Linear Algebra Methods for Data Mining

Saara Hyvönen, Saara.Hyvonen@cs.helsinki.fi

Spring 2007

Text mining & Information Retrieval

What is text mining?

- Methods for extracting useful information from large (and often unstructured) collections of texts.
- Information retrieval.
- E.g. search in data base of abstracts of scientific papers:
  Given a query with a set of keywords, find all documents that are relevant.
- Lots of stuff available (SAS Text Miner, Statistics Text-mining toolbox...)

Collections of texts

- Several test collections, e.g. Medline (1033 documents)

  `http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/`

- Somewhat larger text collections, e.g. Reuters (21578 documents)

  `http://kdd.ics.uci.edu/`

- Real data sets might be much larger: Number of terms: 10^4, number of documents: 10^6.

Example

Search in helka.linneanet.fi:

Search phrase	Result
computer science engineering	Nothing found
computing science engineering	Nothing found
computing in science	Computing in Science and Engineering

Straightforward word matching is not good enough!

Latent Semantic Indexing (LSI): More than literal matching - conceptual-based modelling.

Vector space IR model

Term-document matrix:

	Doc1	Doc2	Doc3	Doc4	Query
Term1	1	0	1	0	1
Term2	0	0	1	1	1
Term3	0	1	1	0	0

- The documents and the query are represented by a vector in $\mathbb{R}^n$ (here $n = 3$).

- In applications n is large!!!

From term-document matrix,

- Find document vector(s) close to query.
  What is "close"? Use some distance measure in $\mathbb{R}^n$!

- Use SVD for data compression and retrieval enhancement.

- Find "topics" or "concepts".

Latent Semantic Indexing (LSI)

1. Document file preparation/ preprocessing:
   - Indexing: collecting terms
   - Use stop list: eliminate ”meaningless” words
   - Stemming
2. Construction term-by-document matrix, sparse matrix storage.
3. Query matching: distance measures.
4. Data compression by low rank approximation: SVD
5. Ranking and relevance feedback.

Stop List

Eliminate words that occur in "all documents":

We consider the **computation** of an **eigenvalue** and corresponding **eigenvector** of a **Hermitian positive definite matrix** assuming that good **approximations** of the wanted **eigenpair** are already available, as may be the case in **applications** such as **structural mechanics**. We analyze **efficient implementations** of **inexact Rayleigh quotient-type methods,** which involve the **approximate solution** of a **linear system** at each **iteration** by means of the **Conjugate Residuals method.**

ftp://ftp.cs.cornell.edu/pub/smart/english.stop

Beginning of the cornell list:

a a's able about above according accordingly across actually after afterwards again against ain't all allow allows almost alone along already also although always am among amongst an and another any anybody anyhow anyone anything anyway anyways anywhere apart appear appreciate appropriate are aren't around as aside ask asking associated at available away awfully b be became because become becomes becoming been before beforehand behind being believe below beside besides best better between beyond both brief but by c c'mon c's came can...

Stemming

$$\left.\begin{array}{l}\text{computable}\\ \text{computation}\\ \text{computing}\\ \text{computational}\end{array}\right\} \Rightarrow \quad \text{comput}$$

`http://gatekeeper.dec.com/pub/net/infosys/wais/ir-book-resources/stemmer`

`http://www.tartarus.org/~martin/PorterStemmer/`

Example

changes of the nucleid acid and phospholipid levels of the livers in the course of fetal and postnatal developement. we have followed the evolution of dna, rna and pl in the livers of rat foeti removed between the fifteenth and the twenty-first day of gestation and of rats newly-born or at weaning. we can observe the following facts.

Stemmed (using Porter stemmer):

chang of the nucleid acid and phospholipid **level** of the **liver** in the **cours** of fetal and **postnat develop**. we have **follow** the **evolut** of dna, rna and pl in the **liver** of rat foeti **remov** between the fifteenth and the **twenti**-first **dai** of **gestat** and of rats **newli**-born or at **wean**. we can **observ** the **follow fact**.

Preprocessed text (stemmed and with stop words removed):

nucleid acid phospholipid **level liver** fetal **postnat develop evolut** dna rna pl **liver** rat foeti **remov** fifteenth **twenti dai gestat rat newli** born **wean observ fact**

Note: Sometimes also words that appear only once in the whole data set are removed.

