LATENT MODELS FOR TEXT MINING

Francois Role - francois.role@parisdescartes.fr

September 2017

1 Introduction

In all methods presented in this chapter, the goal is to create low rank approximations of the term-document (or document-term) matrix. More precisely, given a term-document $X_{m\times n} \geq 0$ with rank r, the goal is to find the best/nearest rank-k approximation of X.

2 Eigenvectors and Eigenvalues

For an $N \times N$ matrix A, the scalar λ and the vector $\vec{x}$ such that:

$$A\vec{x} = \lambda\vec{x}$$

form an **eigenpair** $(\lambda, \vec{x})$ of A. λ is an **eigenvalue** of A and $\vec{x}$ is a (right) **eigenvector** of $\vec{x}$.

Let $\sigma(A) = \{\lambda_1, \cdots, \lambda_n\}$ be the set of all eigenvalues of a matrix $A \in C^{n\times n}$. Then the **spectral radius** of A denoted as $\rho(A)$ is defined as:

$$\rho(A) = \max\{|\lambda_1|, \cdots, |\lambda_n|\}$$

On the other hand, an eigenvalue of A that is larger in absolute value than any other eigenvalue is called the **dominant eigenvalue** (uniqueness!)

Every matrix has a spectral radius. Not every matrix has a dominant eigenvalue.

According to the Perron-Frobenius theorem for positive matrices, some matrices having the "Perron-Frobenius property" have a positive dominant eigenvalue with a corresponding positive eigenvector.

2.1 Some Important Particular Cases and Definitions

- Real identity matrix $I_{n\times n}$: eigenvalue 1 having for eigenvectors all vectors in R^n. This case shows that it is not necessary to have n distinct eigenvalues to have n distinct eigenvectors and hence be diagonalizable. If a matrix has n distinct eigenvalues then it has n distinct eigenvectors but the converse is not true.
- Real diagonal matrix $D_{n\times n}$: the eigenvalues are the elements on the diagonal and the corresponding eigenvectors are the vectors in the canonical basis of R^n.
- Real square matrix with positive entries: a unique largest real eigenvalue paired with an eigenvector which can be chosen to have strictly positive components (Perron-Frobenius eigenvalue, Perron root).

- Symmetric matrices: real eigenvalues and orthogonal eigenvectors with real-valued entries. More precisely, if a $n \times n$ matrix is symmetric it has n linearly independent eigenvectors (even if there are eigenvalues of multiplicity > 1). The X matrix formed by making its columns the eigenvectors is orthogonal.
- Semi-definite positive: non negative, real eigenvalues.
- Definite positive: positive, real eigenvalues.
- Hermitian matrices: real eigenvalues and real or complex eigenvectors.
- Covariance matrix of any random vector: symmetric positive semi-definite.
- Covariance matrix of a multivariate Gaussian: symmetric positive definite since in addition to being symmetric positive semidefinite must also be invertible.
- Stochastic matrix: always has an eigenvalue 1 whose associated eigenvector is a stationary probability vector.
- Simple eigenvector of a matrix A: an eigenvector whose the corresponding eigenvalue is not a multiple root of the characteristic equation for A.

2.2 Decomposition of Diagonalizable Square Matrices

A square, real-valued $N \times N$ matrix A with N linearly independent eigenvectors can be decomposed as:

$$A = U\Lambda U^{-1}$$

where the columns of U are the eigenvectors of A and Λ is a diagonal matrix having as entries the eigenvalues of A in decreasing order. Only square matrices that have a complete basis of eigenvectors can be factorized (diagonalized) in this way. However, real symmetric matrices are always diagonalizable by orthogonal matrices. More precisely, in the case when A is a symmetric $k \times k$ matrix, we have the **spectral decomposition**:

$$A = U\Lambda U^T \tag{1}$$
$$= \lambda_1 u_1 u_1{}', \ldots, \lambda_k u_k u_k{}' \tag{2}$$

The spectral decomposition is useful in many situations. As an example, we show below how it allows to prove that the matrix for a given quadratic form is **positive definite**.

