LATENT MODELS FOR TEXT MINING

Francois Role - francois.role@parisdescartes.fr

September 2019

1 Introduction

In all methods presented in this chapter, the first goal is to create low rank approximations of the term-document (or document-term) matrix. More precisely, given a term-document $X_{mn}0$ with rank r, the goal is to find the best/nearest rank-k approximation of X. The second goal is to investigate the statistical process by wich the document-term matrix has arisen.

2 Singular Value Decomposition (SVD)

To decompose non square matrices, such as term-documents matrices, we can use the SVD technique.

In fact, any $m \times n$ matrix A can be factored into $A = U\Sigma V^T$ where the columns of U ($m \times m$) are the eigenvectors of AA^T, the columns of V ($n \times n$) are the eigenvectors of A^TA, and the diagonal matrix Σ contains the square roots of the eigenvalues of AA^T (or equivalently of A^TA) which are always nonnegative real values.

More precisely, let $A \in R^{m \times n}$ be a matrix of rank r. Then there exists an orthogonal matrix $U \in R^{m \times m}$, an orthogonal matrix $V \in R^{n \times n}$, and a diagonal matrix $\tilde{\Sigma} \in R^{m \times n}$ such that:

$$A = U\tilde{\Sigma}V^T \tag{1}$$

$$\tilde{\Sigma} = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \tag{2}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ contains the nonzero eigenvalues of AA^T.

For example, the following matrix:

$$\begin{pmatrix} 0. & 0.2 & 0. & 0. \\ 0. & 0. & 0. & 4. \\ 0. & 0. & 0. & 0. \end{pmatrix}$$

can be decomposed as:

$$U = \begin{pmatrix} 0. & 1. & 0. \\ 1. & 0. & 0. \\ 0. & 0. & 1. \end{pmatrix} \Sigma = \begin{pmatrix} 4. & 0. & 0. & 0. \\ 0. & 0.2 & 0. & 0. \\ 0. & 0. & 0. & 0. \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0. & 0. & 0. & 1. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 1. & 0. & 0. & 0. \end{pmatrix}$$

The columns of U and of V are the eigenvectors of AA^T and A^TA resp. the columns in U and V are called the left and right **singular vectors** of A resp.

Σ is a diagonal matrix whose (positive) elements are the squared roots of the eigenvalues λ_i of AA^T (or equivalently A^TA) usually arranged in decreasing order and denoted as $\sigma_i = \sqrt{\lambda_i}$. The σ_i values are called the **singular values** of A.

Let r be the rank of A (or equivalently of AA^T or A^TA). It is conventional to write Σ as an $r \times r$ matrix and to omit the rightmost $m - r$ columns of U and the rightmost $n - r$ columns of V.

Thus we end up with matrices U, Σ and V that are $m \times r$, $r \times r$ and $n \times r$ resp. The equivalent spectral version of this is:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

This is an **exact factorization**. In fact, any matrix is the sum of r matrices of rank 1. The first r columns of U (resp. V) give us an orthonormal basis for the column space (resp. row space) of X.

Continuing with our example, we have:

$$U = \begin{pmatrix} 0. & 1. \\ 1. & 0. \\ 0. & 0. \end{pmatrix} \Sigma = \begin{pmatrix} 4. & 0. \\ 0. & 0.2 \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0. & 0. & 0. & 1. \\ 0. & 1. & 0. & 0. \end{pmatrix}$$

Coming back to:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

we note that, as i increases, the contribution of the rank-1 matrix $u_i v_i^T$ decreases. So, if we zero out the smallest $r - k$ eigenvalues, we can expect that the resulting rank-k approximation will remain close to the original matrix:

$$A \approx \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Still continuing with our example, we have:

$$U = \begin{pmatrix} 0. \\ 1. \\ 0. \end{pmatrix} \Sigma = \begin{pmatrix} 4. \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0. & 0. & 0. & 1. \end{pmatrix}$$

and $U\Sigma V^T = \begin{pmatrix} 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 4. \\ 0. & 0. & 0. & 0. \end{pmatrix}$

which is close to:

$$\begin{pmatrix} 0. & 0.2 & 0. & 0. \\ 0. & 0. & 0. & 4. \\ 0. & 0. & 0. & 0. \end{pmatrix}$$

In fact, the SVD decomposition provides us with **the best possible rank-k approximation of a matrix**.