Queries

Also queries are preprocessed $\Rightarrow$ natural language queries:

I want to know how to compute singular values of data matrices, especially such that are large and sparse

becomes

comput singular value data matri large sparse

Inverted file structures

- Document file: Each document has a number and all terms are identified
- Dictionary: Sorted list of all unique terms
- Inversion List: Pointers from a term to the documents that contain that term (column index for non-zeros in a row of the matrix).

Example

Terms	Documents:
T1 Baby	D1: **Infant** & **Toddler** First Aid
T2 Child	D2: **Babies** and **Children's** Room (for your **Home**)
T3 Guide	D3: **Child Safety** at **Home**
T4 Health	D4: Your **Baby's Health** and **Safety**:
T5 Home	From **Infant** to **Toddler**
T6 Infant	D5: **Baby Proofing** Basics
T7 Proofing	D6: Your **Guide** to Easy Rust **Proofing**
T8 Safety	D7: Beanie **Babies** Collector's **Guide**
T9 Toddler	

Term-by-Document Matrix

$$\hat{\mathbf{A}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The element $\hat{\mathbf{A}}_{ij}$ is the weighted frequency of term i in document j.

Normalization

$$\mathbf{A} = \begin{pmatrix} 0 & 0.577 & 0 & 0.447 & 0.707 & 0 & 0.707 \\ 0 & 0.577 & 0.577 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.707 & 0.707 \\ 0 & 0 & 0 & 0.447 & 0 & 0 & 0 \\ 0 & 0.577 & 0.577 & 0 & 0 & 0 & 0 \\ 0.707 & 0 & 0 & 0.447 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.707 & 0.707 & 0 \\ 0 & 0 & 0.577 & 0.447 & 0 & 0 & 0 \\ 0.707 & 0 & 0 & 0.447 & 0 & 0 & 0 \end{pmatrix}$$

Each column has Euclidean length 1.

Simple Query Matching

Given a query vector $\mathbf{q}$, find the columns $\mathbf{a}_j$ of $\mathbf{A}$ which have

$$\text{dist}(\mathbf{q}, \mathbf{a}_j) \leq tol.$$

Common distance definitions:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \begin{cases} \|\mathbf{x} - \mathbf{y}\|_2, \quad \text{(Euclidean distance)} \\ 1 - \cos(\theta(\mathbf{x}, \mathbf{y})) = 1 - \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} \\ \arccos(\theta(\mathbf{x}, \mathbf{y})) \end{cases}$$

Query: child home safety

$$\hat{\mathbf{q}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{q} = \begin{pmatrix} 0 \\ 0.577 \\ 0 \\ 0 \\ 0.577 \\ 0 \\ 0 \\ 0.577 \\ 0 \end{pmatrix}.$$

Cosine similarity:

$$\cos(\theta(\mathbf{x},\mathbf{y})) = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} \geq tol.$$

(Note: $\|\mathbf{x}\| = \|\mathbf{y}\| = 1.$)

In the example, cosine similarity with each column of $\mathbf{A}$:

$$\mathbf{q}^T\mathbf{A} = (0 \quad 0.667 \quad 1.000 \quad 0.258 \quad 0 \quad 0 \quad 0)$$

With a cosine threshold of $tol = 0.5$ documents 2 and 3 would be returned:

Terms	Documents:
T1 Baby	D1: **Infant** & **Toddler** First Aid
T2 Child	D2: **Babies** and **Children's** Room (for your **Home**)
T3 Guide	D3: **Child Safety** at **Home**
T4 Health	D4: Your **Baby's Health** and **Safety**:
T5 Home	From **Infant** to **Toddler**
T6 Infant	D5: **Baby Proofing** Basics
T7 Proofing	D6: Your **Guide** to Easy Rust **Proofing**
T8 Safety	D7: Beanie **Babies** Collector's **Guide**
T9 Toddler	

Query: child home safety.

Weighting schemes

- Don't just count occurences of terms in documents but apply a *term weighting scheme.*

- elements of term-by-document matrix $\mathbf{A}$ weighted depending on characteristics of document collection.

- A common such scheme: TFIDF: TF $=$ Term Frequency, IDF $=$ Inverse Document Frequency

TFIDF

- TF = Term Frequency, IDF = Inverse Document Frequency
- Define

$$\mathbf{A}_{ij} = f_{ij} \log(n/n_i),$$

  where f_{ij} is frequency of term i in document j, n is the number of documents in the collection, and n_i is the number of documents in which the term i appears.