Given the following quadratic form $3x_1^2 + 2x_2^2 - 2\sqrt{2}x_1x_2$, the corresponding matrix notation is $\begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 3 & -\sqrt{2} \\ -\sqrt{2} & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x'Ax$.

And we have to prove that $x'Ax > 0$ for all vectors $x \neq 0$.

The eigenvalues of A are the solutions of $|A - \lambda I\| = 0$ that is $(3 - \lambda)(2 - \lambda) - 2 = 0$, giving $\lambda_1 = 4$ and $\lambda_2 = 1$.

Now the matrix can be rewritten as $A = 4e_1e_1' + e_2e_2'$. We then have:

$$x'Ax = 4x'e_1e_1'x + x'e_2e_2'x = 4{y_1}^2 + {y_2}^2$$

where $y_1 = x'e_1 = e_1'x$ and $y_2 = x'e_2 = e_2'x$.

which already shows that $x'Ax \geq 0$ and therefore that A is at least **positive semi-definite** Since it can it can also be proved that y_1 and y_2 are strictly positive, it can be concluded that A is in fact positive definite. Let E be a matrix whos rows are e_1 and e_2. Since $y = Ex$, and E is orthogonal

thus has inverse E^T we have $E^T y = x$. Since $x \neq 0$, y cannot be the null vector, which concludes the proof.

Important note : this is just an example of a more general fact which is that for a symmetric matrix, being positive definite is equivalent to having all its eigenvalues positive.

2.3 The Rank of a Matrix

If A is an $n \times n$ matrix, the rank of A plus the **nullity** (the dimension of the kernel) of A is equal to n. So, the rank is n minus the dimension of the eigenspace corresponding to eigenvalue 0 (this eigenspace consists og the vectors x such that $Ax = 0x$). If 0 is not an eigenvalue, the matrix has full rank n.

rank = n - dim(kernel) with dim(kernel)=0 if 0 is not an eigenvalue.

The rank of A is the dim of the column space of A.

The number of non-zero eigenvalues of A is at most $rank(A)$.

If A is a diagonalizable matrix, then the rank of A is the number of nonzero eigenvalues of A (the number of nonzero entries in D, which is the number of eigenvalues of D).

A, A^T, $A^T A$ and AA^T have the same rank.

Exercise

1. What is the rank of the matrix A = [[1,2,3], [1,2,3], [1,2,3]] ?
2. Compute its eigenvalues.
3. Compute the images of [2,4,3] and [1,2,1] by A.
4. Let x be the column matrix [1,2,3]^T. What is the rank of xx^T
5. Compare the rank of A, AA^T and $A^T A$.

Exercise

2.3.1 Exercise

1. What is the rank of the matrix $\begin{pmatrix} 30 & 0 \\ 0 & \\ 0 & 20 \\ 0 & \\ 0 & 0 \\ 1 & \end{pmatrix}$
2. Compute its eigenvalues

The eigenvectors u_1, u_2, u_3 form a basis for R^3. Let $v = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = 2u_1 + 4u_2 + 6u_3$.

Compute Av.

2.4 Eigenpairs of a Multiple and of the Square of a Matrix

The multiple or the square of a matrix X has the same eigenvectors as X. However, the eigenvalues differ. This is illustrated in the following exercise.

Exercise

Given the matrix A:

$$\begin{pmatrix} 1 & 4 & 3 \\ 2 & 5 & 2 \\ 3 & 12 & 16 \end{pmatrix}$$

1. Compute the eigenpairs of A.
2. Compute the eigenpairs of $3 * A$. What can you guess?

Given the symmetric matrix B:

$$\begin{pmatrix} 1 & 4 & 3 \\ 4 & 5 & 9 \\ 3 & 9 & 16 \end{pmatrix}$$

3. Compute the eigenpairs of B.
4. Compute the eigenpairs of B^2. What can you guess?

3 Rayleigh Quotient, Dominant Eigenvalue and Power Method

If v is an eigenvector of A, $Av = \lambda v$ and then we get the **Rayleigh quotient**:

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

Given an eigenvector v the Rayleigh quotient allows us to find the corresponding eigenvalue.

