.09 & 0.12 \\
 0.26 & 0.84 & 0.61 & 0.70 & 0.39 & 0.03 & 0.08 & 0.12 & 0.19 \\
 0.45 & 1.23 & 1.05 & 1.27 & 0.56 & -0.07 & -0.15 & -0.21 & -0.05 \\
 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\
 0.16 & 0.58 & 0.38 & 0.42 & 0.28 & 0.06 & 0.13 & 0.19 & 0.22 \\
 0.22 & 0.55 & 0.51 & 0.63 & 0.24 & -0.07 & -0.14 & -0.20 & -0.11 \\
 0.10 & 0.53 & 0.23 & 0.21 & 0.27 & 0.14 & 0.31 & 0.44 & 0.42 \\
 -0.06 & 0.23 & -0.14 & -0.27 & 0.14 & 0.24 & 0.55 & 0.77 & 0.66 \\
 -0.06 & 0.34 & -0.15 & -0.30 & 0.20 & 0.31 & 0.69 & 0.98 & 0.85 \\
 -0.04 & 0.25 & -0.10 & -0.21 & 0.15 & 0.22 & 0.50 & 0.71 & 0.62
 \end{array}
 \end{array}$$

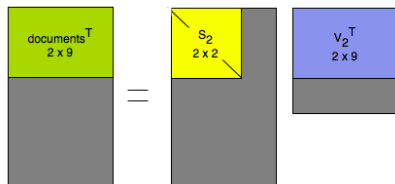
- Rank of approximation is $k = 2$

Representing Words and Documents

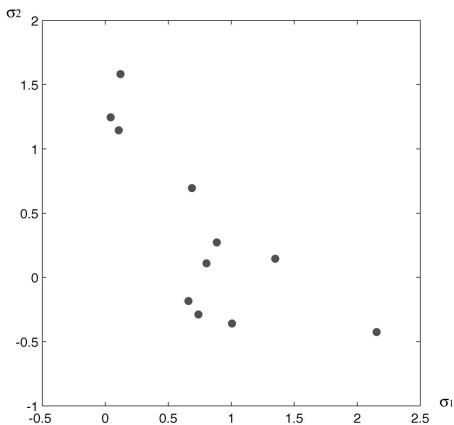
- Project words and documents into 2-space
- Word locations are $U_2 \times S_2$



