

Advanced IR Models

Sergej Sizov

Information Retrieval

Summer term 2008

Key Idea of Latent Concept Models

Objective:

Transformation of document vectors from high-dimensional term vector space into lower-dimensional **topic vector space** with

- exploitation of term correlations
(e.g. „Web“ and „Internet“ frequently occur in together)
- implicit differentiation of polysems that exhibit different term correlations for different meanings
(e.g. „Java“ with „Library“ vs. „Java“ with „Kona Blend“ vs. „Java“ with „Borneo“)

mathematically:

given: m terms, n docs (usually $n > m$) and a

$m \times n$ term-document similarity matrix A ,

needed: largely similarity-preserving mapping of column vectors of A

into k -dimensional vector space ($k \ll m$) for given k

Foundations from Linear Algebra

A set S of vectors is called **linearly independent** if no $x \in S$ can be written as a linear combination of other vectors in S .

The **rank** of matrix A is the maximal number of linearly independent row or column vectors.

A **basis** of an $n \times n$ matrix A is a set S of linearly independent row or column vectors such that all rows or columns are linear combinations of vectors from S .

A set S of $n \times 1$ vectors is an **orthonormal basis** if for all $x, y \in S$:

$$\|x\|_2 := \sqrt{\sum_{i=1}^n X_i^2} = 1 = \|y\|_2 \quad \text{and} \quad x \cdot y = 0$$

Eigenvalues and Eigenvectors

Let A be a real-valued $n \times n$ matrix, x a real-valued $n \times 1$ vector, and λ a real-valued scalar. Solutions $x \neq 0$ and λ of the equation $Ax = \lambda x$ are called an **Eigenvector** and **Eigenvalue** of A .

Eigenvectors of A are vectors whose direction is preserved by the linear transformation described by A .

The Eigenvalues of A are the roots (Nullstellen) of the characteristic polynomial $f(\lambda)$ of A : $f(\lambda) = |A - \lambda I| = 0$

with the determinant (developing the i -th row):

$$|A| = \sum_{j=1}^n (-1)^{i+j} a_{ij} |A^{(ij)}| \quad \text{where matrix } A^{(ij)} \text{ is derived from } A \text{ by removing the } i\text{-th row and the } j\text{-th column}$$

The real-valued $n \times n$ matrix A is **symmetric** if $a_{ij} = a_{ji}$ for all i, j .

A is **positive definite** if for all $n \times 1$ vectors $x \neq 0$: $x^T \times A \times x > 0$.

If A is symmetric then all Eigenvalues of A are real.

If A is symmetric and positive definite then all Eigenvalues are positive.

Singular Value Decomposition (SVD)

Theorem:

Each real-valued $m \times n$ matrix A with rank r can be decomposed into the form $\mathbf{A} = \mathbf{U} \times \mathbf{\Delta} \times \mathbf{V}^T$

with an $m \times r$ matrix \mathbf{U} with orthonormal column vectors, an $r \times r$ diagonal matrix $\mathbf{\Delta}$, and

an $n \times r$ matrix \mathbf{V} with orthonormal column vectors.

This decomposition is called *singular value decomposition* and is unique when the elements of $\mathbf{\Delta}$ are sorted.

Theorem:

In the singular value decomposition $\mathbf{A} = \mathbf{U} \times \mathbf{\Delta} \times \mathbf{V}^T$ of matrix A the matrices \mathbf{U} , $\mathbf{\Delta}$, and \mathbf{V} can be derived as follows:

- $\mathbf{\Delta}$ consists of the singular values of A ,
i.e. the positive roots of the Eigenvalues of $\mathbf{A}^T \times \mathbf{A}$,
- the columns of \mathbf{U} are the Eigenvectors of $\mathbf{A} \times \mathbf{A}^T$,
- the columns of \mathbf{V} are the Eigenvectors of $\mathbf{A}^T \times \mathbf{A}$.

SVD for Regression

Theorem:

Let A be an $m \times n$ matrix with rank r , and let $\mathbf{A}_k = \mathbf{U}_k \times \Delta_k \times \mathbf{V}_k^T$, where the $k \times k$ diagonal matrix Δ_k contains the k largest singular values of A and the $m \times k$ matrix \mathbf{U}_k and the $n \times k$ matrix \mathbf{V}_k contain the corresponding Eigenvectors from the SVD of A .