As said before, if the eigenvalues of a $n \times n$ matrix A are such that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ then λ_1 is said to be the **dominant eigenvalue** of A.

Exercise

1. Does the following matrix $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ have a dominant value?
2. If a matrix has a dominant value does this value equal the spectral radius of the matrix?

3.1 The Power Method

if A is a diagonalizable matrix of size $n \times n$ with a dominant eigenvalue, this method allows to iteratively approximate the vector corresponding to this dominant eigenvalue (and this eigenvalue can then be derived from the found vector via the Rayleigh quotient as shown above).

It can be proved that if A is a diagonalizable matrix of size $n \times n$ with a dominant eigenvalue λ_1 then the power method will converge to a normalized vector corresponding to λ_1.

The normalized x_i vectors obtained at each step have the following form:

$$x_1 = \frac{Ax_0}{\|Ax_0\|_2}$$
$$x_2 = \frac{A\frac{Ax_0}{\|Ax_0\|_2}}{\left\|A\frac{Ax_0}{\|Ax_0\|_2}\right\|} = \frac{\frac{A^2x_0}{\|Ax_0\|_2}}{\left\|\frac{A^2x_0}{\|Ax_0\|_2}\right\|} = \frac{A^2x_0}{\|Ax_0\|_2}\frac{\|Ax_0\|_2}{\|A^2x_0\|_2} = \frac{A^2x_0}{\|A^2x_0\|_2}$$
$$\vdots$$
$$x_k = \frac{A^kx_0}{\|A^kx_0\|_2}$$

On the other end, since A is diagonalizable then it has n independent vectors $v_1, v_2, \cdots, v_n$ which form a basis and so x_0 can be expressed as $x_0 = c_1v_1 + c_2v_2 + \cdots + c_nv_n$ and then we have:

$$\begin{aligned} A^kx_0 &= A^k(c_1v_1 + c_2v_2 + \cdots + c_nv_n) = (c_1\lambda_1^kv_1 + c_2\lambda_2^kv_2 + \cdots + c_n\lambda_n^kv_n) \\ &= \lambda_1^k(c_1v_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \end{aligned}$$

Since $|\lambda_1|$ is the dominant eigenvalue of A, that is since the eigenvalues of a $n \times n$ matrix A are such that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, then A^kx_0 converge to $\lambda_1^kc_1v_1$ as k tends towards ∞. Therefore, as k tends towards ∞,$x_k = \frac{A^kx_0}{\|A^kx_0\|_2}$ tends towards $\frac{\lambda_1^kc_1v_1}{\|\lambda_1^kc_1v_1\|} = \frac{\lambda_1^kc_1}{|\lambda_1^kc_1|}\frac{v_1}{\|v_1\|} = \pm\frac{v_1}{\|v_1\|}$

In conclusion, the power method converges to a normalized vector corresponding to the dominant eigenvalue λ_1 since we have:

$$\lim_{k\to\infty} x_k = \pm\frac{v_1}{\|v_1\|}$$

4 Singular Value Decomposition (SVD)

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

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

$$A = U\tilde{\Sigma}V^T \tag{3}$$
$$\tilde{\Sigma} = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \tag{4}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$.

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 $r \times r$ 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.

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 $Trace(A^TA) = Trace(AA^T)$!

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 $n - r$ columns of U and the rightmost $m - r$ columns of V.