- here TF=f_{ij}, IDF=$\log(n/n_i)$. Other choices possible.

Sparse matrix storage

$$\mathbf{A} = \begin{pmatrix} 0.67 & 0 & 0 & 0.29 \\ 0 & 0.71 & 0.41 & 0.29 \\ 0.33 & 0 & 0.41 & 0.29 \\ 0.67 & 0 & 0 & 0 \end{pmatrix}$$

Compressed row storage:

value	0.67	0.29	0.71	0.41	0.29	0.33	0.41	0.29	0.67
col-ind	1	4	2	3	4	1	3	4	1
row-ptr	1	3	6	9					

Compressed column storage: analogous.

Performance evaluation

Return all documents for which the cosine measure

$$\frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} \geq tol.$$

Low tolerance $\Rightarrow$ lots of documents returned. More relevant documents, but also more irrelevant ones.

High tolerance $\Rightarrow$ less irrelevant documents, but relevant ones may be missed.

Example

Back to our example case: Recall that cosine similarity with each column of $\mathbf{A}$ was

$$\mathbf{q}^T\mathbf{A} = (0 \quad 0.667 \quad 1.000 \quad 0.258 \quad 0 \quad 0 \quad 0).$$

tol$=$ 0.8: one document.

tol$=$0.5: two documents.

tol$=$0.2: three documents.

Example: MEDLINE

Homework data set...

1033 documents, around 5000 terms, 30 querys.

Query 21: language developement in infancy and pre-school age.

tol=0.4: two documents.

tol=0.3: 6 documents.

Precision and Recall

- Precision:

$$P = \frac{D_r}{D_t}$$

- Recall:

$$R = \frac{D_r}{N_r}$$

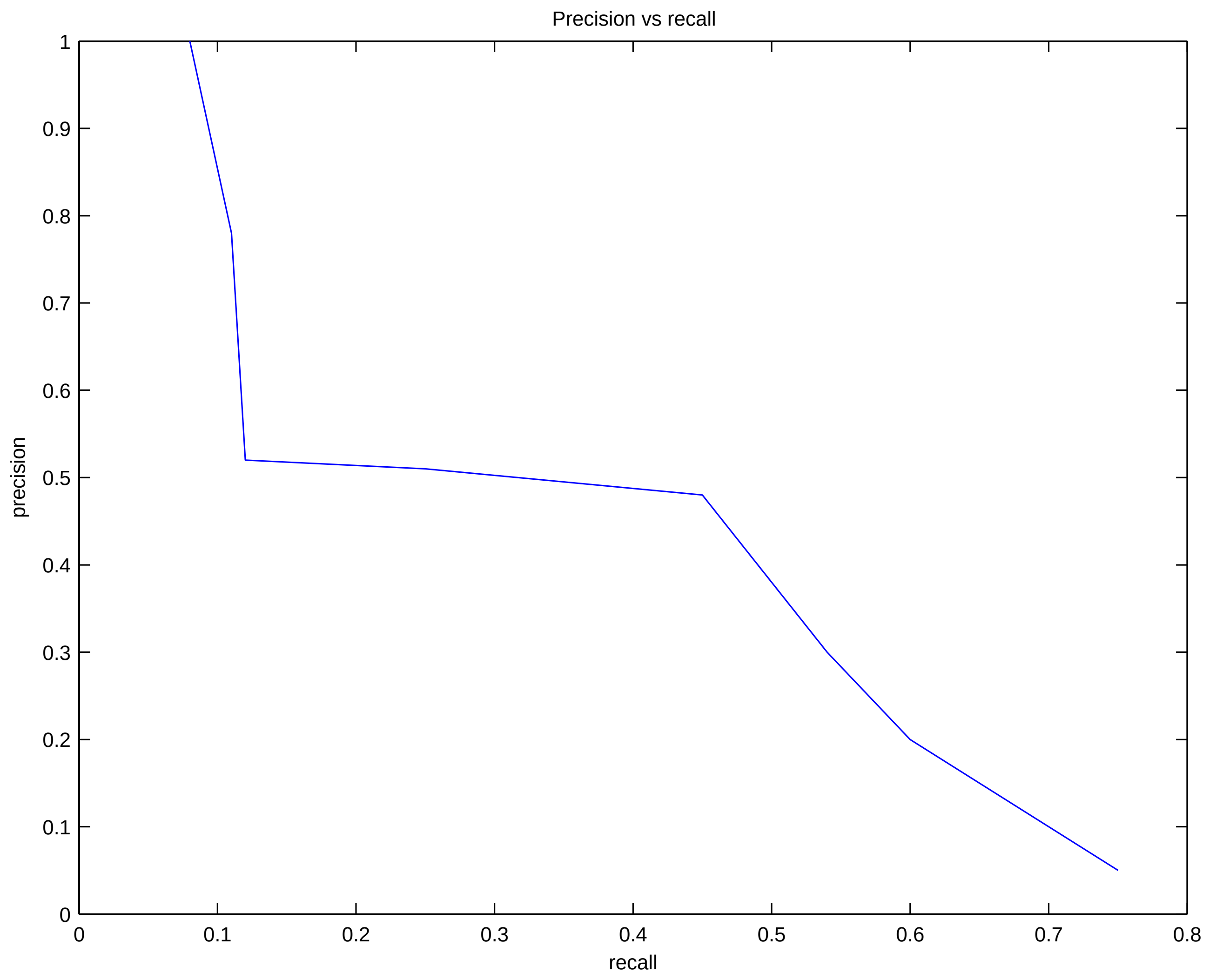
- D_r is the number of relevant documents retrieved,
  D_t is the total number of documents retrieved,
  N_r is the total number of relevant documents in the database.