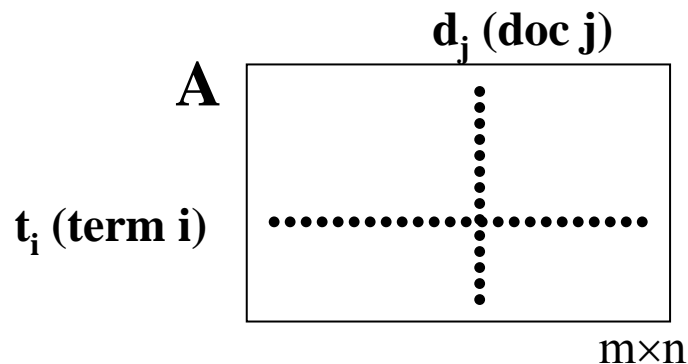
Among all $m \times n$ matrices C with rank at most k

\mathbf{A}_k is the matrix that minimizes the Frobenius norm

$$\|A - C\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n (A_{ij} - C_{ij})^2$$

SVD and Vector Space Model

Term-document matrix:



Observation: dot product $t_i^T \cdot t_j$ gives the „similarity“ between t_i, t_j over the documents. The matrix product $A \times A^T$ contains all these dot products. Element (x,y) (= element (y,x)) contains the dot product $t_x^T \cdot t_y$.

The matrix $A^T \times A$ contains the dot products between document vectors, and gives their „similarity“ over terms: element (m,n) (=element (n,m)) = $d_m^T \cdot d_n$

SVD Theorem: In the singular value decomposition $A = U \times \Delta \times V^T$ of matrix A the matrices U, Δ , and V can be derived as follows:

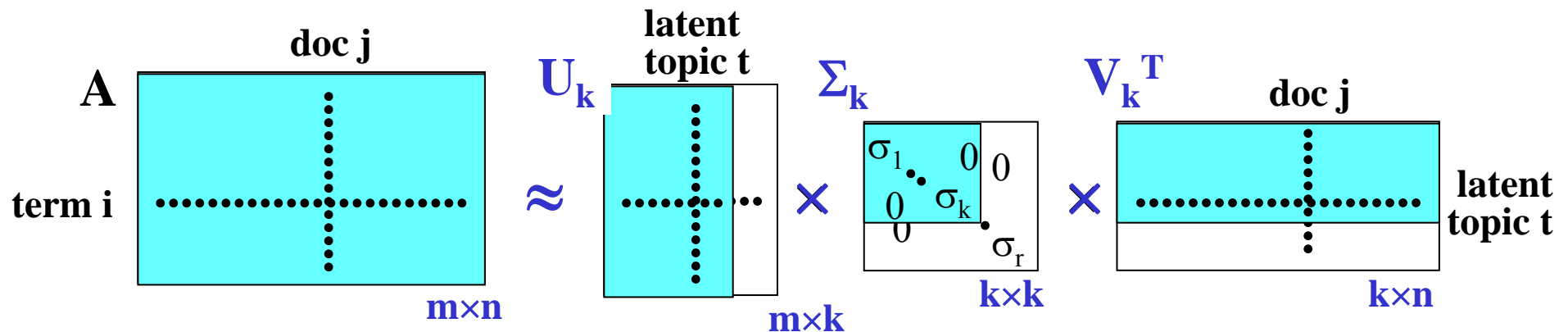
- Δ consists of the singular values of A ,
i.e. the positive roots of the Eigenvalues of $A^T \times A$,
- the columns of U are the Eigenvectors of $A \times A^T$,
- the columns of V are the Eigenvectors of $A^T \times A$.

Latent Semantic Indexing (LSI)

[Deerwester et al. 1990]

A is the $m \times n$ term-document matrix. Then:

- U and U_k are the $m \times r$ and $m \times k$ term-topic similarity matrices,
- V and V_k are the $n \times r$ and $n \times k$ document-topic similarity matrices,
- $A \times A^T$ and $A_k \times A_k^T$ are the term-term similarity matrices,
- $A^T \times A$ and $A_k^T \times A_k$ are the document-document similarity matrices



mapping of $m \times 1$ vectors into latent-topic space: $d_j \mapsto U_k^T \times d_j =: d_j'$

$q \mapsto U_k^T \times q =: q'$

scalar-product similarity in latent-topic space: $d_j'^T \times q' = ((\Delta_k V_k^T)_{*j})^T \times q'$

