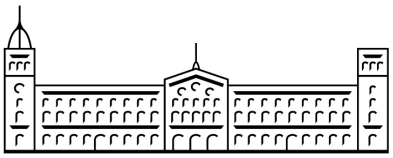


Senior project

BACHELOR OF MATHEMATICS

Faculty of Mathematics
University of Barcelona

A tensor based approach for temporal topic modeling

Author: Oriol Julià Carrillo

Advisor: Dr. Jordi Vitrià Marca
Conducted at: Department of Mathematics and Computer Science

Barcelona, June 26, 2016

Abstract

Latent Dirichlet Allocation (LDA) are a suite of algorithms that are often used for topic modeling. We study the statistical model behind LDA and review how tensor methods can be used for learning LDA, as well as implement a variation of an already existing method. Next, we present an innovative algorithm for temporal topic modeling and provide a new dataset for learning topic models over time. Last, we create a visualization for the word-topic probabilities.

Acknowledgments

This work is based on my stay at the University of California Irvine (UCI) during the first semester of the academic year 2015-16, and thanks to the Balsells mobility program, funded by the Balsells foundation. This is why I would like to start by thanking Pete Balsells and all those, like Roger Rangel, that make the Balsells fellowship program possible and my advisor at UCI, Animashree Anandkumar, as well as those that have been part of her team during my stay.

I would like to thank all my family for their consistent support during all my academic life, without which I wouldn't probably be where I am. They have made possible everything that I now take for granted and they deserve more than I will ever be able to give them back.

All those people that have been part of my life during my academic journey, specially those that have been closer, deserve to be mentioned. My friends have made my aspirations possible and have accompanied me during my best and worst moments, as well as have made me live great times.

Last, I would like to thank those professors that have had an impact in my academic life and what will be my professional life. In particular, Emtiyaz Khan, who introduced me to the exciting world of machine learning and Jordi Vitrià Marca, who has made the presentation of this project possible.

Contents

1 Introduction

Project summary

Topic modeling are a suite of algorithms that allow for discovering the abstract topics that occur in a collection of documents or corpus.

In topic modeling, a document is viewed as a bag of words, which is a representation that disregards grammar and even word order but keeps multiplicity (it consists of an enumeration of the words in a document and the number of times they appear in the text).

One of the state-of-the-art methods for learning a topic modeling problem is using Latent Dirichlet Allocation (LDA). LDA provides a Bayesian statistical approach to model the words in a corpus which is usually solved using efficient approximate inference techniques based on variational methods. LDA is usually used for document classification although it can alse be used for other applications such as image classification and even in the field of bioinformatics for gene function prediction.

During the recent years tensor methods have become popular for solving many machine learning problems. Particularly, tensor methods can be used to learn the parameters for a LDA model using a method of moments.

In this project we aim to review how tensor methods can be used for learning LDA. We then present a new Bayesian approach for learning many tensor decompositions simultaneously with correlated parameters where we assume that the eigenvectors of the different tensors follow a Gaussian distribution. One of its particular applications is very similar to an already successful method for collaborative filtering with a time component. We derive an algorithm for using this method to learn topic models over time.

Moreover, we provide an interactive representation of the word-topic matrix learned by LDA, which, created using D3, allows us to explore the convergence of tensor methods for learning LDA. We also observe a lack of opensource datasets for temporal topic modeling, reason why we provide a new bag of words dataset derived from an existing corpus of State of the Union addresses.

Structure of the report

The body of this report is structured in 5 chapters preceded by a first chapter which presents some basic notions on tensors as well as some practical facts, such as the notation and terminology we use during the report.

One chapter is dedicated to introducing the Latent Dirichlet Allocation model and another chapter explains how it can be learned using tensor methods. Then, another chapter is used to describe a new algorithm for tensor decomposition with correlated parameters. Finally, two chapters are dedicated to provide a new dataset for temporal topic modeling and an innovative visualization method for the word-topic probability matrix, respectively.

2 Notation, terminology and an introduction to tensors

A tensor is a multidimensional array $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$. Working with tensors often requires the use of some expressions that may not be familiar to all the readers. In order to make this report more amenable, we will try to use lowercase letters to denote vectors, uppercase letters to denote matrices and caligraphic letters (like $\mathcal{T}$ or $\mathcal{M}$) to denote tensors of order strictly greater than 2. Unfortunately, sometimes we will omit this convention in order to follow some other notation that is used in most of the publications or to make the text more understandable.

When dealing with probabilities, we will use $\propto$ to denote proportion: we say that $a \propto b$ if and only if $a = \alpha b$ for some $\alpha \in \mathbb{R} - \{0\}$. We will use $\otimes$ to denote the tensor product, which is also refered as Kronecker product when working with matrices and as outer product when when dealing with vectors. We may also use $\otimes^n$ to denote the tensor product of n equal tensors. For example. if $v \in \mathbb{R}^k$ is a vector, we will have $\otimes^3 v = v \otimes v \otimes v$.

We will also denote $[k] := \{1, \ldots, k\}$ and, given a matrix A, we will use $A^\top$ to denote its transpose.

2.1 Some products

Outer product

The outer product $u \otimes v$ is equivalent to a matrix multiplication $uv^\top$, provided that u is represented as a $m \times 1$ column vector and v as a $n \times 1$ column vector (which makes $v^\top$ a row vector).

For instance, if $m = 4$ and $n = 3$, then

$$u \otimes v = uv^\top = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}.$$

Or in index notation:

$$(uv^\top)_{ij} = u_i v_j$$

Kronecker product

If A is a $m \times n$ matrix and B is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the $mp \times nq$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

more explicitly:

$$A \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}$$

More compactly, we have

$$(A \otimes B)_{p(r-1)+v,q(s-1)+w} = a_{rs}b_{vw}$$

Tensor product

The tensor product can be understood as a generalization of the outer product or the Kronecker product to all kind of tensors. Given a tensor $\mathcal{A}$ of order q with dimensions $(i_1, ..., i_q)$, and a tensor $\mathcal{B}$ of order r with dimensions $(j_1, ..., j_r)$, their outer product $\mathcal{C}$ is of order $q + r$ with dimensions $(k_1, ..., k_{q+r})$ which are the i dimensions followed by the j dimensions. It is denoted in coordinate-free notation using $\otimes$ and components are defined index notation by:

$$\mathcal{C} = \mathcal{A} \otimes \mathcal{B}, \quad \mathcal{C}_{ij} = \mathcal{A}_i \mathcal{B}_j$$

Similarly, for higher order tensors:

$$\mathcal{T} = \mathcal{A} \otimes \mathcal{B} \otimes \mathcal{C}, \quad \mathcal{T}_{ijk} = \mathcal{A}_i \mathcal{B}_j \mathcal{C}_k$$

As an example, given three vectors $u \in \mathbb{R}^{n_1}, v \in \mathbb{R}^{n_2}, w \in \mathbb{R}^{n_3}$, we can calculate the tensor product $u \otimes v \otimes w \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ as the tensor $\mathcal{T}$ such that its entry (i, j, k) is $\mathcal{T}_{i,j,k} = u_i v_j w_k$.

Dot product

During the report we will also use the dot product and its generalization to higher order tensors. Given two vectors $u, v \in \mathbb{R}^n$, their dot product is defined as $\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i$. Note that this is a specific case of the inner product.

Given two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$, we will denote

$$\langle \mathcal{A}, \mathcal{B} \rangle := \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=1}^{n_k} \mathcal{A}_{i_1,\ldots,i_k} \mathcal{B}_{i_1,\ldots,i_k} = \sum_{i_1,\ldots,i_k=1}^{n_1,\ldots,n_k} \mathcal{A}_{i_1,\ldots,i_k} \mathcal{B}_{i_1,\ldots,i_k}$$

This last expression can be seen as the inner product of the two vectors obtained by vectorizing the tensors $\mathcal{A}$ and $\mathcal{B}$:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \langle vec(\mathcal{A}), vec(\mathcal{B}) \rangle$$

Hadamard product

In mathematics, the Hadamard product (also known as the Schur product or the entrywise product) is a binary operation that takes two matrices of the same dimensions, and produces another matrix where each element (i, j) is the product of elements (i, j) of the original two matrices.