- What we want: High precision, high recall.

- Return all documents for which the cosine measure

$$\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} \geq tol.$$

- High tolerance: high precision, low recall.
- low tolerance: low precision, high recall.


Precision vs recall
precision
recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

The Singular Value Decomposition

$\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$.

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix} \mathbf{V}^T, \quad \mathbf{U} \in \mathbb{R}^{m \times m}, \quad \mathbf{\Sigma} \in \mathbb{R}^{n \times n}, \quad \mathbf{V} \in \mathbb{R}^{n \times n}.$$

$\mathbf{U}$ and $\mathbf{V}$ are orthogonal, and $\mathbf{\Sigma}$ is diagonal:

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq \sigma_{r+1} = ... = \sigma_n = 0.$$

Rank($\mathbf{A}$)$= r$.

(Analogous for $m < n$. We assume $m \geq n$ for simplicity.)

The approximation problem

$$\min_{\text{rank}(\mathbf{B}) \le k} \|\mathbf{A} - \mathbf{B}\|_2$$

has the solution

$$\mathbf{B} = \mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T,$$

where $\mathbf{U}_k$, $\mathbf{V}_k$ and $\mathbf{\Sigma}_k$ are the matrices with the k first singular vectors and values of $\mathbf{A}$.

Latent Semantic Indexing

- Assumption: there is some underlying latent semantic structure in the data.
- E.g. car and automobile occur in similar documents, as do cows and sheep.
- This structure can be enhanced by projecting the data (the term-by-document-matrix and the queries) onto a lower dimensional space using SVD.

- Normalize columns to have length $=1$ (otherwise long documents dominate first singular values and vectors)

- Compute the SVD of the term-by-document matrix: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and approximate $\mathbf{A}$ by

$$\mathbf{A} \approx \mathbf{U}_k(\mathbf{\Sigma}_k\mathbf{V}_k^T) = \mathbf{U}_k\mathbf{D}_k.$$

- $\mathbf{U}_k$: orthogonal basis, that we use to approximate all the documents

- $\mathbf{D}_k$: Column j hold the coordinates of document j in the new basis.

- $\mathbf{D}_k$ is the projection of $\mathbf{A}$ onto the subspace spanned by $\mathbf{U}_k$.