- Document locations are $(S_2 \times V_2^T)^T$

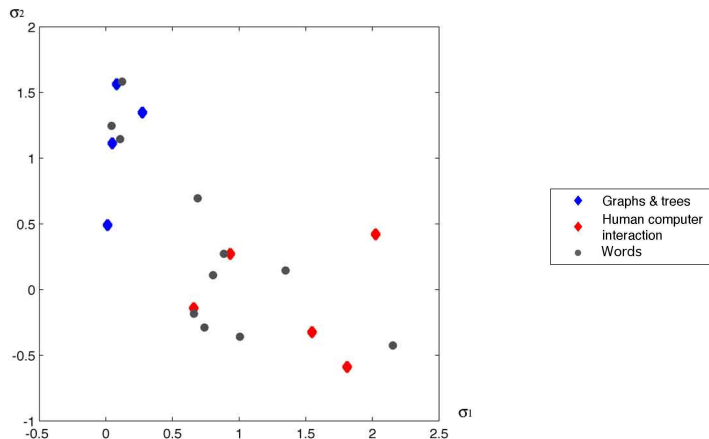


Words in 2-space



Words in 2-space

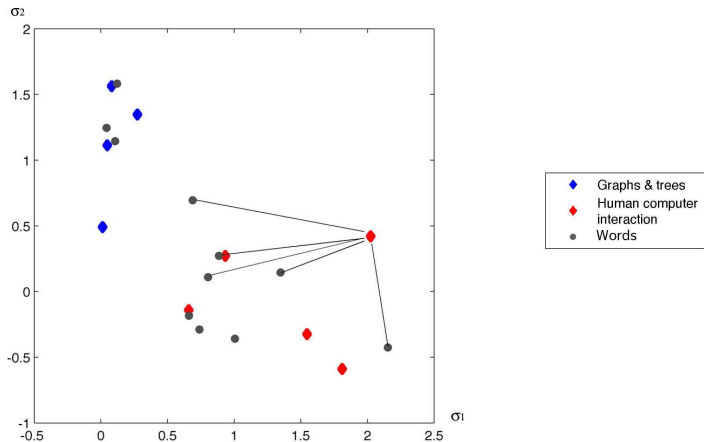
Documents in 2-space



Words and documents in 2-space

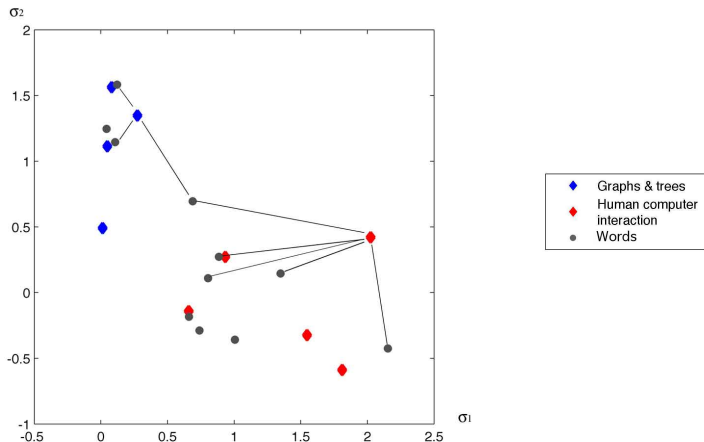
Proximity = relatedness

Spatial Relations



A document and the words it contains

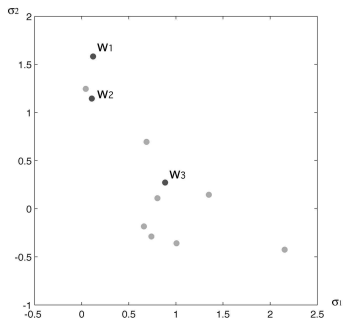
Spatial Relations (2)



Two documents and the words they contain

Comparing Words

Comparing two words: Take the dot product of two row vectors of A_k



$$w_1 =$$

$$[-.06, .34, -.15, -.3, .2, .31, .69, .98, .85]$$

$$w_2 =$$

$$[-.04, .25, -.1, -.21, .15, .22, .5, .71, .62]$$

$$w_3 =$$

$$[.16, .58, .38, .42, .28, .06, .13, .19, .22]$$

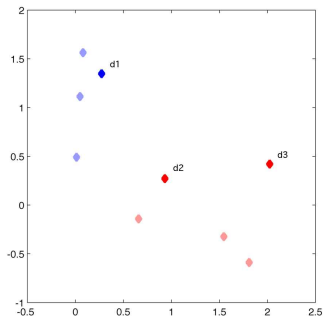
$$w_1 \cdot w_2 = 1.83$$

$$w_1 \cdot w_3 = 0.54$$

$$w_2 \cdot w_3 = 0.41$$

Comparing documents

Comparing documents: Take the dot product of two column vectors of A_k



$$d_1 \cdot d_2 = 1.53$$

$$d_1 \cdot d_3 = 3.42$$

$$d_2 \cdot d_3 = 2.00$$

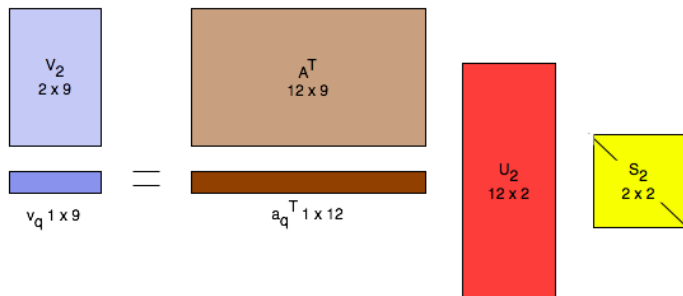
Representing New Documents

- Queries are “pseudo-documents” transformed into document space
- Derive a reduced-rank representation from original term vector

$$v_q = a_q^T U_k S_k^{-1}$$

where a_q is term vector and v_q is the row in V_k for the query

Query Example



- Query = **Human computer** interaction
- $a_q^T = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
- $v_q = [0.14, -0.3]$

Why does truncated SVD work?