LSI: Indexing and Query Processing

- The matrix $\Delta_k V_k^T$ corresponds to a „**topic index**“ and is stored in a suitable data structure.
Instead of $\Delta_k V_k^T$ the simpler **index** V_k^T could be used.
- Additionally the **term-topic mapping** U_k must be stored.
- A **query** q (an $m \times 1$ column vector) in the term vector space is transformed into query $q' = U_k^T \times q$ (a $k \times 1$ column vector) and evaluated in the topic vector space (i.e. V_k) (e.g. by scalar-product similarity $V_k^T \times q'$ or cosine similarity)
- A **new document** d (an $m \times 1$ column vector) is transformed into $d' = U_k^T \times d$ (a $k \times 1$ column vector) and appended to the „index“ V_k^T as an additional column (**„folding-in“**)

LSI: Example

m=6 terms

t1: bak(e,ing)

t2: recipe(s)

t3: bread

t4: cake

t5: pastr(y,ies)

t6: pie

n=5 documents

d1: How to bake bread without bread recipes

d2: The classic art of Viennese Pastry

d3: Numerical recipes: the art of
scientific computing

d4: Breads, pastries, pies and cakes:
quantity baking recipes

d5: Pastry: a book of best French recipes

$$A = \begin{pmatrix} 0.4446 & 0.0000 & 0.0000 & 0.3422 & 0.0000 \\ 0.1083 & 0.0000 & 1.0000 & 0.0833 & 0.4002 \\ 0.8892 & 0.0000 & 0.0000 & 0.3422 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.6010 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.1908 & 0.9164 \\ 0.0000 & 0.0000 & 0.0000 & 0.6010 & 0.0000 \end{pmatrix}$$

LSI: Example

$$A = \begin{pmatrix} -0.1337 & 0.4385 & -0.0916 & -0.0858 \\ -0.4039 & 0.0798 & 0.9089 & 0.06680 \\ -0.1909 & 0.7045 & -0.1092 & -0.5105 \\ -0.1336 & 0.3000 & -0.1298 & 0.5997 \\ -0.8642 & -0.3535 & -0.3464 & -0.0906 \\ -0.1336 & 0.3000 & -0.1298 & 0.5997 \end{pmatrix} \quad U$$

$$\times \begin{pmatrix} 1.4543 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1764 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.9980 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.7115 \end{pmatrix} \quad \Delta$$

$$\times \begin{pmatrix} -0.1878 & -0.5942 & -0.2777 & -0.3233 & -0.65571 \\ 0.7061 & -0.3005 & 0.0678 & 0.5873 & -0.2482 \\ -0.0396 & -0.3471 & 0.9107 & -0.2155 & 0.0464 \\ -0.6817 & -0.1274 & 0.0940 & 0.7100 & -0.0792 \end{pmatrix} \quad V^T$$

LSI: Example

$$A_2 = \begin{pmatrix} 0.4008 & -0.0395 & 0.0890 & 0.3659 & -0.0006 \\ 0.1766 & 0.3209 & 0.1695 & 0.2451 & 0.3619 \\ 0.6373 & -0.0841 & 0.1333 & 0.5766 & -0.0237 \\ 0.2857 & 0.0094 & 0.0779 & 0.2701 & 0.0398 \\ -0.0576 & 0.8718 & 0.3209 & 0.1621 & 0.9273 \\ 0.2857 & 0.0094 & 0.0779 & 0.2701 & 0.0398 \end{pmatrix} = U_2 \times \Delta_2 \times V_2^T$$

LSI: Example

query q : baking bread

$$q = (1 \ 0 \ 1 \ 0 \ 0 \ 0)^T$$

transformation into topic space with $k=2$

$$q' = U_k^T \times q = (-0.3246 \ 1.1430)^T$$

scalar product similarity in topic space with $k=2$:

$$\text{sim}(q, d1) = V_{k*_1}^T \times q' \approx 0.87 \quad \text{sim}(q, d2) = V_{k*_2}^T \times q' \approx -0.15$$

$$\text{sim}(q, d3) = V_{k*_3}^T \times q' \approx 0.17 \quad \text{etc.}$$

Folding-in of a new document $d6$:

algorithmic recipes for the computation of pie

$$d6 = (0 \ 0.7071 \ 0 \ 0 \ 0 \ 0.7071)^T$$

transformation into topic space with $k=2$

$$d6' = U_k^T \times d6 \approx (-0.3801 \ 0.2686)$$

$d6'$ is appended to V_k^T as a new column

LSI: Multilingual Retrieval

- ◆ Queries in one language can retrieve documents in the original and in other languages
- ◆ No query-translation is required