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

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

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

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.

$$\lambda_1 = max_{x \neq 0} \frac{x^T A^T A x}{x^T x} \tag{5}$$

$$= max_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} \tag{6}$$

$$= \|A\|_2^2 \tag{7}$$

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^TA we have:

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

4.1 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 $AA^T = A^TA = AA = A^2$. Besides, Σ contain 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 $A = PDP^{-1}$ and a SVD $A = U\Sigma V$

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 necessairily 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.

```
In [1]: ## SVD and eigendecomposition of a symmetric matrix
        # any symmetric matrix
        import numpy as np
        A= np.array([[1, 2, 3],
                     [2, 1, 2],
                     [3, 2, 1]])
        U, S,Vt= np.linalg.svd(A)
        print(S)
        dia, V= np.linalg.eigh(A)
        print(dia)

        # Gramm matrix
        A= np.array([[1, 0, 3],
                     [2, 1, 2],
                     [6, 2, 1]])

        ATA=A.T.dot(A)
        U, S,Vt= np.linalg.svd(ATA)
        print(S)
        dia, V= np.linalg.eigh(ATA)
        print(dia)
```

```
[ 5.70156212  2.          0.70156212]
[-2.         -0.70156212  5.70156212]
[ 50.82291452   9.           0.17708548]
[  0.17708548   9.          50.82291452]
```

4.2 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 $\frac{1}{N-1} X^T X$. We also saw that a loading vector v must be an eigenvector of the covariance matrix, that is $\frac{1}{N-1} X^T X v = \lambda v$. 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 $X^T X$ and $\frac{1}{N-1} X^T X$ have the same eigenvectors and the eigenvectors of $X^T X$, that is the right singular vectors of X forming the columns of V, are nothing else than the loading vectors of the PCA.

5 Applications of SVD

5.1 Computing principal components

Let A be an instance-feature matrix. The sample covariance matrix C_x of the centered data is given by:

$$C_A = \frac{1}{N-1} A^T A$$

and the (loadings of) the principal components are the eigenvectors of C_A. These eigenvectors can then be obtained using an eigen-decomposition of C_x.

On the other hand we know that when decomposing A as $U\Sigma V^T$, V is the matrix of the eigenvectors of $A^T A$, that is of $\frac{1}{n-1} A^T A$ (see exercise in the previous section), that is of C_A. We also know that the values in Σ are the square roots of the eigenvalues of $A^T A$. Therefore, if σ_i is a singular value of $A^T A$ then σ_i^2 is an eigenvalue of $A^T A$ and $\frac{\sigma_i^2}{n}$ is an eigenvalue of $\frac{1}{n-1} A^T A$, that is of C_A.

Thus, the principal components and corresponding eigenvalues can be found using an SVD of the original (centered) matrix.

Exercise

Let A be a 100×3 random matrix.

1. Center A and compute the covariance matrix C_A of the centered matrix.
2. Compute the eigen-decomposition of C_A.
3. Compute the SVD of A (which has been centered).
4. Compare the eigenvectors obtained in step 2 and the right singular vectors obtained in step 3.
5. Compute the eigenvalues obtained by the eigen-decomposition of C_A in step 2 from the singular values obtained in step 3.

5.2 Computing the 2-norm of a matrix

SVD provides a means to compute the (induced) 2-norm of a matrix:

$$\|A\|_2 = \sigma_1 = \sqrt{\lambda_1}$$

where λ_1 is the greatest eigenvalue of $A^T A$.

As for the Frobenius norm, it can also be computed using the singular values σ_i of A:

$$\|A\|_F = \sqrt{Tr(A^T A)} = \sqrt{Tr(A A^T)} = \sqrt{\left(\Sigma_{i=1}^r \sigma_i^2\right)}$$

by remembering that for a matrix A, $Tr(A)$ is the sum of its eigenvalues.

Exercise

Let A be the following matrix:

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{pmatrix}$$

Compute its 2-norm and its Frobenius norm using the above formula.

5.3 Information Retrieval (LSI)

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:

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

(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:

5.3.1 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!

5.4 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 $X^T X = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

and 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.