- Words do not indicate concepts
- LSI assumes word choice is a noise that can be removed
- Truncated SVD ignores the minor patterns in the data
- Indexing factors replace individual terms

SVD model allows

[DeDuFLH]

- Work on realistically sized problems
- Adjustable dimensionality
- Words and documents represented in same space
- Documents are retrieved directly from query words

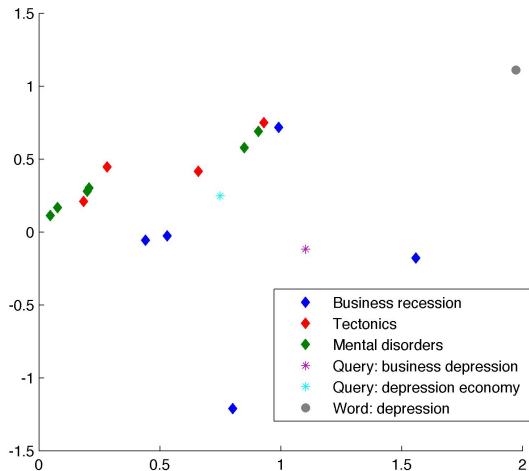
Limitations of LSI with SVD

- Cannot change only some entries in A_k
- Cannot update associations based on feedback
e.g., want economic depression but not geological depressions
- Solution: Riemannian SVD allows perturbations of select entries

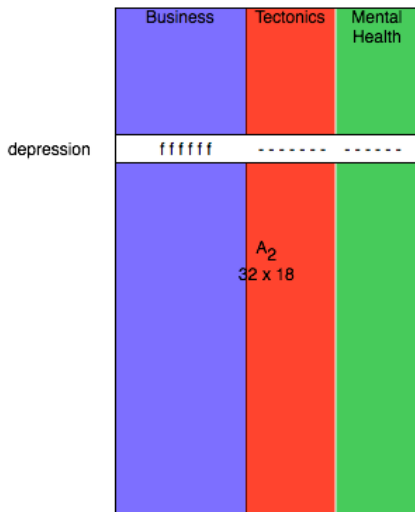
Outline

- 1 Latent Semantic Indexing (LSI)
- 2 Riemannian SVD (RSVD)

Motivation for Updates



Form of Updates



Riemannian SVD

- Solves the structured total linear squares problem
- Similar to SVD but enforces more constraints on the matrix
- Keep some entries the same while changing others

Regular SVD

- As an optimization problem:

$$\min_{B, \text{rank}(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 \text{ subj to } By = 0 \text{ and } y^T y = 1$$

- Solution satisfies:

$$\begin{aligned} Av &= u\sigma, & u^T u &= 1 \\ A^T u &= v\sigma, & v^T v &= 1 \end{aligned}$$

- As an optimization problem:

$$\min_{b,y} \sum (a_i - b_i)^2 \text{ subj to } B(b)y = 0 \text{ and } y^T y = 1$$

$$B(b) = B_0 + b_1 B_1 + \dots + b_q B_q, \quad A = B_0 + \sum_{i=1}^q a_i B_i$$

- For LSI we are interested in low-rank RSVD
- A_k can be written as

$$A_k = A_k^F + A_k^C$$

where A_k^F are the fixed associations and A_k^C are allowed to change

- Optimization problem becomes

$$\min_{B_k, y} \|A_k - B_k\|_F^2 \text{ subj to } B_k y = 0, y^T y = 1, B_k = A_k^F + B_k^C$$

