Search Engines

WS 2009 / 2010

Lecture 10, Thursday January 14th, 2010

(Latent Semantic Indexing)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

Overview of Today's Lecture

UNI FREIBURG

- Learn about Latent Semantic Indexing (LSI)
  - a method that addresses the synonymy problem
  - fully automatic, does not require any understanding of the words
  - uses method from linear algebra, which you learn on the way
    - Eigenvector (Schur) decomposition
    - Singular Value Decomposition (SVD)

The Synonymy Problem

- Here is a toy term-document matrix
  - it's the one from Lecture 3, remember?

	Doc1	Doc2	Doc3	Doc4	Doc5
internet	0.9	0	0	0.6	0
web	0.3	0.9	0	0.4	0
surfing	0.7	0.6	0	0.8	0.6
beach	0	0	1.0	0.3	0.7

Qry
1.0
0
0
0

- Problem
  - Doc2 not retrieved although the words internet and web are synonymous here and so Doc2 is just as relevant as Doc1

Syonym Dictionary

- One solution would be a synonym dictionary
  - often makes sense, but hard to maintain and keep up-to-date
  - in this lecture, we will look at a fully automatic method
  - but how can that work?

The Rank of a Matrix

- Assume our matrix is the product of these two

4×2

1	0
1	0
1	1
0	1

2×5

1	1	0	0.5	0
0	0	1	0.5	1

= [compute product]

4×5

1 1 0 0.5 0
1 1 0 0.5 0
1 1 1 1 1
0 0 1 0.5 1

- This is a matrix with column rank 2
  - column rank k = all columns can be written as a linear combination of k common "base" columns, but not less
  - the row rank is defined analogously
  - Theorem: column rank = row rank

Perturbing a Low-Rank Matrix

UNI FREIBURG

- Assume we change just two of the entries

1	0	0	0.5	0
1	1	0	0.5	0
1	1	1	1	0
0	0	1	0.5	1

- Now the matrix has full rank (4) again
  - but assuming that it came from a rank-2 matrix with just two entries changed
  - it's not hard to guess what the original rank-2 matrix was
  - LSI does this recovering automatically

Latent Semantic Indexing (LSI)

- For a given $m \times n$ term-document matrix A
  - and for a given rank k, typically $<< \min(m, n)$

    (note that the maximal rank is $\min(m, n)$, why?)
  - LSI computes that rank-k matrix A_k with minimal distance to A
  - formally: $\text{argmin}_{A_k,\ \text{rank}(A_k) = k} \ \| A - A_k \|$
  - where $\| . \|$ is the Frobenius norm
    - that is, for a matrix $A = [a_{ij}]$
    - $\| A \| := \text{sqrt}(\Sigma a_{ij}^2)$

How to compute such a low-rank approximation?

Eigenvector (Schur) Decomposition

UNI FREIBURG

- Theorem:
  - let A be a symmetric $m \times m$ matrix
  - then A can be written as $U \cdot D \cdot U^T$
  - where U is unitarian, that is, $U \cdot U^T = U^T \cdot U = I$
  - and D is a diagonal matrix
    - with the eigenvalues on its diagonal
- Recall
  - when $A \cdot x = \lambda \cdot x$
  - then x is called an eigenvector of A with eigenvalue λ
  - if x is an eigenvector then so are all multiples of x
  - A has m linear independent eigenvectors

    which hence form a basis of the R^m

Eigenvector Decomposition — Example

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$$A \cdot x = \lambda \cdot x$$
$$A \cdot x - \lambda \cdot x = 0$$
$$(A - \lambda I) \cdot x = 0 \qquad x \neq 0$$

First, we need the eigenvalues

$$|A - \lambda I| = 0 \qquad \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = (2-\lambda)^2 - 1$$

[do example]

$$= \lambda^2 - 4\lambda + 3$$
$$= (\lambda - 3)(\lambda - 1)$$

Eigenvalue (EV) 3:

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \Rightarrow x_1 = x_2 \Rightarrow \left\langle \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\rangle$$

1st Eigvec.

Eigenvalue 1:

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \Rightarrow x_1 = -x_2 \Rightarrow \left\langle \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\rangle$$

2nd Eigvec

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

EV1: $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ mit val 3

EV2: $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ mit val 1

normalize in L_2-norm:

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \Rightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad U^T = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Svd decomposition:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}}$$

$U \quad D \quad U^T$

Singular Value Decomposition (SVD)

UNI FREIBURG

- Theorem
  - Let A be an arbitrary rectangular m x n matrix A
  - then A can be written as $U \cdot \Sigma \cdot V^T$
  - where U is m x k, Σ is k x k, and V is n x k k = rank(A)
  - and $U^T \cdot U = I$ and $V^T \cdot V = I$ (but not vice versa)
  - and Σ is a diagonal matrix
    - with the so-called singular values on its diagonal
- Example
  - one of the exercises
  - do it by hand please (it is doable by hand)

SVD Example

- Let's take our slightly perturbed rank-2 matrix

1	0	0	0.5	0
1	1	0	0.5	0
1	1	1	1	0
0	0	1	0.5	1

=

-0.32	-0.24	-0.90	-0.17
-0.50	-0.42	0.15	0.74
-0.75	0.05	0.36	-0.55
-0.29	0.87	-0.20	0.34

