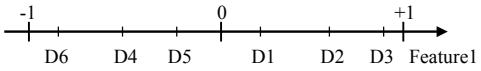
CS630 Representing and Accessing Digital Information

Latent Semantic Indexing

Thorsten Joachims
Cornell University

Low-Dimensional Representation



Problems with Term-Based Representation

- **Text Classification**
  - Too high dimensional for many learning algorithms
- **Information Retrieval**
  - "Vocabulary Mismatch Problem" (car vs. automobile)
  - Query and document use different words
- **General**
  - All terms are assumed orthogonal
  - Need to store high-dimensional (but sparse) vectors

Goal: Find new basis vectors so that each dimension represents an orthogonal concept!

Latent Semantic Indexing

- **Goal:**
  - Make documents (query) similar, even if they do not share any terms
  - Do not use terms as features, but construct (much fewer) new features
  - Find "latent semantic" dimensions of term/document space
- **Assumption**
  - Term co-occurrence reveals semantic information
- **Method:**
  - Begin with term/document matrix
  - Perform singular value decomposition (SVD)
  - Use k (~300) largest singular values to determine new space
  - Map documents into this space using left singular vectors

Co-Occurrence of Terms

	D1	D2	D3	D4	D5	D6
Astronaut	**1**	**0**	**1**	**0**	**0**	**0**
Cosmonaut	**0**	**1**	**0**	**0**	**0**	**0**
Moon	**1**	**1**	**0**	**0**	**0**	**0**
Car	**1**	**0**	**0**	**1**	**1**	**0**
Truck	**0**	**0**	**0**	**1**	**0**	**1**

Example 2

A	D1	D2	D3	D4	D5	D6
W1	1	0	1	0	0	0
W2	0	0	1	0	0	0
W3	1	1	0	0	0	0
W4	0	0	0	1	1	0
W5	0	0	0	1	0	1
W6	0	0	0	0	1	1

- **Frobenius norm of a matrix X with m rows and n columns**

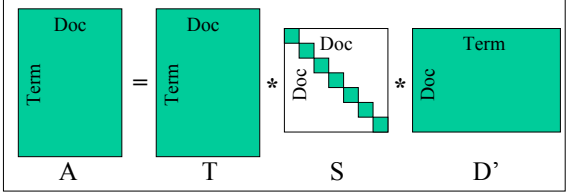
$$||X||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2}$$

Singular Value Decomposition

- **For any matrix A there exist matrices T, S, D so that**

$$A = T * S * D'$$

T and D are orthonormal, S=diag(s_1,s_2,…s_n) is sorted by magnitude (ie. s_1 >= s_2 >= … >= s_n).

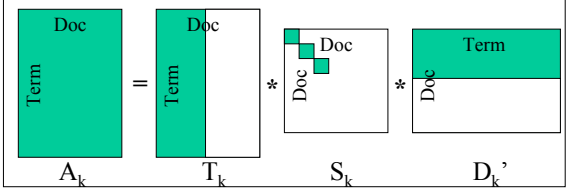


SVD Properties

- **SVD is unique (up to signs T and D)**
- **Setting all but the first k s_i to zero gives best k rank approximation to A (ie. S_k=diag(s_1,s_2,…s_k,0,…,0))**

$$A_k = T * S_k * D'$$

$$min_{rank(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = s_{k+1}^2 + \ldots + s_n^2$$



SVD for Example 2

```
T =         0   -0.7370   -0.3280         0         0   -0.5910
            0   -0.3280   -0.5910         0         0    0.7370
            0   -0.5910    0.7370         0         0    0.3280
      -0.5774         0         0    0.4082    0.7071         0
      -0.5774         0         0    0.4082   -0.7071         0
      -0.5774         0         0   -0.8165         0         0

S = diag(2.0000    1.8019    1.2470    1.0000    1.0000    0.4450)

D =         0   -0.7370    0.3280         0         0   -0.5910
            0   -0.3280    0.5910         0         0    0.7370
            0   -0.5910   -0.7370         0         0    0.3280
      -0.5774         0         0    0.8165         0         0
      -0.5774         0         0   -0.4082    0.7071         0
      -0.5774         0         0   -0.4082   -0.7071         0
```

Rank One Approximation

A_1 =

```
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
```

Rank Two Approximation

A_2 =

```
    0.9787    0.4356    0.7849         0         0         0
    0.4356    0.1938    0.3493         0         0         0
    0.7849    0.3493    0.6294         0         0         0
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
```