- Optimization:

$$\min_{B_k, y} \|A_k - B_k\|_F^2 \text{ subj to } B_k y = 0, y^T y = 1, B_k = A_k^F + B_k^C$$

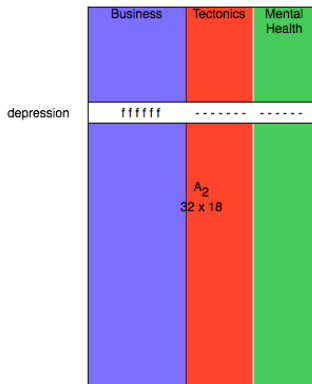
- Solution satisfies:

$$A_k v = D_v u \sigma, \quad u^T D_v u = 1$$

$$A_k^T u = D_u v \sigma, \quad v^T D_u v = 1$$

- D_u and D_v are quadratic in u and v
- P is perturbation matrix: 0's at A_k^F
- $B_k = A_k - \text{diag}(u)P\text{diag}(v)\sigma$

Form of Updates



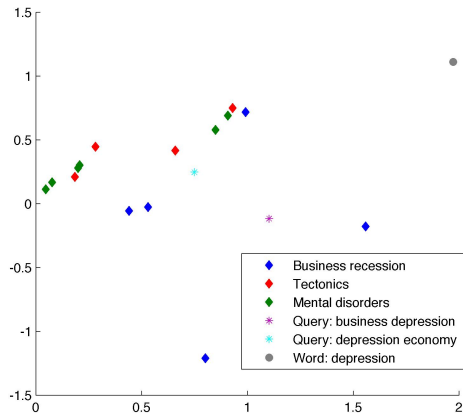
- P is all zeros except for the *depression* row
- $P_{depression} = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$

Steps for using RSVD

- Get k -rank approximation of word-document matrix
- Construct P , determining the set which can change
- Use iterative algorithm to compute RSVD
- Approximate solution yields B_k satisfying constraints

Perturbation example

[Jiang98]



business, depression		economy, depression	
A1	1.00	C3	.96
A7	.99	B2	.96
A5	.99	A4	.96
C3	.77	C2	.95
B2	.76	B4	.93
A4	.75	A1	.90
C2	.74	A5	.84
B4	.71	A7	.84
A2	.65	C6	.82
A3	.65	C1	.81
A6	.52	B1	.76
C6	.52	B3	.74
C1	.50	B5	.74
B1	.43	C4	.69
B3	.40	C5	.67
B5	.40	A3	.28
C4	.34	A2	.28
C5	.31	A6	.28

Perturbation example (2)

business, depression		economy, depression	
A1	1.00	C3	.96
A7	.99	B2	.96
A5	.99	A4	.96
C3	.77	C2	.95
B2	.76	B4	.93
A4	.75	A1	.90
C2	.74	A5	.84
B4	.71	A7	.84
A2	.65	C6	.82
A3	.65	C1	.81
A6	.52	B1	.76
C6	.52	B3	.74
C1	.50	B5	.74
B1	.43	C4	.69
B3	.40	C5	.67
B5	.40	A3	.28
C4	.34	A2	.28
C5	.31	A6	.28

RSVD
→

Perturbed: economy	
A4	.98
A1	.52
A5	.45
A7	.12
A3	.12
A2	.12
A6	.12
C5	-.95
C4	-.95
B3	-.97
B5	-.97
B1	-.99
C1	-.99
C6	-1.00
C3	-1.00
B2	-1.00
C2	-1.00
B4	-1.00

Summary

- LSI attempts to identify underlying semantic structure of a corpus
- Truncated SVD approximation discards minor patterns
- Riemannian SVD allows perturbations to a low-rank matrix

Conclusion

- LSI is about learning a matrix
- Similar to PCA in several ways
- Have an expert choose basis vectors?