Vector Space Model and Latent Semantic Indexing

Jianguo Lu

University of Windsor

December 5, 2013

Table of contents

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Outline

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Binary term-doc incidence matrix

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Slides are from IR book.

CS276: Information Retrieval and Web Search Pandu Nayak and Prabhakar Raghavan Lecture 6: Scoring, Term Weighting and the Vector Space Model

term-doc count matrix

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- count the occurrences of the words in a document;
- each doc is a count vector in N^V;
- Vector representation doesn't consider the ordering of words in a document
- 'John is quicker than Mary' and 'Mary is quicker than John' have the same vectors
- This is called the bag of words model.

Term frequency

tf

The term frequency $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.

- We want to use tf when computing query-document match scores.
- Raw term frequency is not what we want: A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term. But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

log-frequency weight

log weight

The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + log_{10} tf_{t,d}, & \text{if tf>0} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

- 0 –>0, 1–>1, 2–>1.3, 10 –>2, 1000–> 4, etc.

score of a query

Score for a document-query pair: sum over terms t in both q and d:

$$score = \sum_{t \in q \cap d} (1 + log_{10} tf_{t,d}) \quad (2)$$

The score is 0 if none of the query terms is present in the document.

Document frequency

rare terms

- Rare terms are more informative than frequent terms
- Recall stop words
- Consider a term in the query that is rare in the collection (e.g., arachnocentric)
- A document containing this term is very likely to be relevant to the query arachnocentric
- We want a high weight for rare terms like arachnocentric.

frequent terms

- Frequent terms are less informative than rare terms
- Consider a query term that is frequent in the collection (e.g., high, increase, line)
- A document containing such a term is more likely to be relevant than a document that doesn't
- But it's not a sure indicator of relevance.
- For frequent terms, we want high positive weights for words like high, increase, and line
- But lower weights than for rare terms.
- We will use document frequency (df) to capture this.

idf weight

idf

We define the idf (inverse document frequency) of t by

$$idf_t = log_{10} \frac{N}{df_t} \tag{3}$$

where N is the number of documents

- df_t is the document frequency of t: the number of documents that contain t
- df_t is an inverse measure of the informativeness of t
- We use $log(N/df_t)$ instead of N/dft to dampen the effect of idf.

Suppose that N=1,000,000.

term	df	idf
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effects of idf ranking

- Does idf have an effect on ranking for one-term queries, like iPhone?
- idf has no effect on ranking one term queries
- idf affects the ranking of documents for queries with at least two terms
- For the query ‘capricious person’, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

tf-idf

tf-idf weight

The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = log_{10}(1 + tf_{t,d}) log_{10}(N/df_t) \quad (4)$$

- Best known weighting scheme in information retrieval
- Note: the - in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

score for a document given a query

$$score(q, d) = \sum_{t \in q \cap d} tfidf_{t,d} \tag{5}$$

- There are many variants
- How 'tf' is computed (with/without logs)
- Whether the terms in the query are also weighted
- ...

Binary ->count ->weight matrix

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	1	

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*	...
Antony	157	73	0	0	0	0	
Brutus	4	157	0	1	0	0	
Caesar	232	227	0	2	1	1	
Calpurnia	0	10	0	0	0	0	
Cleopatra	57	0	0	0	0	0	
mercy	2	0	3	5	5	1	
worser	2	0	1	1	1	1	

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*	...
Antony	5.25	3.18	0	0	0	0.35	
Brutus	1.21	6.1	0	1	0	0	
Caesar	8.59	2.54	0	1.51	0.25	0	
Calpurnia	0	1.54	0	0	0	0	
Cleopatra	2.85	0	0	0	0	0	
mercy	1.51	0	1.9	0.12	5.25	0.88	
worser	1.37	0	0.11	4.15	0.25	1.95	
...							

- Note that the last matrix is not calculated from the second one. It is from a bigger matrix containing more data items. The last matrix is just an illustrative example.

matlab code to calculate tfidf


```
clear all; format bank

A =[157  73 0 0 0 1
    4   157 0 2 0 0
    232 227 0 2 1 0
    0    10 0 0 0 0
    57   0  0 0 0 0
    2    0  3 8 5 8
    2    0  1 1 1 5];

A_binary=ones-isnan(A./A)
df=sum(A_binary')

A_tf=1+log(A)

N=size(A,2);
M=size(A,1);
for i=1:M
    for j=1:N
        if isinf(A_tf(i,j))
            A_tf(i,j)=0;
        end
    end
end
A_tf
idf=diag(log10(6./df))
B=A_tf'*idf;
tfidf=B'


df =   3     3     4     1     1     5     5
log10(6./df)=  0.3010    0.3010    0.1761    0.7782    0.7782    0.0792    0.0792
```

tfidf calculated from matrix A

$$A = \begin{pmatrix} 157 & 73 & 0 & 0 & 0 & 1 \\ 4 & 157 & 0 & 2 & 0 & 0 \\ 232 & 227 & 0 & 2 & 1 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 57 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 8 & 5 & 8 \\ 2 & 0 & 1 & 1 & 1 & 5 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$tfidf = \begin{pmatrix} 1.82 & 1.59 & 0 & 0 & 0 & 0.30 \\ 0.72 & 1.82 & 0 & 0.51 & 0 & 0 \\ 1.14 & 1.13 & 0 & 0.30 & 0.18 & 0 \\ 0 & 2.57 & 0 & 0 & 0 & \\ 3.92 & 0 & 0 & 0 & 0 & 0 \\ 0.13 & 0 & 0.17 & 0.24 & 0.21 & 0.24 \\ 0.13 & 0 & 0.08 & 0.08 & 0.08 & 0.21 \end{pmatrix}$$

documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors - most entries are zero.