U

·

2.62	0	0	0
0	1.47	0	0
0	0	0.70	0
0	0	0	0.45

Σ

·

-0.60	-0.42	-0.55	0.03	-0.41
-0.48	-0.25	0.73	0.42	0
-0.39	0.63	0.22	-0.48	-0.40
-0.50	0.11	-0.16	-0.22	0.82

V^T

How to Compute the SVD

- Easy via the Eigenvector (Schur) decomposition

$A_{m \times m} \quad A = U \cdot \Sigma \cdot V^T \quad , \quad U^T U = I \quad V^T V = I$

$A^T_{m \times m}$

$$\underset{m \times m}{A A^T} = U \cdot \Sigma \cdot V^T \cdot V \cdot \Sigma \cdot U^T = U \cdot \Sigma^2 \cdot U^T$$

[do the calculations]

$$\underset{n \times n}{A^T A} = V \cdot \Sigma \cdot \underbrace{U^T \cdot U}_{=I} \cdot \Sigma \cdot V^T = V \cdot \Sigma^2 \cdot V^T$$

- This is not the most efficient way however
  - in pratice, use numerical methods
  - one of the most efficient ones is called the Lanczos method
  - which has complexity $O(k \cdot nz)$, where k is the rank and nz is the number of non-zero values in the matrix
  - note that term-document matrices are sparse: $nz << n \cdot m$

Best Rank-k Approximation via SVD

- Take the SVD $U \cdot \Sigma \cdot V^T$ of the given matrix A
  - and keep only the first k columns of U
  - the upper $k \times k$ part of Σ
  - and the first k rows of V^T
  - here is an example for the SVD from two slides ago and $k = 2$

-0.32	-0.24	-0.90	-0.17
-0.50	-0.42	0.15	0.74
-0.75	0.05	0.36	-0.55
-0.29	0.87	-0.20	0.34

·

2.62	0	0	0
0	1.47	0	0
0	0	0.70	0
0	0	0	0.45

·

-0.60	-0.42	-0.55	0.03	-0.41
-0.48	-0.25	0.73	0.42	0.00
-0.39	0.63	0.22	-0.48	-0.40
-0.50	0.11	-0.16	-0.22	0.82

Best Rank-k Approximation via SVD

- Take the SVD $U \cdot \Sigma \cdot V^T$ of the given A
  - and keep only the first k columns of U
  - the upper $k \times k$ part of Σ
  - and the first k rows of V^T
  - here is an example for the SVD from two slides ago and $k = 2$

-0.32	-0.24	0	0
-0.50	-0.42	0	0
-0.75	0.05	0	0
-0.29	0.87	0	0

·

2.62	0	0	0
0	1.47	0	0
0	0	0	0
0	0	0	0

·

-0.60	-0.42	-0.55	0.03	-0.41
-0.48	-0.25	0.73	0.42	0.00
0	0	0	0	0
0	0	0	0	0

Best Rank-k Approximation via SVD

- Take the SVD $U \cdot \Sigma \cdot V^T$ of the given A
  - and keep only the first k columns of U
  - the upper $k \times k$ part of Σ
  - and the first k rows of V^T
  - here is an example for the SVD from two slides ago and $k = 2$

-0.32	-0.24
-0.50	-0.42
-0.75	0.05
-0.29	0.87

·

2.62	0
0	1.47

·

-0.60	-0.42	-0.55	0.03	-0.41
-0.48	-0.25	0.73	0.42	0.00

Best Rank-k Approximation via SVD

- Take the SVD $U \cdot \Sigma \cdot V^T$ of the given A
  - and keep only the first k columns of U
  - the upper k x k part of Σ
  - and the first k rows of V^T
  - here is an example for the SVD from two slides ago and k = 2

1	0	0	0.5	0
1	1	0	0.5	0
1	1	1	1	0
0	0	1	0.5	1

our original A

0.7	0.4	0.2	0.4	0.2
1.0	0.7	0.3	0.4	0.2
1.1	0.8	1.1	0.8	0.7
-0.1	0.0	1.3	0.5	0.8

rank-2 approximation

Problems with LSI

- The approximation is good ...
  - ... but the vectors of the decomposition are not intuitive
  - explain by example on previous slides
- Alternatives
  - PLSI = probabilistic LSI
    - find column-stochastic matrices (entries non-negative, column sum = 1) U and V such that $A = U \cdot \Sigma \cdot V^T$
  - NMF = non-negative matrix factorization
    - find any non-negative matrices U and V such that $A = U \cdot V$
  - Quality of LSI, PLSI, NMF is about the same, but the matrices U and V have a more natural interpretation for PLSI and NMF

Practical Issues

- In practice
  - m (#terms) and n (#documents) are very large
  - decomposition on such large matrices is very expensive
  - also, the concepts found are based on mere co-occurrence
    - correlations found are not always what one would expect
    - many correlations are not found because there is no strong signal in the data
    - here is a demo

Literature

- Latent Semantic Indexing (LSI)
  - Deerwester, Dumais, Landauer, Furnas, Harshman

    Indexing by Latent Semantic Analysis, JASIS 41(6), 1990
  - Bast, Majumdar

    Why Spectral Retrieval Works, SIGIR 2005
- Alternative methods
  - Thomas Hofmann

    Probabilistic Latent Semantic Indexing, SIGIR 1999
  - Daniel Lee, Sebastian Seung

    Algorithms for Non-negative Matrix Factorization, NIPS 2000

Literature 2

- Eigenvalue decomposition, SVD
  - http://en.wikipedia.org/wiki/Schur_decomposition
  - http://en.wikipedia.org/wiki/Singular_value_decomposition