Some other important facts to keep in mind

Also note that we can take the squared roots of the eigenvalues thanks to the following facts: (1) for any matrix A the matrix AA^T is symmetric, positive semidefinite, (2) all eigenvalues of a symmetric matrix are real, (3) all eigenvalues of a positive semidefinite matrix are nonnegative, and (4) AA^T and A^TA have the same eigenvalues. Recall that when talking about the Frobenius norm we saw that $Trace(A^TA) = Trace(AA^T)$!

Recall that the rank of a diagonalizable matrix is the number of nonzero eigenvalues of this matrix. And it turns out that we deal with real symmetric (hence always diagonizable) matrices: AA^T and A^TA.

Last but not least, there is an important connection between the SVD and the spectral norm of a matrix.

Let λ_1 be the largest eigenvalue of a Gramm matrix A^TA. Being the largest eigenvalue of A^TA, λ_1 is then the maximum value of the Rayleigh quotient involving A^TA, and this maximum value is reached for the eigenvector x associated with this largest eigenvalue.

$$\lambda_1 = max_{x \neq 0} \frac{x^T A^T A x}{x^T x} \tag{3}$$

$$= max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} \tag{4}$$

$$= \|A\|_2^2 \tag{5}$$

The last equality follows from $\|A\|_2 = max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$. As a consequence we also have:

$$\|A\|_2 = (\lambda_1)^{1/2}$$

That is, **the (induced) 2-norm of a matrix equals its largest singular value**.

Remember, that for every eigenpair (λ, x) of $A^T A$ we have:

$$\lambda = \frac{x^T A^T A x}{x^T x}$$

Exercise

1. Compute the SVD of the following matrix X using the numpy.linalg.svd function. Understand the meaning of the *full_matrices* argument.

$$X = \begin{pmatrix} 0. & 0.2 & 0. & 0. \\ 0. & 0. & 0. & 4. \\ 0. & 0. & 0. & 0. \end{pmatrix}$$

2. Use the *TruncatedSVD* function from the *sklearn.decomposition* module to compute the SVD for a matrix computed from some text files stored in a directory.

2.1 Relation between Singular Value Decomposition and Matrix Diagonalization

Let a symmetric matrix A be decomposed as $A = U\Sigma V^T$. In this case, U is the matrix of the eigenvectors of A^2 and the same can be said of V since, A being symmetric, we have $AA^T = A^T A = AA = A^2$. Besides, Σ contains the square roots of the eigenvalues of A^2.

Since A^2 and A have the same eigenvectors and since the eigenvalues of A^2 equal the squares of the eigenvalues of A, this brings us back to the spectral decomposition $A = PDP^T$ where P is the matrix of the eigenvectors of A and D is the diagonal matrix of the eigenvalues of A.

Thus, note that the singular values of a symmetric matrix equal its eigenvalues (to a sign).

However, also note important differences between an eigendecomposition PDP^{-1} and a SVD.

The vectors in the eigendecomposition matrix P are not necessarily orthogonal (the change of basis isn't a simple rotation). On the other hand, the vectors in the matrices U and V in the SVD are orthonormal (they represent rotations).

In the SVD, the nondiagonal matrices U and V are not necessarily the inverse of one another. They are usually not related to each other at all. In the eigendecomposition the nondiagonal matrices P and P^{-1} are inverses of each other.

In the SVD the entries in the diagonal matrix σ are all real and nonnegative. In the eigendecomposition, the entries of D can be any complex number - negative, positive, imaginary, whatever.

The SVD always exists for any sort of rectangular or square matrix, whereas the eigendecomposition can only exist for square matrices, and even among square matrices sometimes it doesn't exist.

2.2 Relation between Singular Value Decomposition and PCA

A first important step is to recognize that $X^T X$ is proportional to the empirical sample covariance matrix of the dataset X.