We will extend the idea of the Hadamard product for matrices to a more general product for tensors. Given two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$, we will denote $\mathcal{A} \circ \mathcal{B}$ the tensor such that its entry $(i_1, \ldots, i_k)$ is

$$(\mathcal{A} \circ \mathcal{B})_{i_1,\ldots,i_k} := \mathcal{A}_{i_1,\ldots,i_k} \mathcal{B}_{i_1,\ldots,i_k}$$

2.2 An introduction to tensors

A tensor is a multidimensional array. More formally, an N-way or Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system. We can say that an Nth-order tensor is a tensor of order N and that the order of a tensor is its number of dimensions.

We ofter refer to 1st-order tensors as vectors and 2nd-order tensors as matrices, and tensors of order 3 or more are refered as higher-order tensors.

Definition 2.1 (Cubical tensor)**.** A tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ is a cubical tensor if $I_1 = \cdots = I_n$.

Definition 2.2 (Symmetric tensor)**.** A cubical tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ is a symmetric tensor (or supersymmetric tensor) if its elements remain constant under any permutation of the indices.

Equivalently, if the elements of the tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ are defined as $\mathcal{T}_{i_1 \ldots i_n} = x_{i_1 \ldots i_n}$, we will say the tensor $\mathcal{T}$ is symmetric if for any permutation σ the equality $x_{i_1 \ldots i_n} = x_{\sigma(i_1) \ldots \sigma(i_n)}$ is satisfied.

Definition 2.3 (Rank one tensor)**.** A tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ is a rank one tensor if it can be written as the outer product of n vectors, i.e.,

$$\mathcal{T} = v_1 \otimes \cdots \otimes v_n$$

where $\otimes$ denotes the outer product or tensor product.

Definition 2.4 (Rank of a tensor)**.** We say a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ has rank k if k is the minimum number such that there exists rank-one tensors that generate $\mathcal{T}$ as their sum, or equivalently, if k is the minimum number such that there exists rank-one tensors $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that

$$\mathcal{T} = \sum_{i=1}^{k} \mathcal{M}_i$$

.

It can easily be verified that the given definition of tensor rank is equivalent to the typical definitions of matrix rank we all know, but the properties of the tensor rank can be confusing for the non initiated reader. For example, the rank of a real-valued tensor may actually be different over the two fields $\mathbb{R}$ and $\mathbb{C}$ (see (TENBERGE, 1991) for an example and proof).

Another major difference between matrix and tensor rank is that, in general, there is no straighforward algorithm to determine the rank of a given tensor, and this problem is in general NP-hard.

CP Decomposition

In multilinear algebra, the tensor rank decomposition may be regarded as a generalization of the matrix Singular Value Decomposition (SVD) to tensors. It was introduced in 1927 and later rediscovered several times, reason why it is usually referred as CANDECOMP-PARAFAC (CP) decomposition.

Definition 2.5. Given a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ we define a CP decomposition of $\mathcal{T}$ as

$$\mathcal{T} = \sum_{i=1}^{k} \mathcal{M}_i$$

where k is a natural number and $\{M_i\}_{i=1}^{k}$ are rank-one tensors. When the number k is minimal in the above expression k is called the rank of the tensor.

For example, given a third order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, a CP decomposition of $\mathcal{T}$ would have the following form:

$$\mathcal{T} = \sum_{i=1}^{k} a_i \otimes b_i \otimes c_i$$

where $a_i \in \mathbb{R}^{I_1}$, $b_i \in \mathbb{R}^{I_2}$ and $c_i \in \mathbb{R}^{I_3} \quad \forall i \in \{1, \ldots, k\}$.

The specific tensor structure considered in this work is the symmetric orthogonal decomposition. This decomposition expresses a tensor as a linear combination of rank-1 tensors, which are the tensor product of a vector, and that form an orthogonal basis. An important property of tensors with such decompositions is that they have eigenvectors corresponding to these basis vectors (Anandkumar et al., 2014). This is why given an orthogonal decomposition of a symmetric tensor, $\mathcal{T} = \sum_{n=1}^{k} \otimes^m v_n$, we will denote the vectors v_n as the eigenvectors of $\mathcal{T}$.

3 Topic modeling and Latent Dirichlet Allocation

Topic modeling are a suite of algorithms that allow for discovering the abstract topics that occur in a collection of documents.

In topic modeling, a document is viewed as a bag of words, which is a representation that disregards grammar and even word order but keeps multiplicity (it consists of an enumeration of the words in a document and the number of times they appear in the text).

In general, for each document, a latent variable $h = (h_1, \ldots, h_k)$ represents the proportion of k topics in a given document, and given h the words are considered independent and exchangeable. Given l documents and assuming that d is the size of the vocabulary (i.e., the number of different words), the vectors $x_1, \ldots, x_l \in \mathbb{R}^d$ are used to represent the words on each document. The independence of the words given the topics implies that $\mathbb{E}[x_i|h] = \mu h$ and $\Pr(x_i|h_j) = \mu_j$, $\forall j \in [k]$, where $\mu := [\mu_1, \ldots, \mu_k] \in \mathbb{R}^{d \times k}$ is the word-topic matrix, whose entry i, j contains the probability of the word i belonging to the j topic.

3.1 Latent Dirichlet Allocation

One of the most famous topic models is Latent Dirichlet Allocation, well known as LDA and presented in (Blei et al., 2003). LDA is a three-level hierarchical Bayesian model based on the idea that documents are represented as random mixtures over latent topics, where each topics is characterized by a (multinomial) distribution over words.

LDA assumes that, for each document, the topics h are drawn independently from a Dirichlet distribution with concentration parameter $\alpha = [\alpha_1, \ldots, \alpha_k]$. The Dirichlet concentration (mixing) parameter is defined as $\alpha_0 := \sum_{i=1}^{k} \alpha_i$, and allows to specify the extent of overlap among the topics by controlling for sparsity in topic density function. The special case $\alpha_0 = 0$ is the single topic model whereas a larger α_0 results in more topics, making them be more overlapped.

The following generative process for each document $W = [w_1 \ \ldots \ w_N]$, where w_i represents a word, in a corpus D, is proposed:

1. Choose $\theta \sim Dirichlet(\alpha)$
2. For each word $w_i \in W$:
   (i) Choose a topic $z_i \sim Multinomial(\theta)$
   (ii) Choose a word w_i from $p(w_i|z_i, \beta)$, a multinomial probability conditioned on the topic z_i

where θ is a document-specific distribution over the available topics, z_i is the topic for word w_i, and α and β are hyperparameters for symmetric Dirichlet distributions from which the multinomial probability distributions are drawn.

The success of the model depends on several simplifying assumptions. First, the dimensionality k of the Dirichlet distribution (and thus the dimensionality of the topic variable z) is assumed to be known and fixed. Second, the word probabilities are represented by a fixed matrix $\beta \in \mathbb{R}^{V \times k}$ such that $\beta_{ij} = p(w_i|z_j)$, which we want to estimate, and that we call the word-topic matrix. Also, even though the initial model claims that the number of words N_d for each document is sampled from a Poisson distribution, this hypothesis is normally ignored and it is assumed that the length of each document is fixed.

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of N topics z and a set of N words w is given by:

$$p\left(\theta, z, w|\alpha, \beta\right) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{1}$$

A representation of LDA as a probabilistical graphical model is given in Figure 1. The three hierarchical levels correspond to the corpus, document and word variables. α and β are corpus-level variables, what means they are sampled once on each corpus. The variables θ are document-level variables, sampled once per document, and z_i and w_i are word-level variabels and are sampled once for each word in each document.