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

5.5 SVD and Tikhonov (ridge) regularization

Tikhonov proposed to introduce a regularization parameter in the minimization of squared residuals:

$$\|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$$

where Γ is the Tikhonov matrix which is often chosen to be a multiple of the identity matrix, $\Gamma = \alpha I$. This regularization, which gives preference to solutions with small norms, is also known as L2 regularization. The standard solution is:

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T \mathbf{b}$$

Note that when $\Gamma = 0$, we come back to the standard Ordinary least squares (OLS) solution.

However, with $\Gamma = \alpha I$, the least squares solution can be analyzed in a special way via the singular value decomposition. Given the singular value decomposition of A:

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

with singular values σ_i, the Tikhonov regularized solution can be expressed as :

$$\hat{x} = VDU^T b$$

where the general term of the diagional matrix D is:

$$D_{ii} = \frac{\sigma_i}{\sigma_i^2 + \alpha^2}$$

Exercise

Given a = [1., 2. , 3. , 4. ,5.,6] and b=[5.,7.,10.,11.,13.,14.], verify that:

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T \mathbf{b} = VDU^T b$$

6 Probabilistic Latent Models: from PLSA to LDA

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.

NOTE: 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 $W = \{w_1, \ldots, w_M\}$.

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.

6.1 PLSA

PLSA is a probabilistic interpretation of LSA, which has been proposed by [**?**].

It explains the co-occurrence counts by associating an unbserved variable $z_k \in \{z_1, \times, z_K\}$ with each occurrence of a word w_j in a document d_i:

$$\begin{aligned}
p(d_i, w_j) &= p(d_i)p(w_j|d_i) && (8)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j, z_k|d_i) && (9)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j|z_k, d_i)p(z_k|d_i) && (10)\\
&= p(d_i)\sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) && (11)\\
&= \sum_{i=1}^{K} p(d_i)p(w_j|z_k)p(z_k|d_i) && (12)\\
&= \sum_{i=1}^{K} p(d_i|z_k)p(z_k)p(w_j|z_k) && (13)
\end{aligned}$$

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

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

It suggests the **generative process** that explains the probability occurrence of word w_j in document d_i.

- 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 $p(w_j|z_k)$.

Atr the end of the trainig process we have estimated $p(w_j|z_k)$ and $p(z_k|d_i)$ probabilities: we are 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. In summary, we can describe the **topical structure of a document**.

Fundamentally, we are looking for parameters $p(w_j|z_k)$ and $p(z_k|d_i)$ that maximize the likelihood and therefore minimize the cross-entropy between $p(w_j|d_i)$

The likelihood of a corpus of N documents and M words is :

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

where $n(d_i, w_j)$ is the number of occurrences of the pair d_i, w_j. The data log-likelihood to be maximized is then:

$$logL = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ p(d_i, w_j) \tag{14}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ \left(p(d_i) \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \right) \tag{15}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \left(log\ p(d_i) + log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \right) \tag{16}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log\ p(d_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\ log \sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i) \tag{17}$$

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

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

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

In the above derivation, $n(d_i) = \sum_{j=1}^{M} n(d_i, w_j)$ is the document length. Maximizing the likelihood is equivalent to minimizing the **cross-entropy**:

$$-\sum_{j=1}^{M} \frac{n(d_i, w_j)}{n(d_i)} log\ p(w_j|d_i)$$

The parameters to be estimated using EM are the $ p(d_i)$, $ p(w_j| z_k)$, and $p(z_k|d_i)$.

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

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

Noting 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:

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

$$\geq \tag{22}$$

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

Can you explain why the inequality holds ?

EM is 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** the lower bound is maximized wrt $q(z_k)$ while holding parameters fixed. The solution is to choose $q(z_k)$ as the posterior probability of the latent variable, that is $p(z_k|w_j, d_i)$.