Data compression

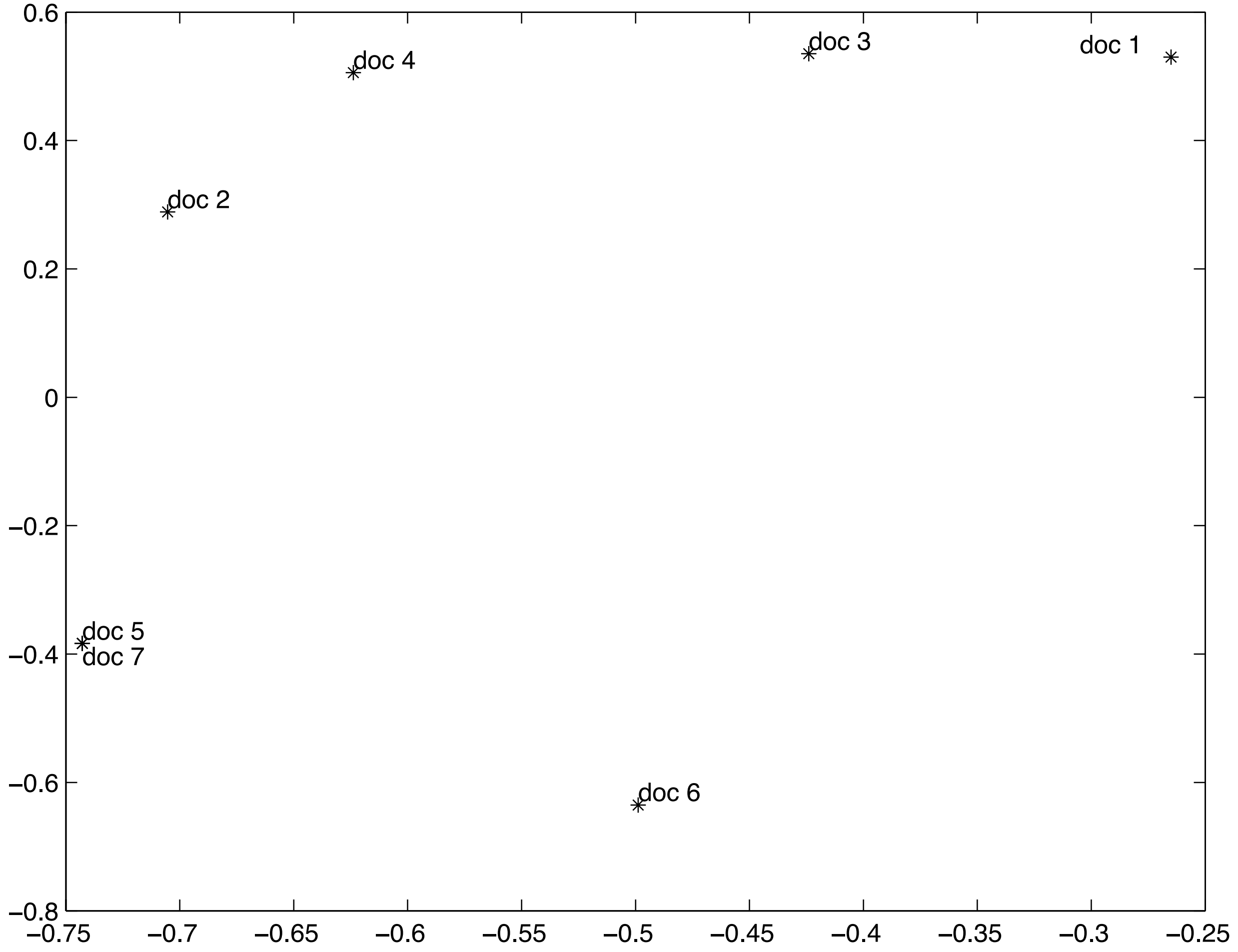
$$\mathbf{A} \approx \mathbf{A}_k \mathbf{U}_k (\mathbf{\Sigma}_k \mathbf{V}_k^T) = \mathbf{U}_k \mathbf{D}_k.$$

$\mathbf{D}_k$ holds the coordinates of all documents in terms of the k first (left) singular vectors $\mathbf{u}_1, ..., \mathbf{u}_k$.