In the lesson on PCA we saw that, assuming that the variables have been centered (column-wise zero empirical mean if the columns represent variables), the ample covariance matrix of X can be computed as $\frac{1}{N-1}X^TX$. We also saw that a loading vector v must be an eigenvector of the covariance matrix, that is $\frac{1}{N-1}X^TXv = \lambda v$. On the other hand, we saw at the beginning of this lesson, that a matrix X and its multiple cX have the same eigenvectors. So X^TX and $\frac{1}{N-1}X^TX$ have the same eigenvectors and the eigenvectors of X^TX, that is the right singular vectors of X forming the columns of V, are nothing else than the loading vectors of the PCA.

2.3 Application of SVD for Information Retrieval (LSA)

In Text Mining and in Information Retrieval it is usual to build low-rank approximations of document-term or term-document matrices. This can be done in the following way:

- Construct the SVD of A
- Build a copy Σ_k of Σ, in which the smallest entries in Σ have been replaced by zeros
- Compute $A_k = U\Sigma_k V^T$

The rank-k approximation of A is $A_k = \Sigma_{i=1}^{k}\sigma_i u_i v_i^T$, that is we truncate the spectral expansion after k terms.

A_k is the best rank k approximation of A: **it is the rank k approximation of A that best minimizes the Frobenius norm of $A - A_k$.**

Let X be a document term matrix. We have:

$$X = U\Sigma V^T \Leftrightarrow XV = U\Sigma \Leftrightarrow XV\Sigma^{-1} = U$$

In the same manner, we can derive V from U and X using:

$$X^T = V\Sigma U^T \Leftrightarrow X^TU = V\Sigma \Leftrightarrow X^TU\Sigma^{-1} = V$$

(Note that you could also use a term-document matrix as opposed to a document-term matrix, as here.)

Hence the "conceptual" representation of the documents (the U matrix) can be obtained by projecting the original representation of the documents into the term latent space V. This fact has many useful applications in many areas, for example in Information Retrieval where the technique has been presented under the name of "Latent Semantic Analysis" (LSA) by [**?**].

The following exercise will illustrate the main features of LSA.

Exercise
Let X be the following document-term matrix:

$$\begin{pmatrix} 1. & 1. & 1. & 0. & 0. \\ 2. & 2. & 2. & 0. & 0. \\ 1. & 1. & 1. & 0. & 0. \\ 5. & 5. & 5. & 0. & 0. \\ 0. & 0. & 0. & 2. & 2. \\ 0. & 0. & 0. & 3. & 3. \\ 0. & 0. & 0. & 1. & 1. \end{pmatrix}$$

cols = car vehicle truck cat dog

a) Determine the rank of X
b) Compute the eigenvectors of XX^T and use them as columns of the U matrix (documents)
c) Compute the eigenvectors of X^TX and use them as columns of the V matrix (terms)
d) Reconstruct the matrix. We can do it exactly. Why?
e) Check the distance between the original and reconstructed matrix.

The first document is represented in the latent space by the first row of matrix U :

[0.1796053 , 0.]

As said before, the relation between the original representation of the documents and their representation in the latent semantic space can be derived as follows:

$$X = U\Sigma V^T \Leftrightarrow XV\Sigma^{-1} = U$$

Hence the "conceptual" representation of the documents (the U matrix) can be obtained by projecting the original representation of the documents into the term latent space V.

Make sure that you have retrieved the two eigenvectors or AA^T.

Also verify that you can retrieve V from U and X.

2.3.1 Application to Query Processing in Information Retrieval

This leads to useful applications. For example, in the Information Retrieval area, suppose we are given a query q. We can treat q as a short document and project it into the latent space.

Using a standard query model, we would miss the second and fourth document altough they are related to the automotive sector.

The solution is to project q into the latent space. If q is represented as a row vector, the projected query $\hat{q}$ will be obtained as:

$$\hat{q} = qV\Sigma^{-1}$$

Exercise

Try to match the query to the documents but do it in the latent space. You should be able to match documents (second and fourth) that were about automotive area but had no terms in common with our query!

2.4 Why LSA works: Intuitive Explanation

Let $X = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ be a document-term matrix where the columns stand for words *cinema*, *movie*, $film$ in this order.

Let $q = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ be a query vector.

That is our query consists of the word $film$. Using X in its original form we won't match the first document altough it contains the word *movie*...