Note that LDA is a clear example of latent variable model, where we observe the words W and we want to recover the variables α, θ, z and β. We give a review of latent variable models in Appendix A.

It is also important to mention that LDA assumes that the words in the document are exchangeable, what is equivalent to saying that it is a bag of words model.

Recall that n random variables $x_1, \ldots, x_n$ are exchangeable if their joint probability distribution is invariant to permutations of the indices. A consequence of De Finetti's theorem is that such exchangeable models can be viewed as mixture models in which there is a latent variable h such that $x_1, \ldots, x_n$ are conditionally independent and identically distributed given h, with identical distributions at all the nodes.

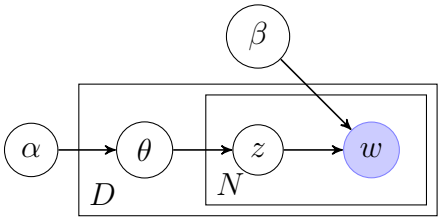


Figure 1: Graphical model representation of LDA. Note that w (in purple) is the only observed variable, while all the other variables are latent variables and are not initially observed. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

3.2 Inference and parameter estimation

The key inferential problem that needs to be solved in order to use LDA is that of computing the posterior distribution of the hidden variables given a document:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{2}$$

First of all, we want to analize the expression in the numerator. Examining the graphical model in Figure 1 we can see the correlations between the variables and obtain the following equality:

$$p(\theta, z, w|\alpha, \beta) = p(w|z, \beta)p(z|\theta)p(\theta|\alpha) \tag{3}$$

As $p(w|z, \beta)$ represents the probability of observing the document determined by the words w conditional on the topic of each word, which is determined by z, we can express $p(w|z, \beta) = \prod_{n=1}^{N} \beta_{w_n, z_n}$, where N represents the number of words in the document and β is the matrix that contains the probability of each word given each topic (the word-topic matrix).

Next, since $z_i \sim Multinomial(\theta)$, we have that $p(z_i|\theta) = \theta_j$, being j the topic that has been sampled from $z_i \sim Multinomial(\theta)$. Last, since $\theta \sim Dirichlet(\alpha)$, $p(\theta|\alpha)$ is determined by the Dirichlet probability density function:

$$p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}$$

where Γ denotes the Gamma function.

Bringing this all together and using Equation (3) the following expression is obtained:

$$p(\theta, z, w|\alpha, \beta) = \left(\frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \prod_{n=1}^{N} \beta_{w_n, z_n} \theta_{z_n} \tag{4}$$

where θ_{z_n} represents the component of θ chosen by z_n.

Using the encoding such that $z_{n,i} = 1$ iff the n^{th} word belongs to the i^{th} topic and it is 0 otherwise, and $w_{n,j} = 1$ iff the n^{th} word of the document is the j^{th} word of the vocabulary and it is 0 otherwise, it follows:

$$p(\theta, z, w|\alpha, \beta) = \left(\frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \prod_{n=1}^{N} \prod_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{j,i})^{w_{n,j} z_{n,i}} \tag{5}$$

where V denotes the number of words in the vocabulary and k is the number of topics. The denominator in Equation (2) is obtained by marginalizing over θ and z:

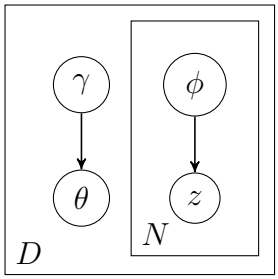


Figure 2: Graphical model representation of the variational distribution used to approximate the posterior in LDA.

$$p(w|\alpha,\beta) = \frac{\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)}{\prod_{i=1}^{k}\Gamma\left(\alpha_i\right)}\int\left(\prod_{i=1}^{k}\theta_i^{\alpha_i-1}\right)\left(\prod_{n=1}^{N}\sum_{i=1}^{k}\prod_{j=1}^{V}\left(\theta_i\beta_{j,i}\right)^{w_{n,j}}\right)d\theta \tag{6}$$

Unfortunately, this distribution is intractable to compute in general due to the coupling between θ and β, as seen in (Dickey, 1983). Despite exact inference not being possible for the problem, a wide variety of approximate inference have been derived, such as Gibbs Sampling or, originally, variational inference.

In the following two sections we will go through the original LDA inference procedure described in (Blei et al., 2003). Note that we just aim to give a review of the method and further details can be found in the previous paper.

3.2.1 Variational inference

The idea of variational inference is to use a simpler and convex distribution that obtains an adjustable lower bound on the log-likelihood of the actual distribution. The original graphical model is modified by dropping the problematic edges and nodes, what removes the coupling between θ and β that makes the inference intractable, and leads to the model represented in Figure 2. The new variational distribution is

$$q(\theta, z|\gamma,\phi) = q(\theta|\gamma)\prod_{n=1}^{k}q(z_i|\phi_i) \tag{7}$$

In particular, we want to determine the values of γ and ϕ. This can be done by finding a tight lower-bound on the log-likelihood, what results in solving the following optimization problem:

$$(\gamma*,\phi*) = \underset{(\gamma,\phi)}{\operatorname{argmin}}\, D\left(q(\theta,z|\gamma,\phi)||p(\theta,z|w,\alpha,\beta)\right) \tag{8}$$

which is a minimization of the Kullback-Leibler (KL) divergence between the variational distribution and the actual posterior distribution. This optimization

problem can be solved using a fixed point iteration method. In particular, computing the derivatives of the KL divergence and setting them equal to zero leads to the following pair of update equations:

$$\begin{aligned} \phi_{n,i} &\propto \beta_{w_n,i} exp\{\mathbb{E}_q\left[log(\theta_i)|\gamma\right]\} \\ \gamma_i &= \alpha_i + \sum_{n=1}^{k} \phi_{n,i} \end{aligned} \tag{9}$$

where $\mathbb{E}_q\left[log(\theta_i)|\gamma\right] = \Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)$ and Ψ is the first derivative of the $log\,\Gamma$ function, which can be approximated numerically using Taylor series.

It is important to note that we have omitted the dependence of the variational distribution on w. As the optimization problem defined in Equation (8) is done with a fixed w, the optimization parameters are, indeed, functions of w, $(\gamma^*, \phi^*) = (\gamma^*(w), \phi^*(w))$.

3.2.2 Parameter estimation

In the previous section we have assumed that α and β were fixed parameters, but we did not calculate them. To approximate the parameters α and β we will assume we know γ and ϕ. It is easy to observe the recursive contradiction that this aproach provides, but it can be solved using an Expectation-Maximization (EM) algorithm (Dempster et al., 1977) on the variational distribution.

The EM algorithm is a two step iterative algorithm. In our case, every iteration of the algoritm will be given by the following two steps:

- **E-Step**: Assume α and β are known and fixed. For each document d in the corpus, find the variational parameters (γ_d^*, ϕ_d^*) that solve the optimization problem defined in Equation (8).
- **M-Step**: Use the variational parameters found in the E-step to maximize the lower bound on the log-likelihood of $L(\alpha, \beta) = \sum_{d=1}^{M} log\ p(w_d|\alpha, \beta)$, where M is the number of documents.

3.3 LDA for document classification

The key point for document classification is finding an adequate feature space to represent the documents. The most straightforward approach is to threat individual words as features, obtaining a very rich but large set of features. However, the curse of dimensionality (Bellman and Bellman, 1961) will make this choice unsuitable.

In order to reduce the feature space we can use LDA for dimensionality reduction, assigning a set of real-valued features to each document. Precisely, we can set the variational parameter $\gamma_d^*(w)$ described in Section 3.2.2 to represent each document

d. Once the feature space has been set and the features of each document have been determined, any classifier such as Support Vector Machines (SVM) can be used to classify the documents.