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

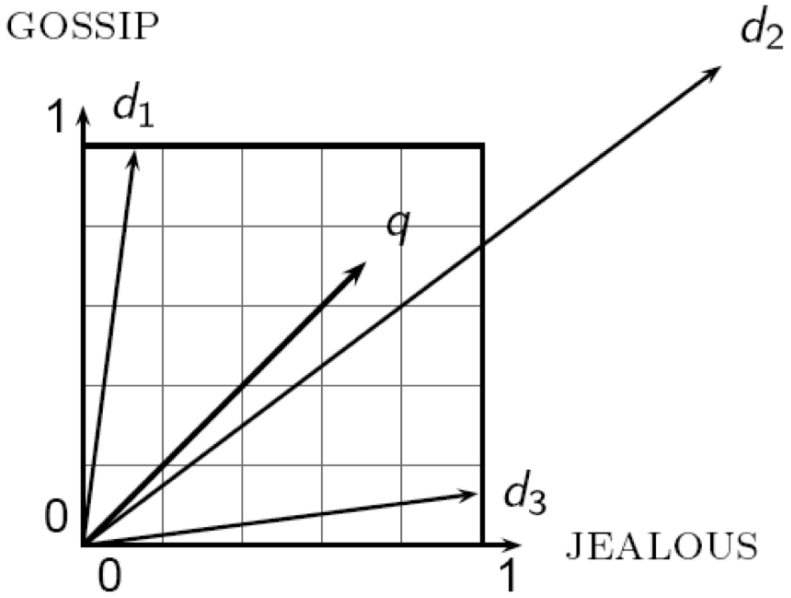
Queries as vectors

- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors
- proximity $\approx$ inverse of distance
- Recall: We do this because we want to get away from the you're-either-in-or-out Boolean model.
- Instead: rank more relevant documents higher than less relevant documents

Formalizing vector space proximity

- First cut: distance between two points
- (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- because Euclidean distance is large for vectors of different lengths.

The Euclidean distance between q and d2 is large even though the distribution of terms in the query q and the distribution of terms in the document d2 are very similar.



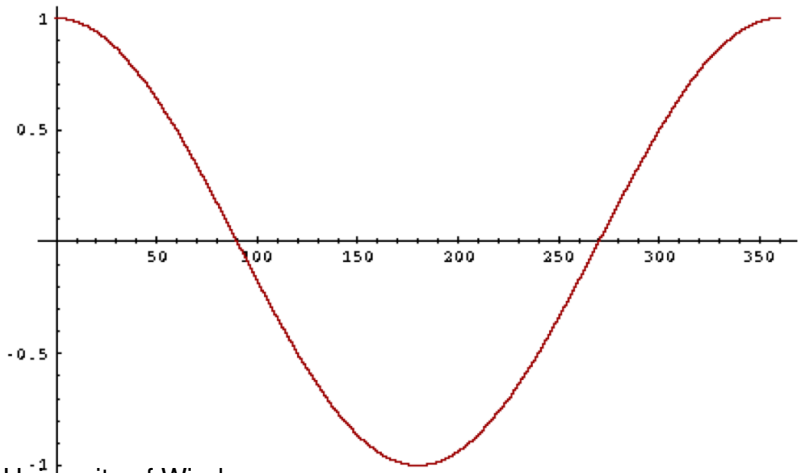
term	d1	d2	d3	q
jealous	0.06	1.85	1.00	0.7
gossip	1.0	1.30	0.06	0.7

Use angle instead of distance

The following two notions are equivalent

1. Rank documents in decreasing order of the angle between query and document
2. Rank documents in increasing order of cosine(query,document)
3. Cosine is a monotonically decreasing function for the interval $[0^o, 180^o]$

- Thought experiment: take a document d and append it to itself. Call this document d'.
- 'Semantically' d and d' have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.
- Key idea: Rank documents according to angle with query.



Length normalization

L_2 norm

$$||x||_2 = \sqrt{\sum_i x_i^2} \tag{6}$$

- A vector can be (length-) normalized by dividing each of its components by its length (L2 norm)
- Dividing a vector by its L2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d' (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
- Long and short documents now have comparable weights

cosine similarity

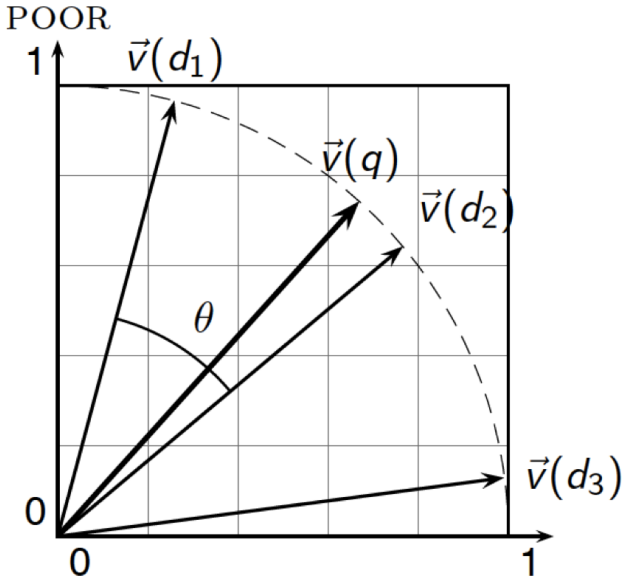
cosine similarity of q and d

$$cos(q,d) = \frac{q \cdot d}{|q||d|} = \frac{q}{|q|} \cdot \frac{d}{|d|} = \frac{\sum q_i d_i}{\sqrt{\sum q_i^2}\sqrt{\sum d_i^2}} \quad (7)$$

- q_i: tfidf weight of term i in the query;
- d_i: tfidf weight of term i in the document;
- the cosine of the angle between q and d.

for length normalized vectors

$$cos(q,d) = q \cdot d = \sum q_i \cdot d_i \quad (8)$$



cos similarity example

How similar are the novels. Note: To simplify this example, we don't do idf weighting.

- SaS: Sense and Sensibility
- PaP: Pride and Prejudice, and
- WH: Wuthering Heights

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

Table: log-frequency weighting

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

Table: length normalization

$$cos(SaS, PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \quad (9)$$

$$\approx 0.94 \quad (10)$$

$$cos(SaS, WH) \approx 0.79 \quad (11)$$

$$cos(PaP, WH) \approx 0.69 \quad (12)$$

Variants of tfidf

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+...+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Outline

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Problem with vector space model

Synonym

- Different terms may have identical or similar meanings (weaker: words indicating the same topic).
- No associations between words are made in the vector space representation.

Polysemy

- Polysemy: Words often have a multitude of meanings and different types of usage (more severe in very heterogeneous collections).
- VSM unable to discriminate between different meanings of the same word.

Term-document matrices are very large

- But the number of topics that people talk about is small (in some sense)
- Clothes, movies, politics, ...
- Can we represent the term-document space by a lower dimensional latent space?