Now, form the matrix $X^T X = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

Now the terms are represented wrt each other, in the term space (just as documents). Then project the first document vector $X_{1:}$ on $X^T X$, giving the following representation for this document $\begin{pmatrix} 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 \end{pmatrix}$.

Document 1 has now been enriched: it has a nonzero entry for $film$ thanks to the cooccurrence between $film$ and *movie* in the second document.

The query will now match the first document.

$\begin{pmatrix} 2 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

3 Probabilistic Latent Models: from PLSA to LDA

In this section, the goal is to explain the document-term co-occurrences found in the corpus using **probabilistic latent models**.

Probabilistic latent models consider that the observed co-occurrence counts have been *generated* according to some probability laws, and try to estimate the parameters that mazimize the likelihood of the observed counts.

In this section, we suppose that we have a set of documents $D = \{d_1, \ldots, d_N\}$ in which occur terms from a vocabulary $W = \{w_1, \ldots, w_M\}$. **The observed data consist of all co-occurrences pairs** (w_j, d_i).

NOTE: there may be occasional variations in our notations. We generally denote topics using either t or z symbols. Also, the number of topics may be be denoted either by K or T.

3.1 PLSA

3.1.1 Relationship with LSA

PLSA is a probabilistic interpretation of LSA, which has been proposed by [**?**]. The goal is to explain the co-occurrence counts as part of a probabilistic framework. Using a latent variable, the probability that word w_j occurs in document d_i is expressed as:

$$
\begin{aligned}
p(d_i, w_j) &= p(d_i)p(w_j|d_i) && (6)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j, z_k|d_i) && (7)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j|z_k, d_i)p(z_k|d_i) && (8)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) && (9)\\
&= \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i)p(d_i) && (10)\\
&= \sum_{i=1}^{K} p(w_j|z_k)p(d_i|z_k)p(z_k) && (11)\\
&= \sum_{i=1}^{K} p(w_j|z_k)p(z_k)p(d_i|z_k) && (12)
\end{aligned}
$$

The rewriting of $p(w_j|z_k, d_i)$ into $p(w_j|z_k)$ is because w_j is supposed to be independent of d_i conditioned on the latent variable.

The last "symmetric" formulation, $p(d_i|z_k)$, $p(z_k)$, and $p(w_j|z_k)$ can be related to U, Σ, and V resp., hence the name PLSA: The rows of U and V are scaled so that **a document is a distribution over the topics**, **a topic is a distribution over words**).

The symmetric formulation also shows explicitly the fact that the document d_i and the word w_j are independent conditioned on z_k.

3.1.2 The generative Model

Another possible interpretation made possible by PLSA, is that a document is a sample of words drawn from a **probabilistic generative model**. The main advantage of this interpretation is that each document can be represented as a mixture of **topics** (the topics the document is about) and that each topic can be represented as a word distribution (allowing to spot the terms that best describe the topic). Here we are not motivated by theoretical considerations (merely "explaining" the co-occurrence but by more practical ones: we want to be able to assign a topic to each occurrence of a word in a document, while being able to know which words best describe each topic: we want to describe the **topical structure of a document**, which has many applications in the Text Mining field.

Looking back at the previous derivation and stopping it at an earlier stage, we see that the occurrence of a word w_j in a document d_i can indeed be seen as the result of a **generative process**.

$$p(d_i, w_j) = p(d_i)p(w_j|d_i) \tag{13}$$
$$= p(d_i) \sum_{i=1}^{K} p(w_j, z_k|d_i) \tag{14}$$
$$= p(d_i) \sum_{i=1}^{K} p(w_j|z_k, d_i)p(z_k|d_i) \tag{15}$$
$$= p(d_i) \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \tag{16}$$
$$\tag{17}$$

The generative process is as follows:

- First, document d_i has been selected with probability $p(d_i)$.
- Then, a latent class z_k has been sampled with probability $p(z_k|d_i)$.
- Finally, a word w_j has been generated with probability $p(w_j|z_k)$.