This approach with all the details, as well as empirical experiments showing the great performance of the method, are described in (Blei et al., 2003).

3.4 LDA for collaborative filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (Wikipedia, 2016). The underlying assumption of the collaborative filtering approach is that if two users have the same or similar opinions on some issues, they are more likely to have a similar opinion on a new issue.

In collaborative filtering we first train a model on a fully observed set of users and we then aim to predict the preferences of an unobserved user. For the unobserved users, we are normally shown a subset of their preferences and we aim to predict what the unobserved elements are.

When using LDA for movie preference prediction, users and movies ara analogous to documents and words in a document. Using previous notation for topic modeling, the probability of a held-out movie is given by integrating over the posterior Dirichlet:

$$p(w|W_{obs}) = \int \sum_{z} p(w|z)p(z|\theta)p(\theta|W_{obs})d\theta \tag{10}$$

where W_{obs} contains all the observed movies and $p(w|W_{obs})$ can be efficiently computed using variational inference.

4 Learning LDA using tensor methods

Unsupervised learning for a wide range of latent variable models can be carried out efficiently via tensor-based techniques with low sample and computational complexities (Anandkumar et al., 2014). Unlike tensor-based methods, other approches such as expectation maximization (EM) and variational Bayes do not have such consistency guarantees.

In the topic modeling problem a latent variable $\theta = (\theta_1, \ldots, \theta_k)$ is interpreted as the mixture of topics for a document in a corpus. Given θ, the $l \geq 3$ words in a document can be sampled from a discrete distribution using the Latent Dirichlet Allocation model as described in Section 3.1. In this process, we sample a sole topic z_n for the n^{th} word from a cathegorical distribution (generalized Bernoulli distribution) from the mixture of topics θ. Then, we sample the n^{th} word x_n from topic z_n using the word-topic matrix, which contains the probability of each word given the topics.

Let v be the vocabulary size and $e_1, \ldots, e_v$ the standard coordinate basis for $\mathbb{R}^v$, we represent the l words in our document as l vectors $x_1, \ldots, x_l \in \mathbb{R}^v$ such that $x_m = e_i$ if and only if the m^{th} word in the vocabulary is i. The advantage of this notation is that it allows us to represent joint probabilities over words using the cross moments of the vectors $x_1, \ldots, x_n$. We observe that $\mathbb{E}[x_p \otimes x_q]_{ij} = P(x_p = e_1, x_q = e_j)$ or, more generally,

$$\mathbb{E}[x_1 \otimes \cdots \otimes x_l] = \sum_{1 \leq i_1, \ldots, i_l \leq v} P(1^{st} word = i_1, \ldots, l^{th} word = i_l)\ e_{i_1} \otimes \cdots \otimes e_{i_l} \quad (11)$$

As we have previously assumed that the number of words on each document is at least three ($l \geq 3$), we can now conclude that estimating joint probabilities of three words x_i, x_j, x_k over all documents is equivalent to estimating their cross moment $\mathbb{E}[x_i \otimes x_j \otimes x_k]$.

Thanks to the vector encoding we can now compute the conditional expectation of a word x_i given its topic $z_i = j$:

$$\mathbb{E}[x_i | z_i = j] = \sum_{m=1}^{v} P(m^{th}\ word = i | z_i = j)\ e_m = \beta_j \quad (12)$$

where β_j is the vector of word probabilities for topic j or, equivalently, the column j of the word-topic matrix β.

Now, due to exchangeability of the words and De Finetti's theorem, we recall that the words are conditionally independent given the topic, what leads to:

$$\mathbb{E}[x_{p_1} \otimes \cdots \otimes x_{p_r} | z = j] = \mathbb{E}[x_{p_1} | z_{p_1} = j] \otimes \cdots \otimes \mathbb{E}[x_{p_r} | z_{p_j} = j] = \beta_j \otimes \cdots \otimes \beta_j \quad (13)$$

This calculations lead to the following theorem:

Theorem 4.1. *(Anandkumar et al., 2012) If*

$$m_1 := \mathbb{E}[x_1]$$
$$M_2 := \mathbb{E}[x_1 \otimes x_2]$$
$$\mathcal{M}_3 := \mathbb{E}[x_1 \otimes x_2 \otimes x_3]$$

then

$$m_1 = \sum_{i=1}^{k} \frac{\alpha_i}{\alpha_0} \beta_i$$
$$M_2 = \sum_{i=1}^{k} \frac{\alpha_i}{\alpha_0} \beta_i \otimes \beta_i$$
$$\mathcal{M}_3 = \sum_{i=1}^{k} \frac{\alpha_i}{\alpha_0} \beta_i \otimes \beta_i \otimes \beta_i$$

where we denote the third order moment using the calligraphic letter $\mathcal{M}_3$ to emphasize that it is a tensor.

Now, as shown in (Anandkumar et al., 2014), we can estimate the topics β by computing a certain symmetric and orthogonal tensor decomposition. Precisely, we need to estimate the second and third order tensors M_2 and $\mathcal{M}_3$, for which we will give the empirical forms in Lemma 4.1. Moreover, due to exchangeability, all previous calculations for pairs or triples of words in a document can be used to form the empirical second and third order moments that estimate the matrix M_2 and the tensor $\mathcal{M}_3$, respectively.

4.1 Recovering the model

We now describe how the moments in Theorem 4.1 can be reduced to an orthogonal form. Denote $w_i := \frac{\alpha_i}{\alpha_0}$ so that we have

$$m_1 = \sum_{i=1}^{k} w_i \, \beta_i$$
$$M_2 = \sum_{i=1}^{k} w_i \, \beta_i \otimes \beta_i$$
$$\mathcal{M}_3 = \sum_{i=1}^{k} w_i \, \beta_i \otimes \beta_i \otimes \beta_i$$

We start by imposing that the word-topic matrix β has linearly independent columns and that the scalars $w_1, w_2, \ldots, w_k > 0$ are strictly positive. Note that the first condition is a very mild assumption since $\beta \in \mathbb{R}^{v \times k}$ is the word-topic matrix (and, in general, $v >> k$) and that the second one will always be true since $\alpha_i > 0$ for $i \in \{0, \ldots, k\}$ following the definition of the Dirichlet distribution. This conditions imply that the second order moment M_2 is positive semidefinite and has rank k.

Now, let $W \in \mathbb{R}^{v \times k}$ be a linear transformation such that $M_2(W, W) = W^\top M_2 W = Id$, where here Id is the $k \times k$ identity matrix (it is said that W whitens M_2). In

particular, the previous conditions imply that M_2 is positive definite and normal, what let us conclude that its SVD decomposition $M_2 = U\Sigma V^\top$ is equivalent to its eigenvalue decomposition. We may for concreteness take $W := U_k\Sigma_k^{-1}$, where $U_k \in \mathbb{R}^{v\times k}$ contains only the top singular vectors and $\Sigma_k \in \mathbb{R}^{k\times k}$ contains only the top singular values.

Let $\tilde{\beta}_i := \sqrt{w_i}W^\top\beta_i$. These vectors are orthonormal, as it can be seen using the following equalities:

$$M_2(W,W) = \sum_{i=1}^{k} W^\top \left(\sqrt{w_i}\beta_i\right)\left(\sqrt{w_i}\beta_i\right)^\top W = \sum_{i=1}^{k} \tilde{\beta}_i\tilde{\beta}_i^{\top} = Id \tag{14}$$

Now, define the whitened third order tensor $\mathcal{M}_3 \in \mathbb{R}^{k\times k\times k}$ so that

$$\mathcal{M}_3 = \sum_{i=1}^{k} \otimes^3 \left(W^\top\beta_i\right) = \sum_{i=1}^{k} \frac{1}{\sqrt{w_i}} \otimes^3 \tilde{\beta}_i \tag{15}$$

Theorem 4.3 in (Anandkumar et al., 2014) proves that the set of robust eigenvalues and eigenvectors of $\mathcal{M}_3$ are $\{\frac{1}{\sqrt{w_i}}\}_{i=1}^k$ and $\{\tilde{\beta}_i\}_{i=1}^k$, respectively, and that if $\left(W^\top\right)^\dagger$ is the Moore-Penrose pseudoinverse of $W^\top$ and v and λ are a robust eigenvector and eigenvalue of $\mathcal{M}_3$, respectively, then $\lambda\left(W^\top\right)^\dagger v = \beta_i$ for some $i \in [k]$.