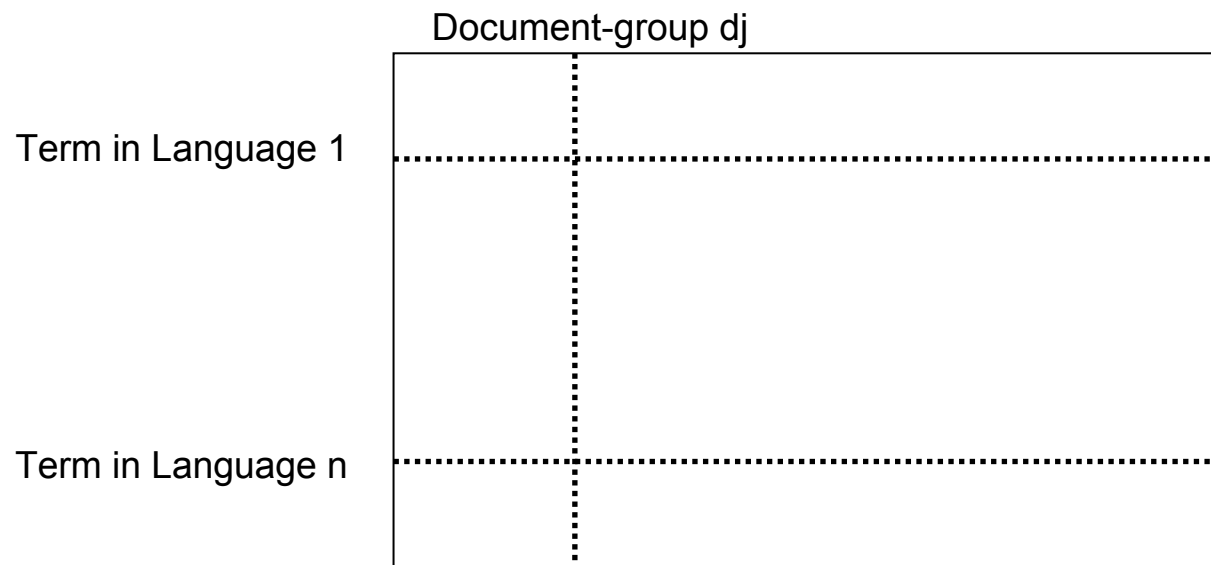
Multilingual corpora with corresponding documents needed

Examples:

- ◆ Wikipedia
- ◆ Reuters Collection
- ◆ FAO UN
- ◆ Sample of documents translated by human or machine

LSI: Multilingual Retrieval

- Construct LSI model (U_k, Δ_k, V_k^T) from training documents that are available in multiple languages:
 - Consider all language variants of the same document as a single document and
 - Extract all terms or words for all languages



- SVD with regression brings terms in different language together
- Folding in documents in one language

LSI: Multilingual Retrieval Example

Example:

d1: How to bake bread without bread recipes.

Wie man einfach Brot selber machen kann.

d2: Pastry: a book of best French recipes.

Gebäck: eine Sammlung der besten französischen Rezepte.

d3: The classic art of Viennese Pastry.

Klassisches Gebäck aus Wien.

Terms are e.g. bake, bread, recipe, back, Brot, Rezept

	Document 1	Document 2	Document 3
bake	0.4037	0	0
bread	0.8074	0	0
recipe	0.1490	0.3272	0
back	0	0.3272	1
Brot	0.4037	0	0
Rezept	0	0.8865	0

LSI: Multilingual Retrieval Example

$$\begin{aligned}
 A_2 = & \begin{pmatrix} 0.0365 & 0.3993 \\ 0.0729 & 0.7986 \\ 0.2140 & 0.1474 \\ 0.8068 & -0.1474 \\ 0.0365 & 0.3993 \\ 0.5434 & 0.0000 \end{pmatrix} & U \\
 & \times \begin{pmatrix} 1.1536 & & 0 \\ & 0 & 1.0000 \end{pmatrix} & \Delta \\
 & \times \begin{pmatrix} 0.1042 & 0.7071 & 0.6994 \\ 0.9891 & -0.0000 & -0.1474 \end{pmatrix} & V^T
 \end{aligned}$$