Given a vocabulary of M unique words, a corpus of N documents and K topics, The model therefore involves the following distributions:

- N multinomial distributions $p(z|d_i)$ $i = 1 \cdots N$ over the topics
- K multinomial distributions $p(w|z_k)$ $k = 1 \cdots K$ over the words
- one distribution $p(d)$ over the documents

ASIDE NOTE: Multinomial distributions are often used in text mining. In the BOW model a document d_j can be seen as a tuple $(n_1, \ldots, n_{|W|})$ where $n_i = tf_{ij}$ and $n_1 + \cdots + n_{|W|} = |d|$. If such a tuple (document) is seen in a corpus, it is supposed to have been sampled from a multinomial distribution $Mult(|d|; p_1, \ldots, p_{|W|})$.

To come back to PLSA, we thus have to find the $N \times K$ parameters $p(zk|d_i)$ and $M \times K$ parameters $p(w_j|z_k)$ that maximize the likelihood or log-likelihood of the corpus. The data likelihood is:

$$\prod_{i=1}^{N} \prod_{j=1}^{M} p(d_i, w_j)^{n(d_i, w_j)}$$

and the data log-likelihood is:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \; log \; p(d_i, w_j)$$

At this stage, the important point is to recall that $p(w_j|d_i)$ can be expressed as $\sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i)$.

This allows to express the data log-likelihood to be maximized as:

$$logL = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ p(d_i, w_j) \tag{18}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ \left(p(d_i) \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \right) \tag{19}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \left(log\ p(d_i) + log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \right) \tag{20}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ p(d_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \tag{21}$$

$$= \sum_{i=1}^{N} n(d_i)\ log\ p(d_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \tag{22}$$

$$= \sum_{i=1}^{N} n(d_i) \left(log\ p(d_i) + \sum_{j=1}^{M} \frac{n(d_i, w_j)}{n(d_i)} log\ \left(\sum_{k=1}^{K} p(z_k|d_i)p(w_j|z_k) \right) \right) \tag{23}$$

$$= \sum_{i=1}^{N} n(d_i) \left(log\ p(d_i) + \sum_{j=1}^{M} \frac{n(d_i, w_j)}{n(d_i)} log\ p(w_j|d_i) \right) \tag{24}$$

The last expression has been inclued to show that maximizing the likelihood is thus equivalent to minimizing the **cross-entropy** $-\sum_{j=1}^{M} \frac{n(d_i,w_j)}{n(d_i)} log\ p(w_j|d_i)$.

Looking again at the previous expression of the log-likelihood:

$$\sum_{i=1}^{N} n(d_i)\ log\ p(d_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i)$$

we note that $p(d_i)$ can be simply estimated as $p(d_i) = \frac{n(d_i)}{\sum_{l=1}^{N} n(d_l)}$, we thus concentrate our attention on the second part $\sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i)$ which is difficult to handle since it involves a logarithm of a sum. We therefore rewrite it so as to exhibit a lower bound (denoted as $L(q, \theta)$) which is easier to maximize and involves as its maximizable part the expectation of the complete data log-likelihood $E[L^c]$ (which serves to form the so-called Q-function $Q(\theta, \theta^{(t)})$:

$$E[L^c] = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} p(z_k|d_i, w_j) \log\ \left(p(w_j|z_k)p(z_k|d_i) \right)$$

EM can then be used to estimate $p(z_k|d_i)$, $p(z_k)$, and $p(w_j|z_k)$. $p(z_k|d_i)$ is the document-specific probability distribution over the latent variable space and $p(w_j|z_k)$ is the class-conditional probability of a word.

During the **E-STEP** we just compute the **the posterior probability of the latent variable**, using the current parameters $\theta^{(t)}$, that is we compute $p(z_k|w_j, d_i, \theta^{(t)})$

This posterior probability can be estimated using Bayes' rule as:

$$p(z_k|w_j, d_i) = \frac{p(z_k, w_j, d_i)}{p(w_j, d_i)} \tag{25}$$

$$= \frac{p(z_k, w_j|d_i)}{p(w_j|d_i)} \tag{26}$$

$$= \frac{p(w_j|z_k)p(z_k|d_i)}{\sum_{l=1}^{K} p(w_j|z_k)p(z_k|d_i)} \tag{27}$$

where we reuse the formula $p(w_j|d_i) = \sum_{l=1}^{K} p(w_j|z_k)p(z_k|d_i)$.