4.2 Tensor forms for the empirical moments

Let $c_t := (c_{1,t}, \ldots, c_{v,t}) \in \mathbb{R}^v$ denote the frequency vector for the t^{th} document, and let n be the number of documents and v de vocabulary size. The vector c_t counts the number of occurrences of each word in a document, and the property of exchangeability makes the order of the words irrelevant. Hence, if l is the number of words in a document t and c_i the number of occurrences of word i in the document t, it follows that $\sum_{i=1}^{v} c_{i,t} = l$.

Calculating the empirical order moment can be computationally very expensive. In practice, one should use all of the words in a document for efficient estimation of the moments. One way to do this is to average over all $\binom{l}{3} \cdot 3!$ ordered triples of words in a document of length l. This method, presented in (Anandkumar et al., 2014), leads to the following lemma:

Lemma 4.1. *The first three order empirical moments are given by*

$$
\begin{aligned}
\widehat{m_1} &:= \frac{1}{n}\sum_{t=1}^{n} c_t \\
\widehat{M_2} &:= \frac{\alpha_0+1}{n}\sum_{t=1}^{n}\left(c_t \otimes c_t - \operatorname{diag}(c_t)\right) - \alpha_0 m_1 \otimes m_1 \\
\widehat{\mathcal{M}_3} &:= \frac{(\alpha_0+1)(\alpha_0+2)}{2n}\sum_{t=1}^{n}\Bigg[c_t \otimes c_t \otimes c_t - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i \otimes e_i \otimes e_j) \\
&\quad - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i \otimes e_j \otimes e_i) - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i \otimes e_j \otimes e_j) + 2\sum_{i=1}^{d} c_{i,t}(e_i \otimes e_i \otimes e_i)\Bigg] \\
&\quad - \frac{\alpha_0(\alpha_0+1)}{2n}\sum_{t=1}^{n}\Bigg(\sum_{i=1}^{d} c_{i,t}(e_i \otimes e_i \otimes m_1) + \sum_{i=1}^{d} c_{i,t}(e_i \otimes m_1 \otimes e_i) \\
&\quad + \sum_{i=1}^{d} c_{i,t}(m_1 \otimes e_i \otimes e_i)\Bigg) + \alpha_0^2 m_1 \otimes m_1 \otimes m_1.
\end{aligned}
\tag{16}
$$

Recall that the parameters α and β that are used in the first three moments of a LDA topic model (Theorem 4.1) can be recovered under a weak non-degeneracy assumption. We will employ the same tensor decomposition techniques used in (Huang et al., 2015) to learn the parameters.

4.3 Dimensionality reduction and whitening

One of the main difficulties to learn the model consists of working with such a large amount of data. In the case of the third order moment, $\widehat{\mathcal{M}_3}$, this has a size of order $O(v^3)$ where v is the size of the word vocabulary. In this regard, a whitening step is performed as presented in Section 4.1, in which $\widehat{\mathcal{M}_3}$ is transformed to another tensor that is orthogonal in expectation.

This whitening step leads also to dimensionality reduction, since it implicitly reduces the tensor $\widehat{\mathcal{M}_3}$ of size $O(v^3)$ to a new tensor of size $O(k^3)$, where k is the number of hidden topics, which we can assume to be much less than the size of the word vocabulary.

A whitening matrix $W \in \mathbb{R}^{v\times k}$ that satisfies $W^\top \widehat{M_2} W = Id$ is defined with the idea that if the bilinear projection of the second order moment onto W is the identity matrix, then a trilinear projection of the third order moment onto W would result in an orthogonal tensor.

First of all a k-truncated Singular Value Decomposition (SVD) for the second order moment $\widehat{M_2}$ is computed, $\widehat{M_2} = U_{\widehat{M_{2,k}}}\Sigma_{\widehat{M_{2,k}}}U^\top_{\widehat{M_{2,k}}}$, where $U_{\widehat{M_{2,k}}}$ and $\widehat{M_{2,k}}$ contain only the first k singular vectors and singular values of $\widehat{M_2}$, respectively. Then, the whitening matrix is computed

$$W = U_{\widehat{M_{2,k}}} \Sigma_{\widehat{M_{2,k}}} \tag{17}$$

and used to calculate the whitened data

$$y_x := W^\top c_x \in \mathbb{R}^k \tag{18}$$

where c_x is the word frequency vector for document x. Using the whitened samples instead of all the frequency vectors, a whitened third order tensor can be calculated using the expression in Lemma 4.1 and replacing each c_t by y_x. From now on, we will refer to this whitened empirical third order moment as $\mathcal{T}$.

Note, however, that performing this whitening step can be computationally very expensive. If carried out naively, it will involve performing a SVD decomposition of the empirical second order moment, a matrix $\widehat{M_2} \in \mathbb{R}^{v \times v}$ where v is the vocabulary size of the corpus, which is normally many tens of thousands. As calculating a SVD decomposition of $\widehat{M_2}$ has, in general, $O(v^3)$ complexity, this computation will be a difficult problem to tackle. Moreover, storing $\widehat{M_2} \in \mathbb{R}^{v \times v}$ will also be a challenging aspect to solve.

A way to efficiently compute the whitening matrix W is to use thin random projections of the data to estimate a thin version of $\widehat{M_2}$ and to take advantage of the fact that we only need to calculate a truncated SVD decomposition of $\widehat{M_2}$ (we don't need to calculate the full SVD decomposition), as explained in (Huang et al., 2015).

4.4 From variational inference to tensor methods

In *Section* 3.2, we provided an overview of how variational methods can be used for learning LDA. The key problem we need to solve in order to recover the model is that of recovering the parameters α and β. Recall that the word-topic matrix, β, is such that $\beta_{ij} = P(word_i|topic_j)$, and gives the probability of each word on each of the topics.

When using variational inference, we recover α and β using an Expectation-Maximization algorithm, what allows us to simultaneously recover the variational parameters for each of the documents d, (γ_d^*, ϕ_d^*).

Using tensor methods we can first recover the parameters α and β by using a method of moments, what determines the LDA model and discovers the topics of a corpus. Then, we can use a variation of the EM algorithm described in *Section* 3.2 to estimate the variational parameters (γ_d, ϕ_d) for each document. Specifically, we can run the same EM algorithm skipping the M-step, as we assume we already know the parameters α and β. This approach will let us compute a feature set for each document, γ_d, which can be used for document classification as described in Section 3.3.

4.5 Learning using third order moments

The implementation of an algorithm to learn the parameters of a LDA model using tensor decomposition and up to third-order moments is described in (Huang et al., 2015). Stochastic gradient descent (Kushner and Yin, 2003) is used to solve the optimization problem.