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

$$p(z_k|w_j, d_i) = \frac{p(z_k, w_j, d_i)}{p(w_j, d_i)} \tag{24}$$

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

$$= \frac{p(w_j|z_k)p(z_k|d_i)}{\sum_{i=1}^{K} p(w_j|z_k)p(z_k|d_i)} \tag{26}$$

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

In the **M-STEP**, the expected complete data log-likelihood $E(L^c)$ (also denoted as $Q(\theta, \theta_{(t)}$ to be maximized wrt the probability functions $p(w_j|z_k)$ and $p(z_k|d_i)$ is:

$$E(L^c) = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \log \prod_{k=1}^{K} \left(p(w_j|z_k)p(z_k|d_i)\right)^{p(z_k|d_i, w_j)} \tag{27}$$

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

$$\tag{29}$$

Note:it is interesting to compare the data log-likelihood:

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

and the expectation of the complete data log-likelihood :

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

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:

$$p(w_j|z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j)}{\sum_{m=1}^{M}\sum_{j=1}^{N} n(d_i, w_m)p(z_k|d_i, w_m)} \tag{30}$$

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

For a high $p(w_j|z_k)$ the intuitive meaning is that the occurrences of w_j in all documents very often can be assigned to the k-th topic with high probability compared to the occurrences of other words. For a high $p(z_k|d_i)$, the intuition is that most of the word occurrences in d_i can be assigned, with high probabiity, to the k-th topic.

The limits of PLSA are as follows:

- The number of parameters to be estimated is $M + MK + KN$ (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.

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
$p(w_1|z_1) = 0,1$
$p(w_1|z_2) = 0,1$
$p(w_1|z_3) = 0,2$
and
$p(z_1|d_1) = 0,3$
$p(z_2|d_1) = 0,3$
$p(z_3|d_1) = 0,4$
and also knowing that document d_1 contains 2000 tokens, which is the probability for word w_1 to show up in document d_1 ?

6.2 Latent Dirichlet Allocation (LDA)

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

6.2.1 Generative process

A document d is supposed to be generated as follows:

- Select a distribution over topics $\Theta_d|\alpha \sim Dir(\alpha)$
- For $i = 1 : N_d$
  - sample a topic from the document: $z_{di} \sim Mult(\Theta_d)$
  - sample a token from the topic : $w_{di}|z_{di} \sim Mult(\Phi_{z_{di}})$

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.

6.2.2 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.

- Go through each document, and randomly assign each word in the document to one of the K topics.
- For each document d_i
  - Go through each word w_j in d_i and for each topic z_k, compute
    - $p(z_k|d_i)$: the proportion of words in document d that are currently assigned to topic z_k.
    - $p(wordw_j|z_k)$: the proportion of assignments to topic z_k over all documents that come from this word w.
    - Reassign w_j a new topic z_k which maximizes $p(z_k|d_i)d) * p(p(wordw_j|z_k)$.

6.2.3 Comparison with other techniques

PLSA and LDA use a similar generative process. In LDA, the conditional probability of a word in a document can be denoted as:

$$p(w|d) = \sum_t p(w|t)p(t|d) = \sum_t \phi_{wt}\theta_{td}$$

This is the formulation we gave for $p(w|d)$ when describing PLSA:

$$\sum_{i=1}^{K} p(v|z=i)p(z=i|d)$$

An important theoretical difference is the following. PLSA treats the conditional probabilities as parameters. LDA, in contrast considers that the word-topic distribution Φ and the topic-document distribution θ are themselves generated from Dirichlet distributions with parameters α and β resp. (which is the reason why α and β are called "hyperparameters").

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.

6.3 NMF

In text mining we most often deal with nonnegative matrices. It is not natural to approximate such matrices with negative factors as with LSI. This obscures interpretation. Nonnegative matrix factorisation (NMF) solves this problem [**?**].