Solution: LSI (Latent Semantic Indexing)

LSI

- Perform a low-rank approximation of document-term matrix (typical rank 100-300)
- Map documents (and terms) to a low-dimensional representation.
- Design a mapping such that the low-dimensional space reflects semantic associations (latent semantic space).
- Compute document similarity based on the inner product in this latent semantic space

How?

- From term-doc matrix A, we compute the approximation Ak.
- There is a row for each term and a column for each doc in Ak
- Thus docs live in a space of k«r dimensions
- These dimensions are not the original axes

LSI goals

- Similar terms map to similar location in low dimensional space
- Noise reduction by dimension reduction

- Each row and column of A gets mapped into the k-dimensional LSI space, by the SVD.
- Claim: this is not only the mapping with the best (Frobenius error) approximation to A, but in fact improves retrieval.
- A query q is also mapped into this space, by

$$q = q^T U \Sigma^{-1} \tag{13}$$

- Query NOT a sparse vector.

Outline

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Length

If $v = (v_1, v_2, \ldots, v_n)$ is a vector, its length |v| is defined as

$$|v| = \sqrt{\sum_{1}^{n} v_i^2} \tag{14}$$

Inner product

Given two vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. The inner product of x and y is

$$x \cdot y = \sum_{1}^{n} x_i y_i \tag{15}$$

Orthonmal Vectors

Orthogonal vector

- Two vectors are orthogonal to each other if their inner product equals zero.
- In two dimensional space, this is equivalent to saying that the vectors are perpendicular, or that the only angle between them is a 90 angle.

For example, the vectors [2, 1,-2, 4] and [3,-6, 4, 2] are orthogonal because

$$(2, 1, -2, 4) \cdot (3, -6, 4, 2) = 2 \times 3 + 1 \times (-6) - 2 \times 4 + 4 \times 2 = 0 \qquad (16)$$

Normal vector

A normal vector (or unit vector) is a vector of length 1.
Any vector with an initial length greater than 0 can be normalized by dividing each component in it by the vector's length.

Orthonormal Vectors

Orthogonal and normal vectors

Identity Matrix I

The identity matrix (denoted as I) is a square matrix with entries on the diagonal equal to 1 and all other entries equal zero.

A matrix A is orthogonal if $AA^T = A^TA = I$.

Outline

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Eigenvalues and Eigenvectors

Given an $m \times m$ square matrix S. An eigenvector v is a nonzero vector that satisfes the equation

$$Sv = \lambda v \tag{17}$$

λ is a scalar, called an eigenvalue.

$$S = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}$$

$$\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 7 \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{18}$$

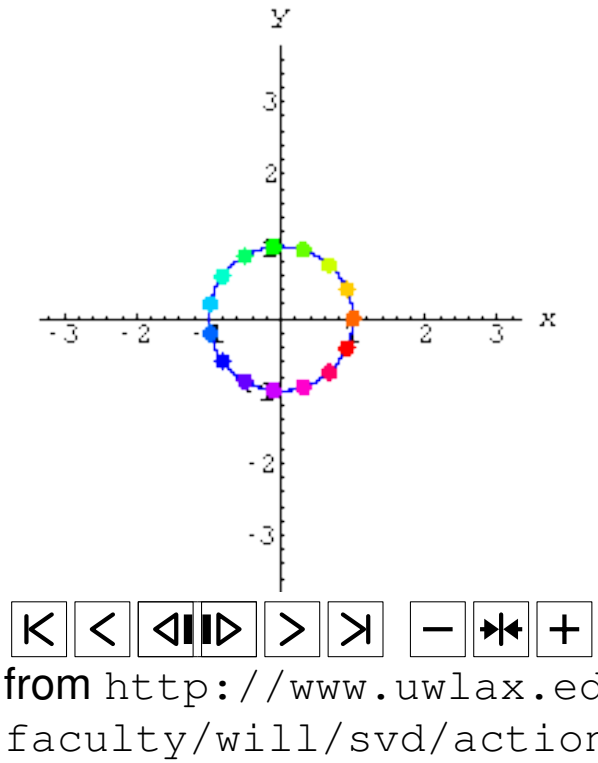
Normalized solution:

$$\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{pmatrix} = 7 \begin{pmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{pmatrix} \tag{19}$$

- Intuition: v is unchanged by S (except for scaling)
- Example: stationary distribution of a Markov chain
- There are at most m distinct solutions. can be complex even though S is real.

Matrix action

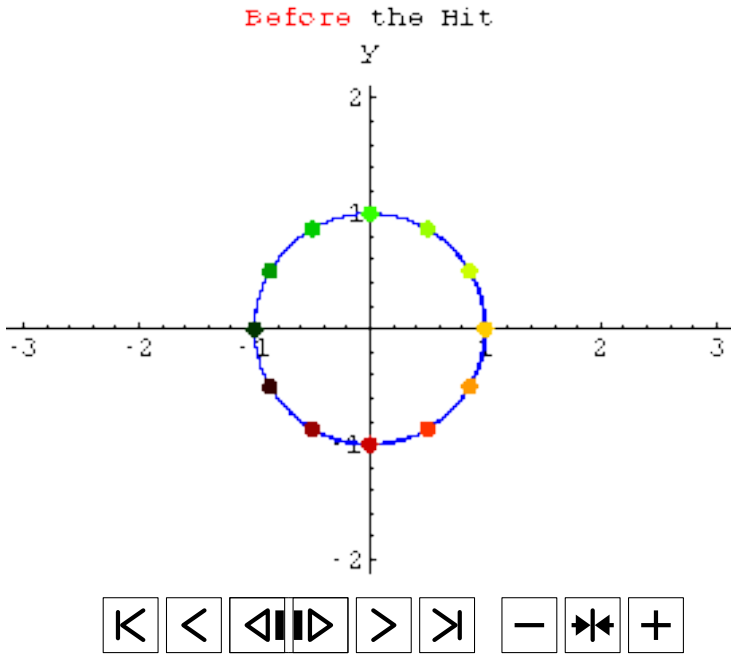
- The plot shows some color coded points on the unit circle.
- Move your mouse over the plot to see what happens when the matrix hits the points on the unit circle.
- You can see that the matrix stretches and rotates the unit circle–that's Matrix Action.
  $M = \begin{pmatrix} 1 & 3 \\ -3 & 2 \end{pmatrix}$
- e.g., $M\binom{1}{0} = \binom{1}{-3}$



from `http://www.uwlax.edu/faculty/will/svd/action/`

Matrix action-Strecher

- When you look at that action you can see why it's natural to call a diagonal matrix a "stretcher" matrix.
- The diagonal matrix stretches in the x direction by a factor of "a" and in the y direction by a factor of "b".
- You can verify this by hand using the column way to multiply a matrix times a vector:
- $M = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$
- e.g., $M\binom{1}{0} = \binom{3}{0}$