The process to learn the parameters for a single LDA model consists of:

1. Estimate the second order empirical moment $\widehat{M_2}$
2. Perform pre-processing using the second-order moment $\widehat{M_2}$
3. Compute the empirical third order moment using whitened data
4. Perform tensor decomposition using *stochastic gradient descent* on the whitened third order empirical tensor
5. Apply post processing to obtain the word-topic matrix and the Dirichlet parameters

Stochastic Tensor Gradient Descent

We want now to estimate the empirical third order tensor $\mathcal{T}$ obtained using whitened data, and our goal is to find its CP decomposition:

Definition 4.1. Our optimization problem is given by

$$\underset{v_j, j\in[k]}{\operatorname{argmin}} \left\| \frac{1}{n_X} \sum_{x\in X} \mathcal{T}_x - \sum_{j=1}^{k} \otimes^3 v_j \right\|_F^2 + \theta \left\| \sum_{j=1}^{k} \otimes^3 v_i \right\|_F^2 \tag{19}$$

where v_i are the unknown components to be estimated, and $\theta > 0$ is some fixed parameter.

In order to encourage orthogonality between the eigenvectors of the third order tensors the extra term $\theta \left\| \sum_{j=1}^{k} \otimes^3 v_i \right\|_F^2$ is added, where the fixed parameter $\theta > 0$ weighs its relevance. This is motivated in (Huang et al., 2015) and we provide an explanation for it later.

Define the matrix of the empirical eigenvectors of $\mathcal{T}$, $V := [v_1| \dots |v_k]$.

Lemma 4.2. *The optimization problem given in Definition 4.1 is equivalent to minimizing a loss function $L(V) := \frac{1}{n_x} \sum_{x\in X} L_x(v)$, where $L_x(V)$ is the loss function evaluated at a document x that belongs to a set of documents X, and is given by*

$$L_x(V) := (1+\theta) \left\| \sum_{j=1}^{k} \otimes^3 v_j \right\|_F^2 - 2 \left\langle \sum_{j=1}^{k} \otimes^3 v_j,\ \mathcal{T}_x \right\rangle \tag{20}$$

Proof. We have

$$\begin{aligned}
&\left\|\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x - \sum_{j=1}^{k}\otimes^3 v_j\right\|_F^2 + \theta\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 = \\
&= \left\|\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x\right\|_F^2 - 2\left\langle\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle + \left\|\sum_{j=1}^{k}\otimes^3 v_j\right\|_F^2 + \theta\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 \\
&= (1+\theta)\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 - 2\left\langle\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle + \left\|\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x\right\|_F^2
\end{aligned} \tag{21}$$

Now, since $\left\|\sum_{x\in X}\mathcal{T}_x\right\|_F^2$ is a constant, the equation to optimize in $Definition$ 4.1 becomes

$$\begin{aligned}
&(1+\theta)\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 - 2\left\langle\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle = \\
&(1+\theta)\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 - 2\sum_{x\in X}\left\langle\frac{1}{n_x}\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle = \\
&\sum_{x\in X}\left(\frac{1}{n_x}(1+\theta)\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 - 2\left\langle\frac{1}{n_x}\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle\right) = \\
&\frac{1}{n_x}\sum_{x\in X}\left((1+\theta)\left\|\sum_{j=1}^{k}\otimes^3 v_i\right\|_F^2 - 2\left\langle\mathcal{T}_x,\ \sum_{j=1}^{k}\otimes^3 v_j\right\rangle\right) = \frac{1}{n_x}\sum_{x\in X}L_x(V)
\end{aligned} \tag{22}$$

□

We recall that we use $\langle\mathcal{A},\mathcal{B}\rangle$ to denote the sum of the entries of the elementwise product of $\mathcal{A}$ and $\mathcal{B}$.

Lemma 4.3. *The partial derivatives of $L_x(V)$ are:*

$$\frac{\partial L_x(V)}{\partial v_j} = 6\left[(1+\theta)\sum_{p=1}^{k}\langle v_p, v_j\rangle^2 v_p - \sum_{i=1}^{k}\left(v_j^\top \mathcal{T}_{x,(:,:,i)} v_j\right)\otimes e_i\right] \tag{23}$$

Proof. For each of the two terms in the sum, we will do the calculations elementwise. Note that in order to reduce the complexity of the notation, and as all the sums are finite, we will omit some superscripts in this proof. Also, we will use the convention that $\mathcal{T}_{x,(:,:,t)}$ denotes the matrix whose element in the position i,j is $\mathcal{T}_{x,(i,j,t)}$, the element in the position i,j,t of $\mathcal{T}_x$, and that $v_{i,m}$ denotes the m^{th} element of the vector v_i, which is also the i^{th} column of the matrix V.

We first have:

$$\left\|\sum_{j=1}^{k}\otimes^3 v_j\right\|_F^2 = \sum_{i,j,r}\left(\sum_p v_{p,i}v_{p,j}v_{p,r}\right)^2 = \left(\sum_p v_{p,t}^3\right)^2 + 3\sum_{i\neq t}\left(\sum_p v_{p,i}v_{p,t}^2\right)^2 + 3\sum_{i,j\neq t}\left(\sum_p v_{p,i}v_{p,j}v_{p,t}\right)^2 + \sum_{i,j,r\neq t}\left(\sum_p v_{p,i}v_{p,j}v_{p,r}\right)^2 \tag{24}$$

what let us compute its derivatives:

$$\begin{aligned}\frac{\partial}{\partial v_{k,t}}\left(\left\|\sum_{j=1}^{k}\otimes^3 v_j\right\|_F^2\right) &= 6\left[v_{k,t}^2\left(\sum_p v_{p,t}^3\right) + 2\sum_{i\neq t} v_{k,i}v_{k,t}\left(\sum_p v_{p,i}v_{p,t}^2\right)\right.\\ &\left. + \sum_{i,j\neq t} v_{k,i}v_{k,j}\left(\sum_p v_{p,i}v_{p,j}v_{p,t}\right)\right] = 6\sum_{i,j} v_{k,i}v_{k,j}\left(\sum_p v_{p,i}v_{p,j}v_{p,t}\right)\\ &= 6\sum_{p,i,j} v_{k,i}v_{k,j}v_{p,i}v_{p,j}v_{p,t} = 6\sum_p v_{p,t}\left(\sum_j v_{p,j}v_{k,j}\right)\left(\sum_i v_{p,i}v_{k,i}\right)\\ &= 6\sum_p \langle v_p, v_k\rangle^2 v_{p,t}\end{aligned} \tag{25}$$

For the second term of the sum, we get a similar expression:

$$\begin{aligned}\left\langle\sum_{i\in[k]}\otimes^3 v_i, \mathcal{T}^x\right\rangle_t &= \sum_{i,j,r}\left(\mathcal{T}_{x,(i,j,r)}\sum_p v_{p,i}v_{p,j}v_{p,r}\right) = \mathcal{T}_{x,(t,t,t)}\sum_p v_{p,t}^3\\ &+ 3\sum_{i\neq t}\left(\mathcal{T}_{x,(i,t,t)}\sum_p v_{p,i}v_{p,t}^2\right) + 3\sum_{i,j\neq t}\left(\mathcal{T}_{x,(i,j,t)}\sum_p v_{p,i}v_{p,j}v_{p,r}\right)\\ &+ \sum_{i,j,r\neq t}\left(\mathcal{T}_{x,(i,j,r)}\sum_p v_{p,i}v_{p,j}v_{p,r}\right)\end{aligned} \tag{26}$$

which let us, again, compute the derivative:

$$\begin{aligned}&\frac{\partial}{\partial v_{k,t}}\left(\left\langle\sum_{i\in[k]}\otimes^3 v_i, \mathcal{T}_x\right\rangle\right) =\\ &= 3\left[\mathcal{T}_{x,(t,t,t)}v_{k,t}^2 + 2\sum_{i\neq t}\mathcal{T}_{x,(i,t,t)}v_{k,i}v_{k,t} + \sum_{i,j\neq t}\mathcal{T}_{x,(i,j,t)}v_{k,i}v_{k,j}\right]\\ &= 3\sum_{i,j} v_{k,i}\mathcal{T}_{x,(i,j,t)}v_{k,j} = 3v_k^\top\mathcal{T}_{x,(:,:,t)}v_k\end{aligned} \tag{27}$$