Rank Three Approximation

A_3 =

```
    0.8446    0.1938    1.0863         0         0         0
    0.1938   -0.2417    0.8924         0         0         0
    1.0863    0.8924   -0.0479         0         0         0
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
         0         0         0    0.6667    0.6667    0.6667
```

Rank Four Approximation

A_4 =

0.8446	0.1938	1.0863	0	0	0
0.1938	-0.2417	0.8924	0	0	0
1.0863	0.8924	-0.0479	0	0	0
0	0	0	1.0000	0.5000	0.5000
0	0	0	1.0000	0.5000	0.5000
0	0	0	0.0000	1.0000	1.0000

Document Similarities

- **Matrix of all document similarities is**

$$A' * A = (D * S) * (D * S)'$$

- **Matrix of all SVD (rank k) document similarities is**

$$\begin{aligned} A_k' * A_k &= (D * S_k) * (D * S_k)' \\ &= (T_k' * A)' * (T_k' * A) \end{aligned}$$

T_k **is submatrix of T using only the first k columns**

- **Similarity between new document x and new query q**

$$x_k' * q_k = (T_k' * x)' * (T_k' * q)$$

Rank Five Approximation

A_5 =

0.8446	0.1938	1.0863	0	0	0
0.1938	-0.2417	0.8924	0	0	0
1.0863	0.8924	-0.0479	0	0	0
0	0	0	1.0000	1.0000	-0.0000
0	0	0	1.0000	0.0000	1.0000
0	0	0	0.0000	1.0000	1.0000

Rank 2 Document Similarities

A'*A =

2	1	1	0	0	0
1	1	0	0	0	0
1	0	2	0	0	0
0	0	0	2	1	1
0	0	0	1	2	1
0	0	0	1	1	2

$A_2'*A_2$ =

1.7635	0.7849	1.4143	0	0	0
0.7849	0.3493	0.6294	0	0	0
1.4143	0.6294	1.1341	0	0	0
0	0	0	1.3333	1.3333	1.3333
0	0	0	1.3333	1.3333	1.3333
0	0	0	1.3333	1.3333	1.3333

Rank Six (Approximation)

A = A_6 =

1.0000	-0.0000	1.0000	0	0	0
0.0000	0	1.0000	0	0	0
1.0000	1.0000	-0.0000	0	0	0
0	0	0	1.0000	1.0000	-0.0000
0	0	0	1.0000	0.0000	1.0000
0	0	0	0.0000	1.0000	1.0000

Experimental Results

Test Collection	Average Precision	
	LSI	Keyword
Med-e	.66	.51
Med	.52	.46
Cran	.39	.29
ADI	.29	.26
Cisi	.11	.11
News	.61	.55
TM	.40	.35
TREC	.30	.26

Table from S. Dumais

Pros and Cons for LSI

- **Pro:**
  - Can improve retrieval and overcome "vocabulary match" problem
  - Gives lower-dimensional vectors
    - Nice for machine learning algorithms that cannot handle high dimensional spaces
    - Each dimension more "semantic" (?)
- **Contra:**
  - New vectors are dense
    - We do not necessarily save memory
    - Inverted index does not work for dense vectors => less efficient retrieval
  - Words with several meanings confounded by LSI
  - Expensive to compute, but can be done offline

Term Similarities

- **Idea: Terms are similar, if they occur in similar documents**
- **Matrix of all term similarities is**

$$A * A' = (T * S) * (T * S)'$$

- **Matrix of all SVD (rank k) document similarities is**

$$A_k * A_k' = (T * S_k) * (T * S_k)'$$

TOEFL Synonyms

- **Task:**
  - Multiple-choice test for synonym
  - Given one word, find best match out of 4 alternatives
- **Training:**
  - Corpus of 30,473 articles from Grolier's Academic American Encyclopedia
  - Used first ~150 words from each article => 60,768 unique words that occur at least twice
  - 300 singular vectors
- **Result**
  - LSI gets 52.5% correct (corrected for guessing)
  - Non-LSI similarity gets 15.8% (other paper 29.5%) correct
  - Average (foreign) human test taker gets 52.7%

Landauer, T. K. and Dumais, S. T. (1997) A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. *Psychological Review, 104(2)* , 211-240.