Another example for eigenvectors

$$S = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

its eigenvalues are 30, 20, 1, and the corresponding eigenvectors are

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$Sv_1 = \lambda_1 v_1$$
$$Sv_2 = \lambda_2 v_2$$
$$Sv_3 = \lambda_3 v_3$$

$$S(v_1, v_2, v_3) = (\lambda_1 v_1, \lambda_2 v_2, \lambda_3 v_3) \qquad = (v_1, v_2, v_3) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

$$SU = U\Sigma$$
$$S = U\Sigma U^{-1}$$

Find solutions for eigenvalues and eigen vectors

$$Sv = \lambda v \tag{20}$$

$$(S - \lambda I)v = 0 \tag{21}$$

$$\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} - \lambda I = \begin{pmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{pmatrix} \tag{22}$$

The determinant is set 0, i.e.,

$$(3 - \lambda)(6 - \lambda) - 4 = 0$$

There are two solutions:

$$\lambda = 7, \lambda = 2$$

For the principal eigenvalue 7, the corresponding eigenvector can be solved from

$$\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 7 \begin{pmatrix} x \\ y \end{pmatrix} \tag{23}$$

The normalized principal eigenvector is $\begin{pmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{pmatrix}$ The eigenvector corresponding to eigenvalue 2 is $\begin{pmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{pmatrix}$.

Matrix of eigenvectors

there are n eigenvectors. Let U is an $n \times n$ matrix consists of n column eigenvectors. U is orthonormal, i.e.,

$$UU^T = U^T U = I$$

For example,

$$U = \begin{pmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5} \end{pmatrix}.$$

$$UU^T = \begin{pmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5} \end{pmatrix} \begin{pmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ 2/\sqrt{5} & -1/\sqrt{5} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Finding eigen pairs by power iteration

- Recall the 'power method' for PageRank–calculate the principal eigenvector for stochastic matrix.
- the method can be extended for other eigenvectors
- when the principal eigenvector is obtained, modify the matrix so that in the new matrix, the principle eigenvector is the second eigenvector

More details can be found in MMD.

$$M = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}$$

Iteration 1: $\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \end{pmatrix}$

Iteration 2: $\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 0.530 \\ 0.848 \end{pmatrix} = \begin{pmatrix} 3.286 \\ 6.148 \end{pmatrix}$

Iteration 3: $\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 0.471 \\ 0.882 \end{pmatrix} = \begin{pmatrix} \dots \\ \dots \end{pmatrix}$

Eigenvalues and eigenvectors for column stochastic matrix

Recall the PageRank algorithm $Mr = r$.

$$\begin{pmatrix} .5 & .2 \\ .5 & .8 \end{pmatrix} - \lambda I = \begin{pmatrix} .5-\lambda & .2 \\ .5 & .8-\lambda \end{pmatrix}$$

The determinant is set 0, i.e.,

$$(.5-\lambda)(.8-\lambda) - .1 = 0$$

There are two solutions:

$$\lambda = 1, \lambda = 0.3$$

For the principal eigenvalue 1, the corresponding eigenvector can be solved from

$$\begin{pmatrix} 0.5 & 0.2 \\ 0.5 & 0.8 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{24}$$

The normalized principal eigenvector is $\begin{pmatrix} 2/\sqrt{29} \\ 5/\sqrt{29} \end{pmatrix}$

Outline

1 Vector Space Model

2 LSI

3 Matrix and Vectors

4 Eigenvalues and Eigenvectors

5 Matrix Decomposition

Singular value decomposition

$$M = U\Sigma V^T$$

Example:

- U: document-to-concept similarity matrix
- V: term-to-concept similarity matrix
- Σ: its diagonal elements: “strength" of each concept

Eigen diagonal decomposition

Matrix diagonalization theorem

there is an eigen decomposition

$$S = U\Sigma U^{-1} \tag{25}$$

- Columns of U are the eigenvectors of S;
- Diagonal elements of Σ are eigenvalues of S

$$S(v_1, v_2, v_3) = (\lambda_1 v_1, \lambda_2 v_2, \lambda_3 v_3) \qquad = (v_1, v_2, v_3) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \tag{26}$$

$$SU = U\Sigma$$

$$S = U\Sigma U^{-1}$$

Symmetric Eigen Decomposition

Perron-Frobenius Theorem

If S is symmetric non-negative matrix, there is an unique eigen decomposition

$$S = U\Sigma U^T \tag{27}$$

- U is orthogonal;
- eigenvalues are non-negative;
- $U^{-1} = U^T$
- Columns of U are normalized eigenvectors
- Columns are orthogonal.
- (everything is real)

Singular Value Decomposition

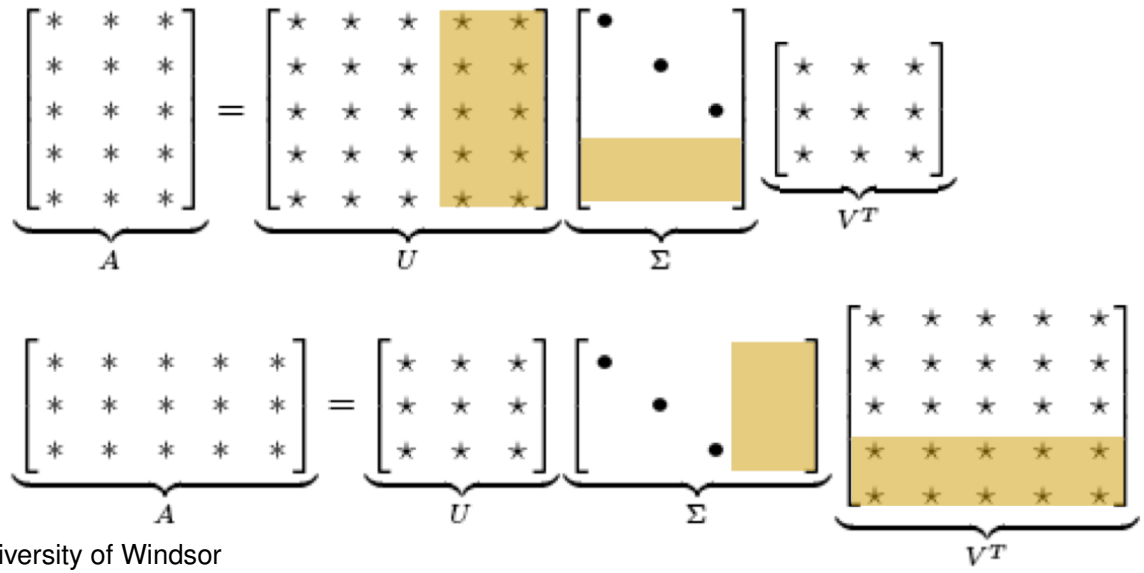
THEOREM [Press+92]