Last, using the linearity of the derivatives, we obtain:

$$\frac{\partial L_x(V)}{\partial v_{k,t}} = 6\left[(1+\theta)\sum_p \langle v_p, v_k\rangle^2 v_{p,t} - v_k^\top T_{x,(:,:,t)} v_k\right] \tag{28}$$

We note that the derivative of the first term can be calculated for all the entries of the vector v_k at the same time, what leads to a more general formula:

$$\frac{\partial L_x(V)}{\partial v_k} = 6\left[(1+\theta)\sum_p \langle v_p, v_k\rangle^2 v_p - \sum_i \left(v_k^\top T_x(:,:,i) v_k\right) \otimes e_i\right] \tag{29}$$

□

Now, using the derivative of the optimization function we get the iterative update for the stochastic gradient descent:

$$\tilde{v}_{i,(n+1)} \leftarrow \tilde{v}_{i,(n)} - \alpha_n \frac{\partial L_x(V)}{\partial v_i}\bigg|_{\tilde{V}_{(n)}}, \ \forall i \in [k] \tag{30}$$

where α_n is the learning rate at the n^{th} iteration, the derivatives are calculated for each document $x \in X$, $\tilde{v}_{i,(n)}$ contains the learnt eigenvector i at iteration n and $\tilde{V}_n := \left[\tilde{v}_{1,(n)} \ \dots \ \tilde{v}_{k,(n)}\right]$.

Note that the first term of the derivatives is determined by the dot product between pairs of eigenvectors, weighted by $6(1+\theta)$, and will reduce its norm when the eigenvectors are orthogonal, what justifies the regularization term $\theta \left\| \sum_{j=1}^k \otimes^3 v_i \right\|_F^2$ in Definition 4.1.

Remark 4.1. *Using the linearity of the derivatives we obtain*

$$\frac{1}{n_x}\sum_{x\in X} \frac{\partial L_x(V)}{\partial v_i} = \frac{\partial\left(\frac{1}{n_x}\sum_{x\in X} L_x(V)\right)}{\partial v_i} = \frac{\partial L(V)}{\partial v_i} \tag{31}$$

What let us conclude that we can use our stochastic approach, which calculates the derivatives for each document x, to calculate the full derivatives too. While using the full derivatives may give us faster convergence, they may be too expensive to calculate when dealing with large datasets. In this regard, we may be interested in using an stochastic approach where on each iteration we choose in which datapoint or datapoints (document or documents) we evaluate the derivatives depending on a random distribution. The approach is the following:

1. Choose N, the number of documents to evaluate the derivatives
2. Sample $X_N \sim U(k)$, where $U(K)$ represents a discrete uniform distribution with N different possible outputs and X_N is a set containing the chosen N documents where we will calculate the derivatives.

3. Calculate the derivatives $\frac{1}{|X_N|}\sum_{x\in X_N}\frac{\partial L_x(V)}{\partial v_i}$ on the N randomly chosen datapoints using Lemma 4.3.

4. Update the eigenvectors using Equation (30) and the calculated derivatives

Note that even when we can easily calculate the derivatives using all the datapoints we may be interested in using a stochatic gradient descent approach instead of a gradient descent with the full derivatives. This is because the randomness we add on the derivatives when we use stochastic gradient descent can help our eigenvectors escape from saddle points, which have null gradient, or even from local minima that are far away from the best solution.

Indeed, further analysis of stochastic gradient descent for tensor reconstruction can be found in (Ge et al., 2015), where it is proved that stochastic gradient descent will avoid saddle points in optimization problems like ours and will converge to a local minimum in a polynomial number of iterations with respect to the number of topics k. Note also that our problem doesn't have a unique global minimum, since the topics are exchangeable and any permutation of them is an equally good solution.

4.6 Post-processing

Finally we do an inverse process to the one carried out in the whitening step. First, we compute the eigenvalues $\Lambda := [\lambda_1, \lambda_2, \ldots, \lambda_k]$ as the norm of the estimated eigenvectors: $\lambda_i = \|\tilde{v}_i\|^3$, $i \in \{1, \ldots, k\}$.

Then, we make an inverse process to the one carried out in Equation (18) on the whitening step to recover an estimation of the word-topic matrix

$$\hat{\beta} = W^{\top\dagger}V \tag{32}$$

where $W^{\top\dagger}$ denotes the Monroe pseudo inverse of the transpose of the whitening matrix W.

Last, we obtain the Dirichlet distribution parameters

$$\hat{\alpha_i} = \gamma^2\lambda_i^{-2}, \quad \forall i \in [k]. \tag{33}$$

where γ^2 is chosen such that we have a normalization $\sum_{i\in[k]}\hat{\alpha}_i := \sum_{i\in[k]}\frac{\alpha_i}{\alpha_0} = 1$.

4.7 Implementation

We provide a modification of Furong Huang's C++ code for learning LDA using tensor methods https://github.com/FurongHuang/TensorDecomposition4TopicModeling The main difference is that in our code we use stochastic gradient descent, as described in the previous sections of this report, instead of alternating least squares, as described in (Anandkumar et al., 2014) for calculating an orthogonal CP decomposition of the whitened empirical third order moment of the LDA model.

The code uses the Eigen[1]library to implement linear algebra structures and operations (specially useful to deal with sparse matrices and vectors). While making the modification of the code may seem simple and straightforward, it has been very time consuming due to the lack of explicative comments on the code, which consists of more than 2000 lines of C++ code splited in many different files. Debugging has also been a tedious task as compiling the code lasts around 30 seconds.

The code can be devided in the following sections:

1. Read the corpus in bag-of-words format
2. Estimate the secod order moment
3. Perform dimensionality reduction and whitening
4. Compute the whitened third order empirical moment and get a CP decomposition of it
5. Post-processing: use the estimated rank one components (or eigenvectors) and the eigenvalues to recover the LDA model (Dirichlet parameter α and word-topic matrix β).
6. Perform variational inference on a testing dataset using the learnt LDA model

It is also worth mentioning that the inference section of the code is a modified version of the lda-c code in https://github.com/blei-lab/lda-c which implements the variational inference algorithm described in Section 3.2 and originally presented in (Blei et al., 2003).

4.7.1 Code release

Our code is released on https://github.com/ourii/Tensor_LDA_STGD

In our published version we have also modified the code to easily accept as input many corpus and learn many LDA models in the same run of the program. This provides the basics for the implementation of Algorithm 1, presented in subsequent pages of this report.

4.7.2 Computing the gradients

Computing the derivatives in Lemma 4.3 can be computationally expensive if carried out naively. This is why we suggest an implementation that uses mainly vectorized and matrix operations, what will perform significantly faster in many programming languages such as Matlab, R or when using the Eigen library in C++.

We recall we want to compute the following derivatives:

[1]eigen.tuxfamily.org

$$\frac{\partial L_x(V)}{\partial v_j} = 6\left[(1+\theta)\sum_p \langle v_p, v_j\rangle^2 v_p - \sum_i \left(v_j^\top \mathcal{T}_x(:,:,i)v_j\right)\otimes e_i\right] \tag{34}$$

We first aim to calculate $\sum_p \langle v_p, v_j\rangle^2 v_p$ simultaneously for all the vectors v_j, $j \in [k]$. Denote $V = [v_1|\dots|v_k]$. This calculation can be efficiently carried out in the following way:

$$\left[\sum_p \langle v_p, v_1\rangle^2 v_p \,|\dots|\, \sum_p \langle v_p, v_k\rangle^2 v_k\right] = V\left(V^\top V \circ V^\top V\right) \tag{35}$$

As for the second term in Lemma 4.3, $\sum_i \left(v_k^\top \mathcal{T}_{x,(:,:,i)} v_k\right)\otimes e_i$, it can be calculated row by row. Its i^{th} row is the column-wise sum of the matrix $\left(\mathcal{T}_{(:,:,i)}^\top V\right)\circ V$, where we recall that $\circ$ is used to denote the entry-wise product, as described in Section 2.1.