Take a look at our "child home safety" -example:

$$\mathbf{D}_2 = \begin{pmatrix} -0.27 & -0.71 & -0.42 & -0.62 & -0.74 & -0.50 & -0.74 \\ 0.53 & 0.29 & 0.54 & 0.51 & -0.38 & -0.64 & -0.38 \end{pmatrix}$$

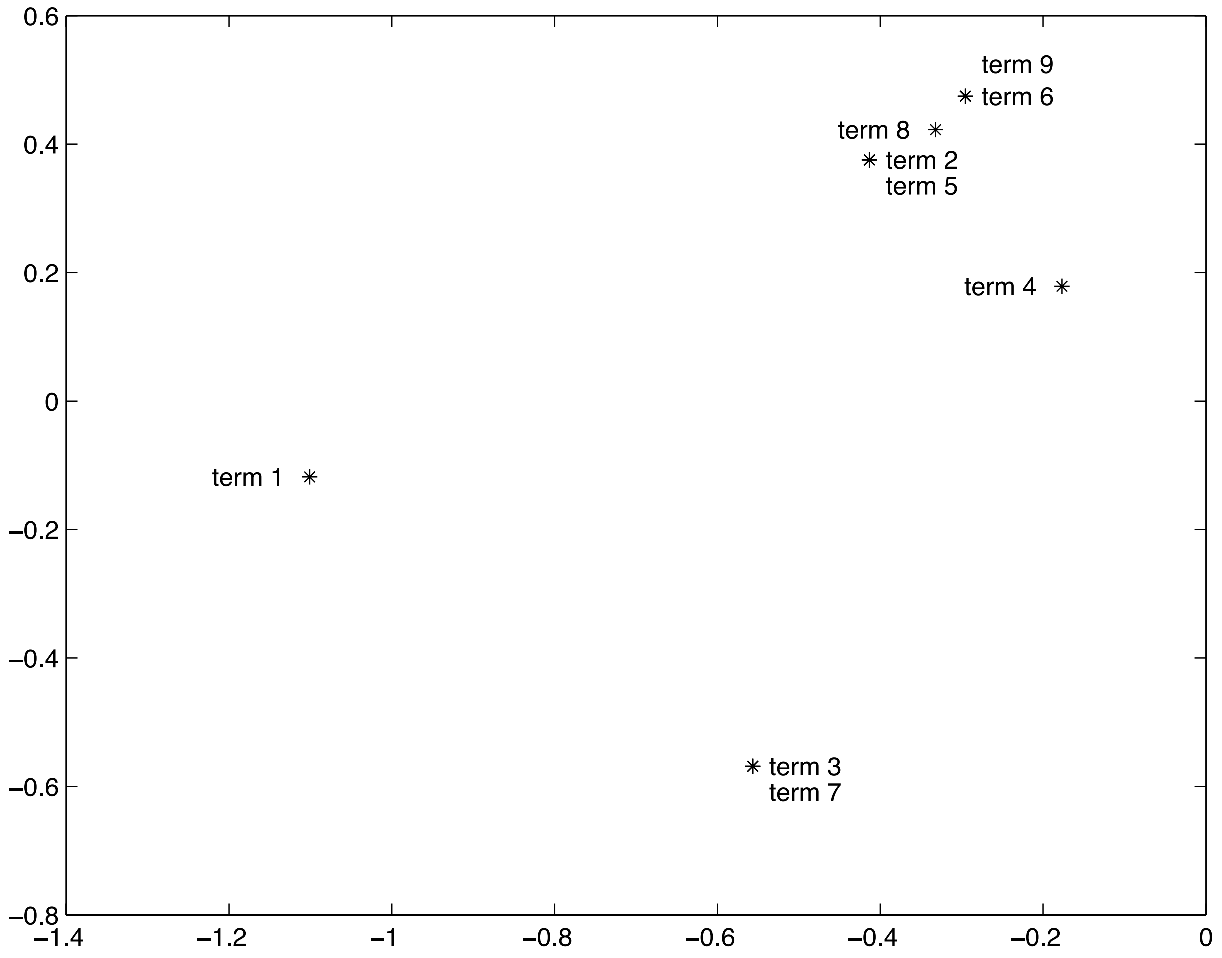
$\mathbf{D}_k$ holds the coordinates of all documents in terms of the k first singular vectors, $k = 2$. Documents plotted in this basis:



$$\mathbf{A} \approx \mathbf{A}_k = (\mathbf{U}_k \mathbf{\Sigma}_k)\mathbf{V}_k^T = \mathbf{T}_k \mathbf{V}_k^T$$