Always possible to decompose matrix A into

$$A = U\Sigma V^T, \tag{28}$$

where

- $A : m \times n; U : m \times r; \Sigma : r \times r; V : n \times r;$
- U, V: column orthonormal (ie., columns are unit vectors, orthogonal to each other)
- $U^T U = I; V^T V = I$ (I: identity matrix)
- Σ: singular are positive, and sorted in decreasing order



SVD Interpretation: documents, terms and concepts

How to compute U, V, Σ?
Note that U and V are orthonormal, so $UU^T = I, VV^T = I$.

$$A = U\Sigma V^T$$
$$A^T = (U\Sigma V^T)^T = V\Sigma^T U^T$$
$$A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T$$

$$A^T AV = V\Sigma^2 V^T V$$
$$A^T AV = V\Sigma^2$$

V is the eigenvectors of $A^T A$, Σ is the square root of the eigenvalues.
Similarly,

$$AA^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T$$
$$(AA^T)U = U\Sigma^2$$

- Why $r \times r$?
- decided by the 'rank' r of the matrix A.
- there are r number of non-zero eigenvalues.

Documents, Terms, and Concepts

- Q: if A is the document-to-term matrix, what is $A^T A$?
- term-to-term ([m x m]) similarity matrix
- Q: AA^T ?
- document-to-document ([n x n]) similarity matrix

$$A = \begin{pmatrix} 2 & 0 & 8 & 6 & 0 \\ 1 & 6 & 0 & 1 & 7 \\ 5 & 0 & 7 & 4 & 0 \\ 7 & 0 & 8 & 5 & 0 \\ 0 & 10 & 0 & 0 & 7 \end{pmatrix}$$

$$AA^T = \begin{pmatrix} 104 & 8 & 90 & 108 & 0 \\ 8 & 87 & 9 & 12 & 109 \\ 90 & 9 & 90 & 111 & 0 \\ 108 & 12 & 111 & 138 & 0 \\ 0 & 109 & 0 & 0 & 149 \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 79 & 6 & 107 & 68 & 7 \\ 6 & 136 & 0 & 6 & 112 \\ 107 & 0 & 177 & 116 & 0 \\ 68 & 6 & 116 & 78 & 7 \\ 7 & 112 & 0 & 7 & 98 \end{pmatrix}$$

LSI: Latent Semantic Indexing

- This matrix is the basis for computing the similarity between documents and queries.
- Can we transform this matrix, so that we get a better measure of similarity between documents and queries?

	Antony&Cleopatra	*JuliusCaesar*	*TheTempest*	*Hamlet*	*Othello*	*Macbeth*
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Overview of SVD (Singular Value Decomposition

- Decompose the term-document matrix into a product of matrices.
- Singular value decomposition (SVD).
- $A = U\Sigma V^T$ (where A = term-document matrix)
- We will then use the SVD to compute a new, improved term-document matrix A'.
- We'll get better similarity values out of A' (compared to A).
- Using SVD for this purpose is called latent semantic indexing or LSI.

Term-doc Matrix

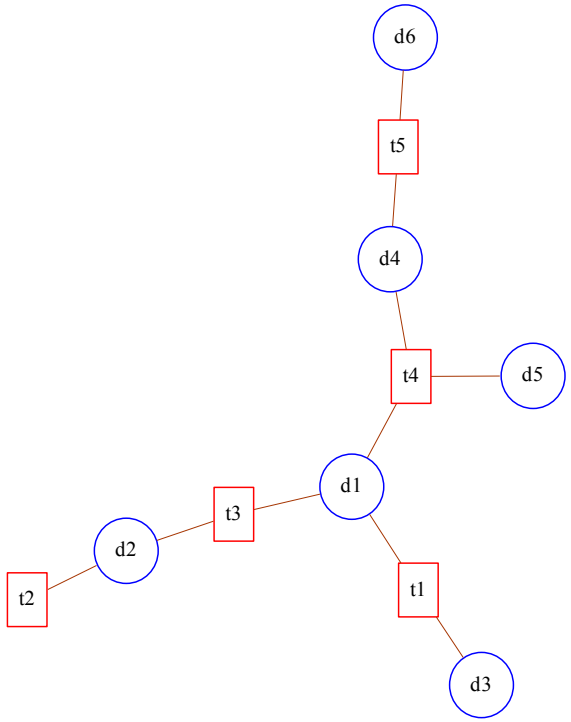
Example from the IR book.