In the **M-STEP**, the expected complete data log-likelihood $E[L^c]$ to be maximized wrt the probability functions $p(w_j|z_k)$ and $p(z_k|d_i)$ is:

$$E(L^c) = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \log \prod_{k=1}^{K} \left(p(w_j|z_k)p(z_k|d_i)\right)^{p(z_k|d_i,w_j)} \tag{28}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} p(z_k|d_i, w_j) \log \left(p(w_j|z_k)p(z_k|d_i)\right) \tag{29}$$

$$\tag{30}$$

The posterior probability distribution is held fixed and to maximize the lower bound $L(q^{(t)}, \theta))$, we maximize $Q(\theta, \theta^{(t)})$ that is $\sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} p(z_k|d_i, w_j, \theta^{(t)}) \log \left(p(w_j|z_k)p(z_k|d_i)\right)$.

Maximizing the expected complete data log-likelihood (augmented with some Lagrange multipliers (not shown here) to account for probability constraints) wrt parameters $p(w\_j|z\_k)$ and $p(z_k|d_i)$ lead to the following reestimation equations:

$$p(w_j|z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j)}{\sum_{m=1}^{M}\sum_{j=1}^{N} n(d_i, w_m)p(z_k|d_i, w_m)} \tag{31}$$

$$p(z_k|d_i) = \frac{\sum_{j=1}^{M} n(d_i, w_j)p(z_k|d_i, w_j)}{n(d_i)} \tag{32}$$

$p(w_j|z_k)$ is therefore estimated as:

approximate number of ocurrences of term w_j in the whole corpus that are assigned to z_k / approximate number of ocurrences (in the whole corpus) assigned to z_k.

$p(z_k|d_i)$ is therefore estimated as:

approximate number of ocurrences in document d_i that are assigned to z_k / number of occurrences in d_i.

The limits of PLSA are as follows:

- The number of parameters to be estimated is $N + MK + KN$ (grows with the number of documents and prone to overfitting).
- PLSA is not really a generative model: there is no way of predicting the probability of an unseen document since the topic coverage distributions $p(z_k|d_i)$ are tied to the observed documents.

Exercise

(a) Prove that the posterior probability of the latent variables $p(z_k|w_j, d_i)$ can be expressed as :

$$p(z_k|w_j, d_i) = \frac{p(z_k, w_j|d_i)}{\sum_{l=1}^{K} p(w_j, z_l|d_i)}$$

(b)
A PLSA model with three topics has been fitted to a corpus of 100000 tokens. Given the following estimated probabilities
A PLSA model with three topics has been fitted to a corpus of 100000 tokens. Given the following

$$\text{estimated probabilities} \quad \begin{array}{l} p(w_1|z_1) = 0,1 \\ p(w_1|z_2) = 0,1 \\ p(w_1|z_3) = 0,2 \end{array}$$

$$\text{and} \quad \begin{array}{l} p(z_1|d_1) = 0,3 \\ p(z_2|d_1) = 0,3 \\ p(z_3|d_1) = 0,4 \end{array}$$

and also knowing that document d_1 contains 2000 tokens, which is the probability of the co-occurrence pair (w_1, d_1) ?

3.1.3 NMF and PLSA

Let X be a $m \times n$ word-to-document matrix where X_{ij} is the number of occurrences of word w_i in document d_j. We divide each cell of X by the total number of occurrences leading to a matrix $\tilde{X}$ such that $\sum_{i=1}^{m} \sum_{j=1} \tilde{X}_{ij} = 1$.

To find two non-negative matrices W and H such that $\tilde{X} = HW^T$ we try to minimize:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{X}_{ij} \log \frac{\tilde{X}_{ij}}{(WH^T)_{ij}} - \tilde{X}_{ij} + (WH^T)_{ij}$$

In contrast PLSA maximizes the likelihood:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{X}_{ij}\log P_{ij}$$

where $P_{ij} = p(w_i, d_j) = \sum_{i=1}^{K} p(d_i|z_k)p(z_k)p(w_j|z_k)$.