Folding-in

d4: Breads, pastries, pies and cakes: quantity baking recipes.

$$d4 = (0.5774 \quad 0.5774 \quad 0.5774 \quad 0 \quad 0 \quad 0)$$

$$d4' = (0.1868 \quad 0.7767)^T$$

LSI: Multilingual Retrieval Example

Query: Brot backen

$$q = (0 \ 0 \ 0 \ 1 \ 1 \ 0)$$

$$q' = (0.8433 \ 0.2519)$$

$$\text{sim}(q, d1) \approx 0.3370 \quad \text{sim}(q, d2) \approx 0.5963$$

$$\text{sim}(q, d3) \approx 0.5527 \quad \text{sim}(q, d4) \approx 0.3532$$

LSI: Multilingual Retrieval

DFG-Project: Multipla - Multi-Ontology Learning: Crossing the Boundaries of Domains and Languages

How can you get information in a language you don't know well?

- ◆ Translation?
- ◆ Construction of multilingual Ontologies?
- ◆ Information Retrieval?

Current Work

- ◆ Multilingual Retrieval with Wikipedia

LSI: Summary

- + Elegant, mathematically well-founded model
- + „Automatic learning“ of term correlations (incl. morphological variants, multilingual corpus)
- + Implicit thesaurus (by correlations between synonyms)
- + Implicit discrimination of different meanings of polysems (by different term correlations)
- + Improved precision and recall on „closed“ corpora (e.g. TREC benchmark, financial news, patent databases, etc.) with empirically best k in the order of 100-200
- In general difficult choice of appropriate k
- Computational and storage overhead for very large (sparse) matrices
- No convincing results for Web search engines (yet)

PLSI: Motivation and Key Idea

If you search for „Bank Überweisung“ do you want to retrieve documents about furniture?

LSI can't handle polysemous words but PLSI can deal with polysemous words and distinguish between different meanings and different types of words usage.

PLSI doesn't use approximation like LSI but solid statistical foundation and defines a proper generative data model.

PLSI is based on the likelihood principle.

Standard techniques from statistics can be applied for question like model fitting, model combination and complexit control.

PLSI: The Aspect Model

The Aspect Model is the core of PLSI.

Latent variable model associates unobserved class variables z_1, \dots, z_K with each observation, i.e. with each occurrence of a word w in a document d

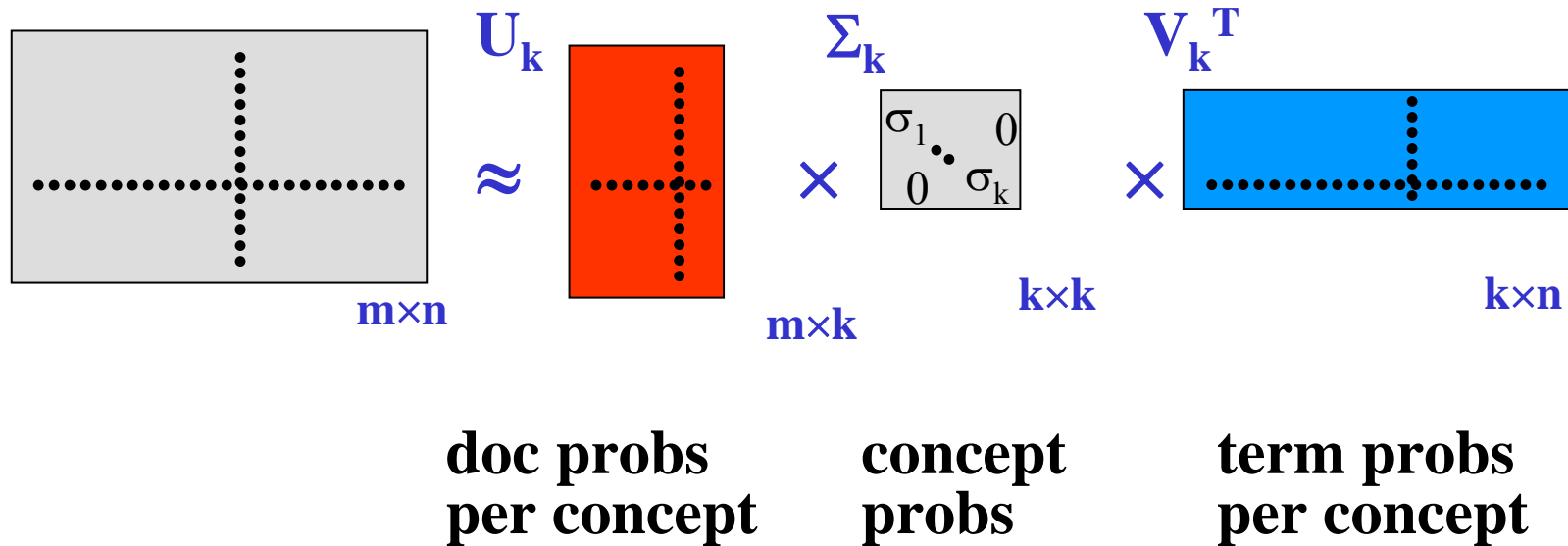
- Select a document d with probability $P(d)$
- Pick a latent class z with probability $P(z|d)$
- Generate a word w with probability $P(w|z)$