$\mathbf{T}_k$ holds the coordinates of all terms in terms of the k first (right) singular vectors $\mathbf{v}_1, ..., \mathbf{v}_k$:

$$\mathbf{T}_2 = \begin{pmatrix} -1.10 & -0.41 & -0.56 & -0.18 & -0.41 & -0.30 & -0.56 & -0.33 & -0.30 \\ -0.12 & 0.38 & -0.57 & 0.18 & 0.38 & 0.47 & -0.57 & 0.42 & 0.47 \end{pmatrix}$$


term 9
term 6
term 8
term 2
term 5
term 4
term 1
term 3
term 7
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1.4
−1.2
−1
−0.8
−0.6
−0.4
−0.2
0

Error in matrix approximation

Relative error:

$$\mathbf{E} = \frac{\|\mathbf{A} - \mathbf{A}_k\|_2}{\|\mathbf{A}\|_2}$$

Example:

k	1	2	3	4	5	6	7
error	0.80	0.75	0.50	0.45	0.36	0.12	0

(Note: often the Frobenius norm is used instead of the 2-norm!)

Query matching after data compression

Represent term-document matrix by $\mathbf{A}_k = \mathbf{U}_k \mathbf{D}_k$.

Compute

$$\mathbf{q}^T \mathbf{A}_k = \mathbf{q}^T \mathbf{U}_k \mathbf{D}_k = (\mathbf{U}_k^T \mathbf{q})^T \mathbf{D}_k.$$

Project the query:

$$\mathbf{q}_k = \mathbf{U}_k^T \mathbf{q}.$$

Cosines:

$$\cos\theta_j = \frac{\mathbf{q}_k^T (\mathbf{D}_k \mathbf{e}_j)}{\|\mathbf{q}_k\|_2 \|\mathbf{D}_k \mathbf{e}_j\|_2}$$

Query matching performed in k-dimensional space!

Example, continued

Cosines for query and original data:

$$\mathbf{q}^T\mathbf{A} = (0 \quad 0.667 \quad 1.000 \quad 0.258 \quad 0 \quad 0 \quad 0)$$

With a cosine threshold of $tol = 0.5$ documents 2 and 3 returned.

Cosines after projection to two-dimensional space:

$$(0.979 \quad 0.872 \quad 1.000 \quad 0.976 \quad 0.192 \quad -0.233 \quad 0.192)$$

Now documents 1, 2, 3 and 4 are considered relevant!

Terms	Documents:
T1 Baby	D1: **Infant** & **Toddler** First Aid
T2 Child	D2: **Babies** and **Children's** Room (for your **Home**)
T3 Guide	D3: **Child Safety** at **Home**
T4 Health	D4: Your **Baby's Health** and **Safety**:
T5 Home	From **Infant** to **Toddler**
T6 Infant	D5: **Baby Proofing** Basics
T7 Proofing	D6: Your **Guide** to Easy Rust **Proofing**
T8 Safety	D7: Beanie **Babies** Collector's **Guide**
T9 Toddler	

Query: child home safety.

LSI

- LSI (SVD compression) enhances retrieval quality!
- Even if the approximation error is still quite high!
- LSI is able to deal with synonyms (toddler, child, infant)...
- ...and with polysemy (same word has different meanings).
- True also for larger document collections...
- ...and for surprisingly low rank and high matrix approximation error.
- But: not for all queries - for most, and average performance counts.

How to choose k?

Typical term-document matrix has no clear gaps n singular values.

Rank must be chosen by retrieval experiments.

Improved retrieval event when matrix approximation error is high $\Rightarrow$
2-norm may not be the right way to measure information content in the term-document matrix.

Updating

A new document $\mathbf{d}$ arrives: compute its coordinates in terms of $\mathbf{U}_k$:

$$\mathbf{d}_k = \mathbf{U}_k^T \mathbf{d}.$$

Note: we no longer have a SVD of $(\mathbf{A}\ \mathbf{d})$. Recomputing the SVD is expensive!

References

[1] Lars Eldén: Numerical Linear Algebra and Data Mining Applications, Draft.

[2] D. Hand, H. Mannila, P. Smyth, Principles of Data Mining, The MIT Press, 2001.

[3] Soumen Chakrabarti: Mining the Web, Morgan Kaufmann Publishers, 2003.

[4] `http://www.cs.utk.edu/~lsi/`

[5] M. W. Berry and M. Browne, Understanding Search Engines, Mathematical Modeling and Text Retrieval, SIAM, Philadelphia, 1999.