Let C=

	d1	d2	d3	d4	d5	d6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

- This is a standard term-document matrix. We use a non-weighted matrix here to simplify the example.
- Use Matlab to calculate the decomposition

[U, S, V]=svd(C)



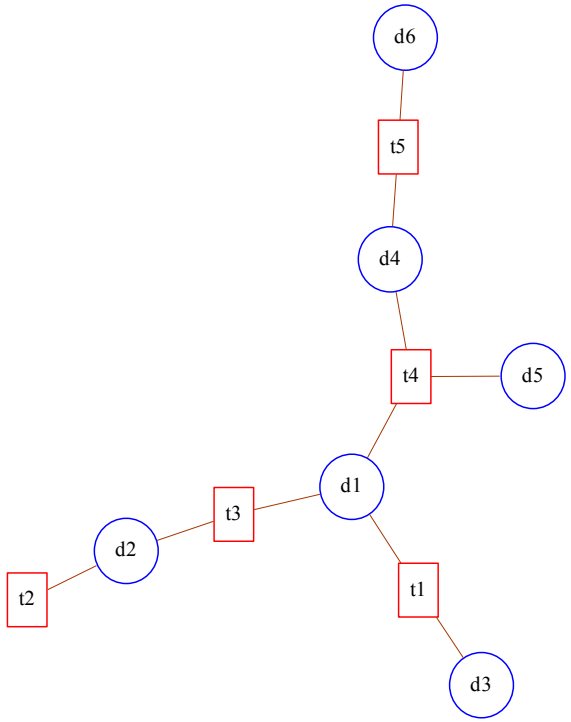
$C * C'$ and $C^T * C$

term-term similarity:

$$C * C^T = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 1 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

doc-doc similarity:

$$C^T * C = \begin{pmatrix} 3 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$



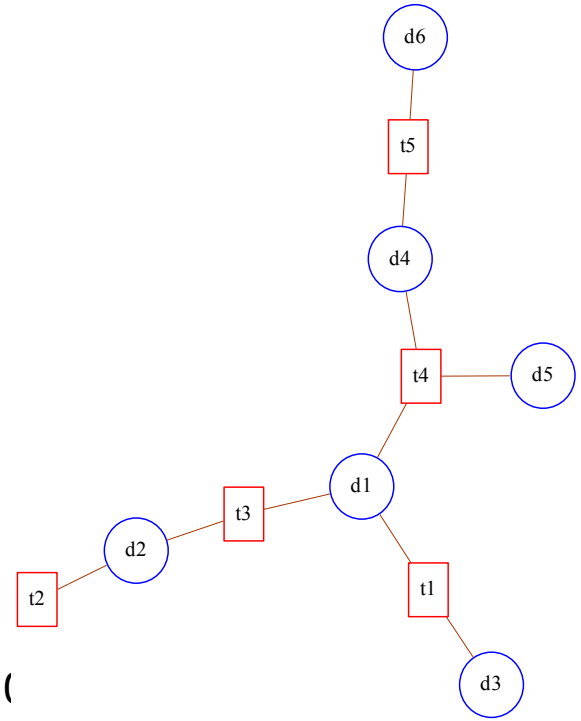
$C = U\Sigma V^T$

$$C = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 0.44 & -0.29 & -0.56 & 0.57 & -0.24 \\ 0.12 & -0.33 & 0.58 & -0.00 & -0.72 \\ 0.47 & -0.51 & 0.36 & -0.00 & 0.61 \\ 0.70 & 0.35 & -0.15 & -0.57 & -0.15 \\ 0.26 & 0.64 & 0.41 & 0.57 & 0.08 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 2.16 & 0 & 0 & 0 & 0 \\ 0 & 1.59 & 0 & 0 & 0 \\ 0 & 0 & 1.27 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 0.39 \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0.74 & -0.28 & -0.27 & -0.00 & 0.52 & -(\\ 0.27 & -0.52 & 0.74 & 0.00 & -0.28 & 0.00 \\ 0.20 & -0.18 & -0.44 & 0.57 & -0.62 & 0.00 \\ 0.44 & 0.62 & 0.20 & 0.00 & -0.18 & -0.57 \\ 0.32 & 0.21 & -0.12 & -0.57 & -0.40 & 0.57 \\ 0.12 & 0.40 & 0.32 & 0.57 & 0.21 & 0.57 \end{pmatrix}$$



$C = U\Sigma V^T$

$$U = \begin{pmatrix} 0.4403 & -0.2962 & -0.5695 & 0.5774 & -0.2464 \\ 0.1293 & -0.3315 & 0.5870 & -0.0000 & -0.7272 \\ 0.4755 & -0.5111 & 0.3677 & -0.0000 & 0.6144 \\ 0.7030 & 0.3506 & -0.1549 & -0.5774 & -0.1598 \\ 0.2627 & 0.6467 & 0.4146 & 0.5774 & 0.0866 \end{pmatrix}$$

- one row per term
- one column per 'concept'. at most there are min(M,N) of columns. M is the number of terms and N is the number of documents.
- This is an orthonormal matrix:
  - Row vectors have unit length.
  - Any two distinct row vectors are orthogonal to each other.
- Think of the dimensions as "semantic" dimensions that capture distinct topics like politics, sports, economics.
- Each number u_{ij} in the matrix indicates how strongly related term i is to the topic represented by semantic dimension j .

Σ in $C = U\Sigma V^T$

$$\Sigma = \begin{pmatrix} 2.1625 & 0 & 0 & 0 & 0 \\ 0 & 1.5944 & 0 & 0 & 0 \\ 0 & 0 & 1.2753 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0.3939 \end{pmatrix}$$

- This is a square, diagonal matrix of dimensionality $min(M, N) \times min(M, N)$.
- The diagonal consists of the singular values of C.
- The magnitude of the singular value measures the importance of the corresponding semantic dimension.
- We will make use of this by omitting unimportant dimensions.

V^T in $C = U\Sigma V^T$

$$V^T = \begin{pmatrix} 0.7486 & -0.2865 & -0.2797 & -0.0000 & 0.5285 & -0.0000 \\ 0.2797 & -0.5285 & 0.7486 & 0.0000 & -0.2865 & 0.0000 \\ 0.2036 & -0.1858 & -0.4466 & 0.5774 & -0.6255 & 0.0000 \\ 0.4466 & 0.6255 & 0.2036 & 0.0000 & -0.1858 & -0.5774 \\ 0.3251 & 0.2199 & -0.1215 & -0.5774 & -0.4056 & 0.5774 \\ 0.1215 & 0.4056 & 0.3251 & 0.5774 & 0.2199 & 0.5774 \end{pmatrix}$$

- One column per document,
- one row per min(M,N) where M is the number of terms and N is the number of documents.
- This is an orthonormal matrix: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.
- These are again the semantic dimensions from the term matrix U that capture distinct topics like politics, sports, economics.
- Each number v_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension j .

Summary of LSI

- We have decomposed the term-document matrix C into a product of three matrices.
- The term matrix U: consists of one (row) vector for each term
- The document matrix V^T: consists of one (column) vector for each document
- The singular value matrix Σ: diagonal matrix with singular values, reflecting importance of each dimension
- Next: Why are we doing this?