$P(w,d) = P(d)P(w|d)$ can be expressed via z

$$P(w|d) = \sum_z P(w|z)P(z|d)$$

$$P(w,d) = \sum_z P(z)P(w|z)P(d|z)$$

$$P[d, w] = \sum_z P[d/z] \cdot P[z] \cdot P[w/z]$$



Key difference to LSI:

- non-negative matrix decomposition
- with L1 normalization

PLSI: Expectation-Maximization Method (EM)

Key idea:

when $L(\theta, X_1, \dots, X_n)$ (where the X_i and θ are possibly multivariate) is analytically intractable then

- introduce *latent (hidden, invisible, missing) random variable(s) Z* such that
 - the *joint distribution $J(X_1, \dots, X_n, Z, \theta)$ of the „complete“ data* is tractable (often with Z actually being Z_1, \dots, Z_n)
- derive the incomplete-data likelihood $L(\theta, X_1, \dots, X_n)$ by *integrating (marginalization) J:*

$$\hat{\theta} = \arg \max_{\theta} \sum_z J[\theta, X_1, \dots, X_n, Z | Z = z] P[Z = z]$$

PLSI: EM Procedure

Initialization: choose start estimate for $\theta^{(0)}$

Iterate ($t=0, 1, \dots$) until convergence:

E step (expectation):

estimate posterior probability of Z : $P[Z | X_1, \dots, X_n, \theta^{(t)}]$
assuming θ were known and equal to previous estimate $\theta^{(t)}$,
and compute $E_{Z | X_1, \dots, X_n, \theta^{(t)}} [\log J(X_1, \dots, X_n, Z | \theta)]$
by integrating over values for Z

M step (maximization, MLE step):

Estimate $\theta^{(t+1)}$ by maximizing
 $E_{Z | X_1, \dots, X_n, \theta^{(t)}} [\log J(X_1, \dots, X_n, Z | \theta)]$

convergence is guaranteed

(because the E step computes a lower bound of the true L function,
and the M step yields monotonically non-decreasing likelihood),

but may result in local maximum of log-likelihood function

PLSI: EM at Indexing Time (PLSI Model Fitting)

observed data: $n(d,w)$ – absolute frequency of word w in doc d

model params: $P[z|d]$, $P[w|z]$ for concepts z , words w , docs d

maximize log-likelihood $\sum_d \sum_w n(d,w) \cdot \log P[dw]$

E step: posterior probability of latent variables

$$P[z|d,w] = \frac{P[z|d]P[w|z]}{\sum_y P[y|d]P[w|y]}$$

prob. that occurrence of word w in doc d can be explained by concept z

M step: MLE with completed data

$$P[w|z] \sim \sum_d n(d,w)P[z|d,w]$$

freq. of w associated with z

$$P[z|d] \sim \sum_w n(d,w)P[z|d,w]$$

freq. of d associated with z

actual procedure „perturbs“ EM for „smoothing“ (avoidance of overfitting) → tempered annealing