Given a matrix $A \in \mathbb{R}^{m\times n}$ we will then find a low-rank approximations $A = WH$ with only nonnegative factors. More precisely, we have to find $W \in \mathbb{R}^{m\times k}$ such that $W \geq 0$, and $H \in \mathbb{R}^{k\times n}$ such that $H \geq 0$ minimizing:

$$\|A - WH\|_F$$

Note that when W is fixed solving for $argmin_{H\geq 0}\|A - WH\|_F$ reduces to solving n least-square problems of the form $\|a_i - Wh_i\|$ where a_i and h_i denote the columns of A and H. For all i, the h_i minimizing $\|a_i - Wh_i\|$ is:

$$h_i = (W^TW)^{-1}W^Ta_i$$

So, if W is fixed, the H minimizing $\|A - WH\|_F$ is such that $H = (W^TW)^{-1}W^TA$.
This remark leads to a simple alternating least square (ALS) algorithm:

1. Guess an initial $W^{(0)}$.
2. for k in $0, 1, 2, \ldots$

- $H^{(k)} = argmin_{H\geq 0}\|A - W^{(k)}H\|_F$
- $W^{(k+1)} = argmin_{W\geq 0}\|A - WH^{(k)}\|_F$
- Set all negative elements in $W^{(k+1)}$ to 0.

Other algorithms are possible, based on gradient descent or multiplicative updates. In the original paper, the objective function to be minimized is:

$$\begin{aligned}
&\|X - ZW\|_F^2 && (32)\\
=&trace[(X - ZW)^T(X - ZW)] && (33)\\
=&trace[(X^T - W^T Z^T)(X - ZW)] && (34)\\
=&trace[(X^T X - X^T ZW - W^T Z^T X + W^T Z^T ZW)] && (35)\\
=&trace(X^T X) - 2trace(W^T Z^T X) + trace(W^T Z^T ZW) && (36)
\end{aligned}$$

6.4 Applications of NMF

- Clustering
- Image Processing
- Topic detection

6.5 NMF and PLSI

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 PLSI 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:

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=1}^{n} -\tilde{X}_{ij} \log P_{ij} &= \sum_{i=1}^{m} \sum_{j=1}^{n} -\tilde{X}_{ij} \log P_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{X}_{ij} \log \tilde{X}_{ij} && (37)\\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{X}_{ij} \log \frac{\tilde{X}_{ij}}{P_{ij}} && (38)\\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{X}_{ij} \log \frac{\tilde{X}_{ij}}{P_{ij}} - \tilde{X}_{ij} + P_{ij} && (39)
\end{aligned}$$

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.

6.5.1 Some Problems with NMF

- The convergence to a global minimum is not guaranteed and may be slow: to speed it up, smart initialisations can be used.
- Solutions are not unique: constraints have to be imposed.

Suppose we have found a solution (W, H) for the NMF problem. There exist matrices A and B such that $AB^T = I$, and matrices WA and HB are both nonnegative. So we have $(WA)(B^T H^T) = WH^T$ and so (WA, HB) is also a solution!

Expressing the columns of W and H as $W = (w_1, \ldots, w_k)$ and $H = (h_1, \ldots, h_k)$, we seek column-normalized versions of W and H, i.e. $W = (\tilde{w_1}, \ldots, \tilde{w_k})$ and $H = (\tilde{h_1}, \ldots, \tilde{h_k})$.

Let D_w and D_h be two diagonal matrices where $(D_w)_{ii} = \|w_i\|_p$ and $(D_h)_{ii} = \|h_i\|_p$ (if $p = 2$ this is the L-2 norm).

With $\tilde{W} = WD_w^{-1}$, $\tilde{H} = HD_h^{-1}$, and $S = D_w D_h$ we then have:

$$\tilde{W}S\tilde{H}^T = WD_w^{-1}D_w D_h D_h^{-1}H^T = WH^T \tag{40}$$

Exercise

Using the class *sklearn.decomposition.NMF* factorize the document-term matrix:

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

Then print the document-topic matrix, the term-topic matrix and, finally, reconstruct the initial matrix.