Maximizing $\sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{X}_{ij}\log P_{ij}$ is equivalent to minimizing:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} -\tilde{X}_{ij}\log P_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} -\tilde{X}_{ij}\log P_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{X}_{ij}\log\tilde{X}_{ij} \tag{33}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{X}_{ij}\log\frac{\tilde{X}_{ij}}{P_{ij}} \tag{34}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{X}_{ij}\log\frac{\tilde{X}_{ij}}{P_{ij}} - \tilde{X}_{ij} + P_{ij} \tag{35}$$

The first step amounts to adding a constant and the last step follows from $\sum_{i=1}^{m}\sum_{j=1}^{n}[P_{ij} - \tilde{X}_{ij}] = 1 - 1 = 0$.

We therefore retrieve the objective function for NMF.

3.2 Latent Dirichlet Allocation (LDA)

It is a widely used generative model in Text Mining. The initial version has been described in [**?**].

PLSA treats the conditional probabilities as parameters of the model. LDA, in contrast considers that the word-topic distribution Φ and the topic-document distribution θ are not parameters but are themselves generated from Dirichlet distributions with parameters α and β resp. (which is the reason why α and β are called "hyperparameters").

LDA borrows from PLSA the idea of a generative process.

For each document d, we have a multinomial topic distribution $\Theta_d \sim D(\alpha)$, which is drawn from a Dirichlet distribution with parameter α. $\Theta_{td} = p(z = t|d)$

For each topic t, we have a multinomial distribution of terms $\Phi_t \sim D(\beta)$ which is drawn from a Dirichlet distribution with parameter β. $\Phi_{vt} = p(w = v|z = t)$

In pratical terms, the main data structures are: - a matrix Θ of size $T \times D$ containing probabilities of topics given documents ($\Theta_{td} = p(z = t|d)$) , meaning that each column of Θ is a document-specific distribution, which sums to 1 :$\sum_t \theta_{td} = 1$. - a matrix Φ of size $W \times T$ containing probabilities of words given topics ($\phi_{vt} = p(w = v|z = t)$), meaning that each column of Φ is a topic-specific distribution of terms, which sums to 1 :$\sum_v \phi_{vt} = 1$

where D, W and T are the number of documents, words and topics resp.

α and β are *global* hyperparameters. Θ is a *document-level* variable. z and w are defined for each word.

3.2.1 Inference Techniques

In LDA, unlike PLSA, Φ and Θ (the conditional probabilities $\phi_{vt} = p(w_j|z_k)$ and $\Theta_{td} = p(z_k|d_i)$) are random variables generated from Dirichlet distributions and are inferred using variational Bayesian inference or Gibbs sampling.

Exercise

Analyze the NG20 dataset using the class *sklearn.decomposition.LatentDirichletAllocation* using 10 topics. Tune the model by tweaking the parameters (α, numer of iterations, etc.)

Exercise

Use Gensim implementation on a simple example.

```
from gensim import corpora from gensim.models.ldamodel import LdaModel
////////////////// Training
stoplist = ["a","is","many","wants","to","I","i","this","and"]
documents = ["Apple is releasing a new product", "Amazon sells things", "Apple
produces things", "Amazon provides new books", "Microsoft announces Nokia
acquisition", "IBM wants to acquire Nokia"]
train_documents=[ "I like to eat broccoli and bananas",
"I ate a banana and spinach smoothie for breakfast",
"Chinchillas and kittens are cute",
"My sister adopted a kitten yesterday",
"Look at this cute hamster munching on a piece of broccoli" ]
texts = [[word for word in document.lower().split() if word not in stoplist] for
document in train_documents]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
///corpus : pour chaque doc une liste de tuple (id_word,cnt_word)
/// [[(0, 1), (1, 1), (2, 1), (3, 1)], [(4, 1), (5, 1), (6, 1)],...]
lda = LdaModel(corpus=corpus, id2word=dictionary,  num_topics=2, update_every=1,
chunksize=10000, passes=1)
/// top terms of a topic
lda.print_topics(5)

////////////////// New documents
test_document="I like to eat spinach and fruits"
texts=[[word for word in test_document.lower().split() if word not in stoplist]]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
print(lda.get_document_topics(corpus[0])
```