EM Details (PLSI Model Fitting)

$$P[z | d, w] = \frac{P[z | d]P[w | z]}{\sum_y P[y | d]P[w | y]} \quad (\text{E})$$

$$P[w | z] = \frac{\sum_d n(d, w)P[z | d, w]}{\sum_{d, u} n(d, u)P[z | d, u]} \quad (\text{M1})$$

$$P[z | d] = \frac{\sum_w n(d, w)P[z | d, w]}{\sum_{w, y} n(d, w)P[y | d, w]} \quad (\text{M2})$$

**or equivalently compute $P[z]$, $P[d|z]$, $P[w|z]$ in M step
(see S. Chakrabarti, pp. 110/111)**

PLSI: Folding-in of Queries

keep all estimated parameters of the pLSI model fixed
and treat query as a „new document“ to be explained

→ find concepts that most likely generate the query

(query is the only „document“, and $P[w | z]$ is kept invariant)

→ EM for query parameters

$$P[z | q, w] = \frac{P[z | q] \hat{p}[w | z]}{\sum_y P[y | q] \hat{p}[w | y]}$$

$$P[z | q] = \frac{\sum_w n(q, w) P[z | q, w]}{\sum_{w, y} n(q, w) P[y | q, w]}$$

PLSI: Query Processing

Once documents and queries are both represented as **probability distributions over k concepts**

(i.e. $k \times 1$ vectors with L1 length 1),

we can use any convenient vector-space similarity measure (e.g. scalar product or cosine or KL divergence).

PLSI: Example

d1: How to bake bread without bread recipes

d2: The classic art of Viennese Pastry

d3: Numerical recipes: the art of scientific computing

d4: Breads, pastries, pies and cakes: quantity baking recipes

d5: Pastry: a book of best French recipes

Start with:

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$P[z|d]$ and $P[w|z]$ are random stochastic matrices

PLSI: Example

Using this PLSI-implementation:

http://www.cs.bham.ac.uk/%7Eaxk/ML_CODE/PLSA.m

With 1000 iteration steps

$$P[z|d]= \begin{pmatrix} 0.0000 & 0.2335 \\ 0.4481 & 0.1827 \\ 0.0000 & 0.3503 \\ 0.0000 & 0.1168 \\ 0.5519 & 0.0000 \\ 0.0000 & 0.1168 \end{pmatrix}$$

$$P[w|z]= \begin{pmatrix} 0.0000 & 1.0000 & 1.0000 & 0.2392 & 1.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.7608 & 0.0000 \end{pmatrix}$$

PLSI: Summary

- + Probabilistic variant of LSI
(non-negative matrix factorization with L1 normalization)
- + Achieves better experimental results than LSI
- + Very good on „closed“, thematically specialized corpora,
inappropriate for Web
- Computationally expensive (at indexing and querying time)
 - may use faster clustering for estimating $P[d|z]$ instead of EM
 - may exploit sparseness of query to speed up folding-in
- pLSI does not have a generative model (rather tied to fixed corpus)
 - LDA model (Latent Dirichlet Allocation)
- number of latent concept remains model-selection problem
 - compute for different k , assess on held-out data, choose best

Additional Literature

Latent Semantic Indexing:

- Grossman/Frieder Section 2.6
- Manning/Schütze Section 15.4
- M.W. Berry, S.T. Dumais, G.W. O'Brien: Using Linear Algebra for Intelligent Information Retrieval, SIAM Review Vol.37 No.4, 1995
- S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman: Indexing by Latent Semantic Analysis, JASIS 41(6), 1990
- H. Bast, D. Majumdar: Why Spectral Retrieval Works, SIGIR 2005
- W.H. Press: Numerical Recipes in C, Cambridge University Press, 1993, available online at <http://www.nr.com/>
- G.H. Golub, C.F. Van Loan: Matrix Computations, John Hopkins University Press, 1996

pLSI and Other Latent-Concept Models:

- Chakrabarti Section 4.4.4
- T. Hofmann: Unsupervised Learning by Probabilistic Latent Semantic Analysis, Machine Learning 42, 2001
- T. Hofmann: Matrix Decomposition Techniques in Machine Learning and Information Retrieval, Tutorial Slides, ADFOCS 2004
- D. Blei, A. Ng, M. Jordan: Latent Dirichlet Allocation, Journal of Machine Learning Research 3, 2003
- W. Xu, X. Liu, Y. Gong: Document Clustering based on Non-negative Matrix Factorization, SIGIR 2003