Dimensionality reduction

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the ‘details’.
- These details may
  - be noise– in that case, reduced LSI is a better representation because it is less noisy
  - make things dissimilar that should be similar–again reduced LSI is a better representation because it represents similarity better.
- Analogy for “fewer details is better"
  - Image of a bright red flower
  - Image of a black and white flower
  - Omitting color makes is easier to see similarity

Reduce the dimensionality to 2

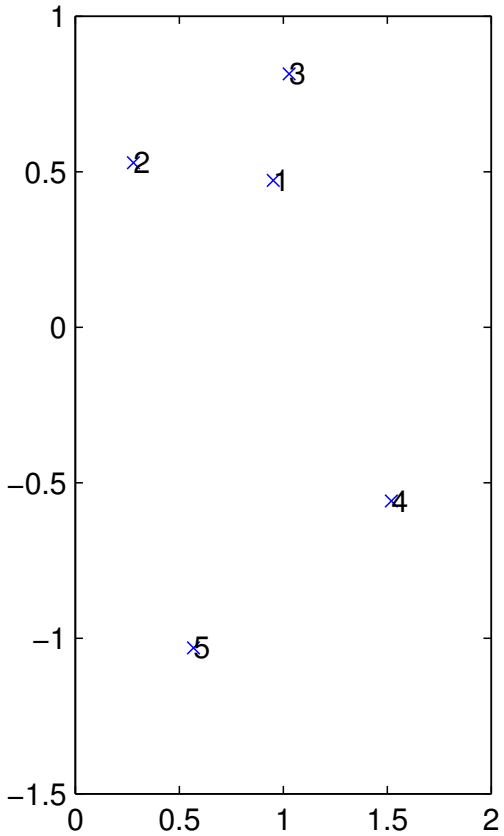
- we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V T to zero when computing the product $C = U\Sigma V^T$.

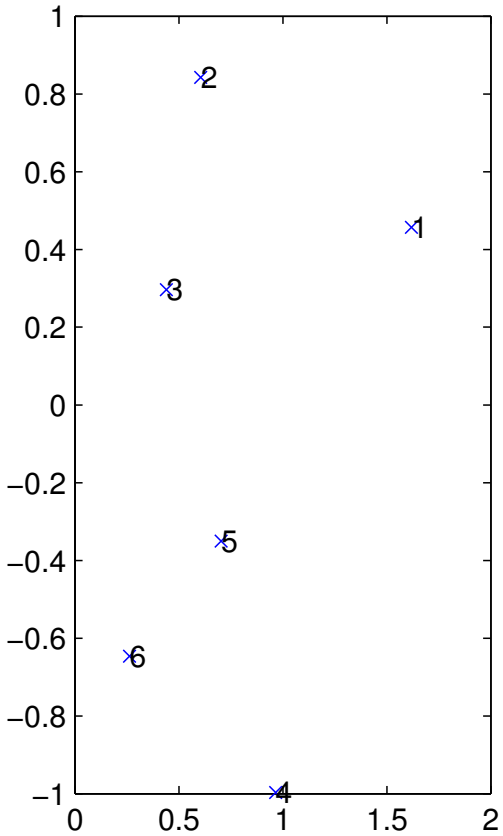
U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

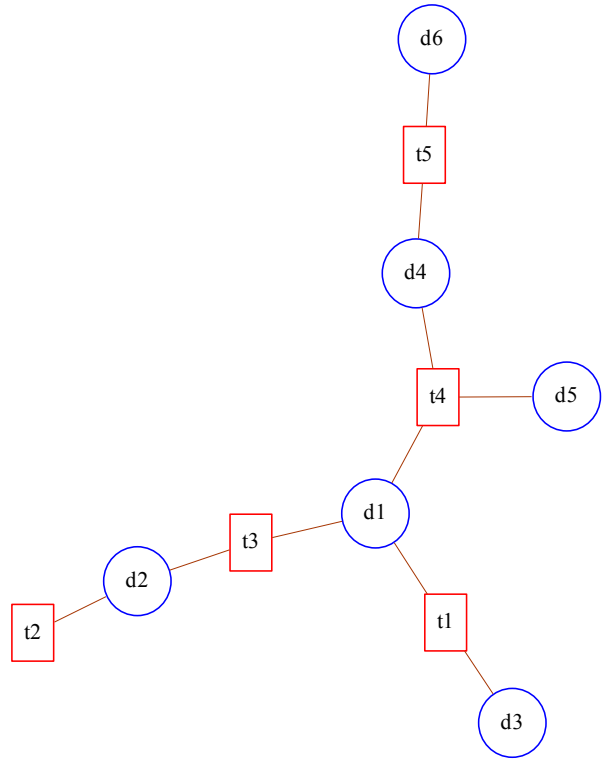
Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

2D visualization







1	U	0.44	0.30
2		0.13	0.33
3		0.48	0.51
4		0.70	-0.35
5		0.26	-0.65

1	U*S	0.95	0.47
2		0.28	0.53
3		1.03	0.81
4		1.52	-0.56
5		0.57	-1.03

C vs. C2

- We can view C2 as a two-dimensional representation of the matrix.
- We have performed a dimensionality reduction to two dimensions.
- Similarity of d2 and d3 in the original space: 0.
- Similarity of d2 and d3 in the reduced space:

$$0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$$

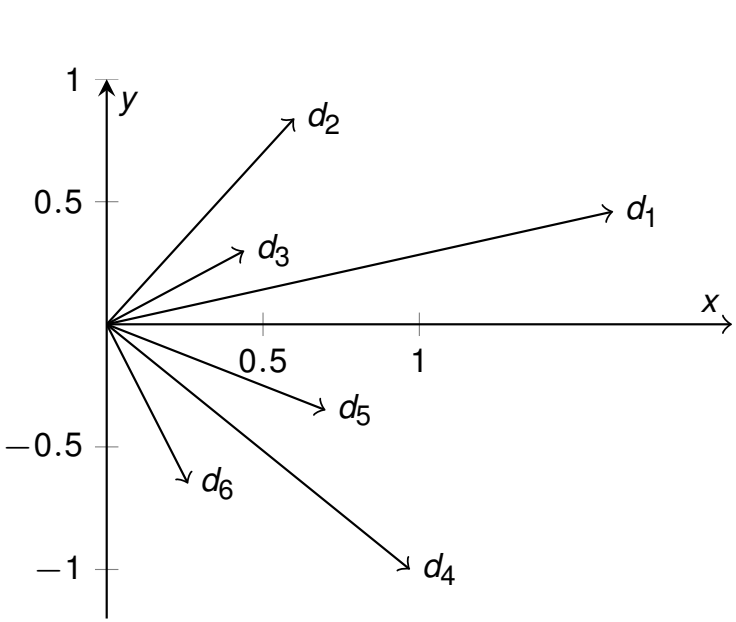
C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

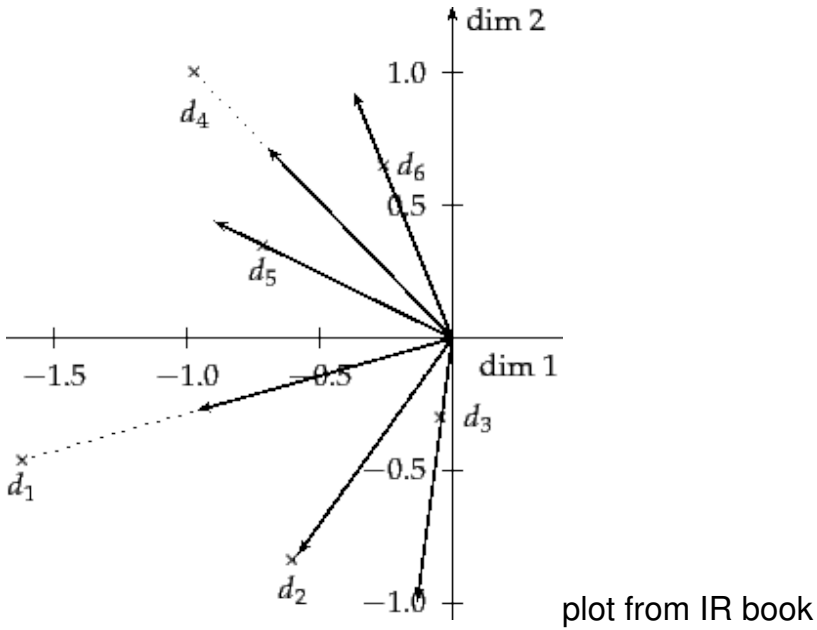
C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

cosine distance in 2 dimensions

$\Sigma_2 V_2^T =$

1.62	0.60	0.44	0.97	0.70	0.26
0.46	0.84	0.30	−1.00	−0.35	−0.65





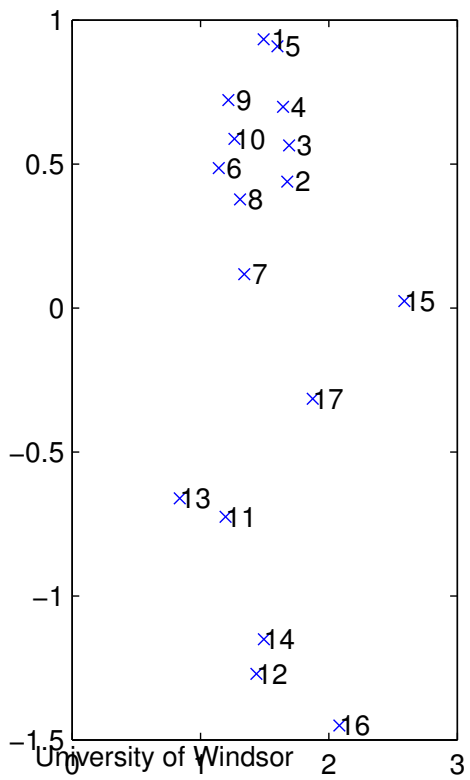
plot from IR book

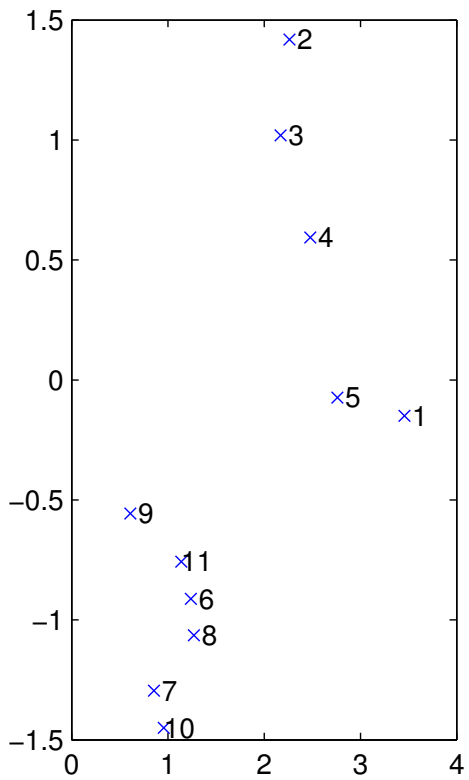
Example for 2D plot

$$\begin{matrix}
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0
\end{matrix}
\qquad
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

$$\begin{matrix}
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0
\end{matrix}
\qquad
\begin{matrix}
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1
\end{matrix}$$

$$\lambda_1 = 6.48, \lambda_2 = 3.17, \lambda_3 = 2.83, \ldots$$





Listing 1: Matlab code

```
1  [u,s,v]=svds(A,2);
2  dots=(u*s)'
3  x=dots(1, :);
4  y=dots(2,:);
5  m=size(A,2);
6  n=size(A,1);
7  subplot(1,2,1);
8  plot(x,y ,'x')
9  a=num2str([1:n]');
10 text(x,y,cellstr(a));
```

Summary

- LSI takes documents that are semantically similar (= talk about the same topics), . . .
- . . . but are not similar in the vector space (because they use different words) . . .
- . . . and re-represents them in a reduced vector space . . .
- . . . in which they have higher similarity.
- Thus, LSI addresses the problems of synonym and semantic relatedness.
- Standard vector space: Synonyms contribute nothing to document similarity.
- Desired effect of LSI: Synonyms contribute strongly to document similarity.

SVD and Eigen Decomposition

The singular value decomposition and the eigen decomposition are closely related.

- The left-singular vectors of M are eigenvectors of MM^T.
- The right-singular vectors of M are eigenvectors of $M^T M$.
- The non-zero singular values of M (found on the diagonal entries of Σ) are the square roots of the non-zero eigenvalues of both $M^T M$ and MM^T.