Note that we base our gradients on theoretical calculations. However, the derivatives are the key point of a gradient descent method (otherwise we would be converging to the minimum of another optimization function). Hence, not only do we want to make sure we are not making any mistake in the calculation of the derivatives, but also that our implementations are correct and our C++ functions are accurate.

A common approach for this problem is to make a numerical verification of the gradient by using the definition of derivative. Theoretically, we know that the value of the symmetric derivatives is the following:

$$\left[\frac{\partial L_x(V)}{\partial v_k}\right]_m = \lim_{h\to 0}\frac{L_x\left(V + H(m,k,h)\right) - L_x\left(V - H(m,k,h)\right)}{2h} \tag{36}$$

where $H(m,k,h)$ is a matrix such that

$$H(m,k,h)_{ij} = \begin{cases} V_{ij} + h & \text{if } (i,j) = (m,k) \\ V_{ij} & \text{if } (i,j) \neq (m,k) \end{cases} \tag{37}$$

Note that we have prefered to use the symmetric derivatives, and approximate the derivatives using centered differences instead of the ordinary derivatives, because this method has the same computational cost but it can be proved using Taylor's formula that the result is much more accurate (the error is $O(h^2)$ instead of $O(h)$ for twice twice differentiable functions).

In practise, most of the times we can get a good estimation of the gradients by computing the derivatives element-wise following Equation (36) and taking h sufficiently small (for example, $h \approx 10^{-4}$). While this approach is numerically vulnerable and may lead to bad estimations depending on the function L_x, it provides a fast and easy way for verifying the derivatives and their implementation. It is also easy to improve its accuracy by estimating the derivatives using higher order methods.

It is importat to remark that two reasons make numerical methods for calculating the derivatives not preferable when the gradients can be calculated and have a closed form. First, computing the derivatives numerically may lead to bad estimations in some cases (depending on the function for which we are calculating the derivatives). Second, the computational cost of computing the derivatives numerically is, in general, much greater than that of calculating them using a closed form. Indeed, the error in the estimations can only be reduced using higher order methods which require more evaluations of the optimization function and are slower.

We have implemented numerical verification for the gradients of the optimization function in Lemma 4.2 to verify that the derivatives in Lemma 4.3 and our posterior implementation are correct.

4.7.3 Results

We run our code on the New York Times bag of words dataset (Lichman, 2013). As in (Huang et al., 2015), we set the fixed parameters to be $\alpha_0 = 1$ and $\theta = 1$. We run the code with 50 topics ($k = 50$). The top words (those that have a higher probability on each topic according to the estimated word-topic matrix) for the 4 topics that we find more significative are represented in Table 1. Note that we have labeled each of the presented topics, but this was not part of the outcome of the algorithm, as LDA is, in general, an unsupervised method.

We also wanted to evaluate the extra time needed to compute the gradients using first order numerical methods compared to using the exact derivatives (recall that both methods are explained in Section 4.7.2).

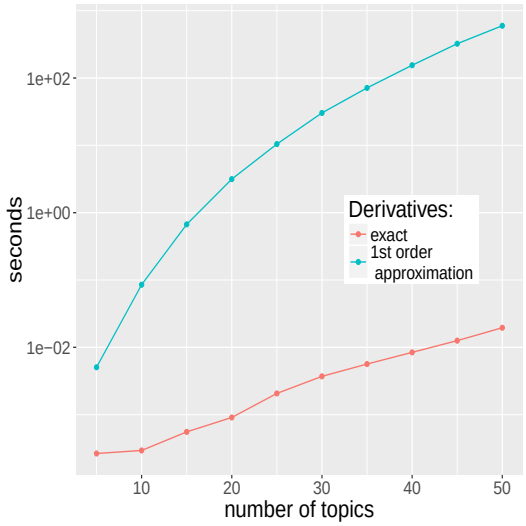
A graphical presentation of the results, plotted in R by means of the library *ggplot2*[2], can be found in Figure 3. It let us conclude that while learning our third order structures can be done efficiently using exact methods to calculate the derivatives, this becomes an unattainable problem if we use numerical methods instead. Note also that we are only using a very simple first order method to estimate the derivatives, but getting more robust results using higher order methods would require to evaluate the loss function at more points and would slow down the method significantly.

[2]http://ggplot2.org/

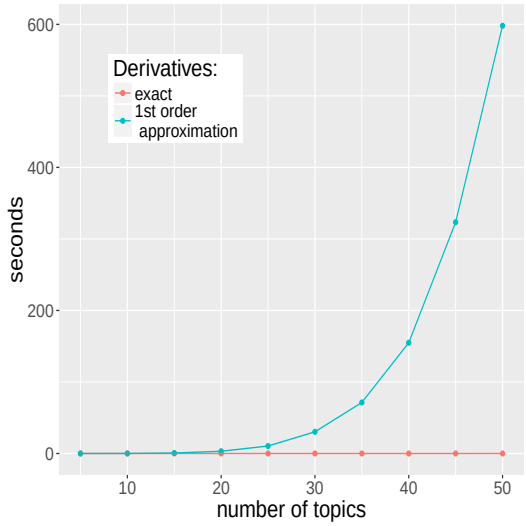
"business"	*"sport"*	*"Israel"*	*"War on terror"*
million	game	company	zzz_bush
percent	team	palestinian	tonight
company	newspaper	newspaper	official
question	zzz_boston_globe	women	attack
file	player	drug	team
slugged	season	zzz_israel	palestinian
shares	play	zzz_killed	season
com	school	zzz_mandatory_kill	newspaper
spot	spot	premature	military
web	zzz_held	guard	terrorist
billion	tax	zzz_los_angeles_daily_new	question
site	games	book	zzz_israel
official	zzz_bush	school	zzz_afghanistan
onlytest	coach	send	file
money	www	leader	zzz_taliban
companies	com	percent	fall
government	home	race	zzz_diane
zzz_al_gore	cut	zzz_united_states	zzz_israeli
quarter	won	student	zzz_united_states
sales	percent	military	show
stock	win	war	onlytest
business	point	released	zzz_u_s
www	run	zzz_israeli	bin
fund	tonight	zzz_yasser_arafat	laden
zzz_xxx	file	earlier	war

Table 1: Top words of some of the recovered topics learning LDA using tensor methods and stochastic gradient descent.

Figure 3: Time required to compute the derivatives on each step of the stochastic gradient descent, using 10 documents on each iteration



(a) Number of topics vs computational time (seconds), both in linear scales.



(b) Number of topics vs computational time (seconds), the first in linear scale and the second in *log* scale.

5 A tensor decomposition with correlated parameters

Assume we are given n tensors $\mathcal{T}^{(t)} \in \mathbb{R}^{d_1 \times \cdots \times d_l}$, $t \in [n]$, of some order l and that each tensor $\mathcal{T}^{(t)}$ has an approximate CP decomposition of rank k, namely,

$$\mathcal{T}^{(t)} = \sum_{j=1}^{k} \lambda_j^{(t)} \, a_{1j}^{(t)} \otimes \cdots \otimes a_{lj}^{(t)} + N^{(t)}, \tag{38}$$

for some $a_{ij}^{(t)} \in \mathbb{R}^{d_i}$, $\lambda_j^{(t)} \in \mathbb{R}$ and $N^{(t)}$ a noise tensor which we assume to be sufficiently small. To refer to the vectors $\{\{a_{ij}^{(t)}\}_{j=1}^{k}\}_{i=1}^{l}$ we will use the matrices $A_i^{(t)} = \begin{bmatrix} a_{i1}^{(t)} & \dots & a_{ik}^{(t)} \end{bmatrix} \in \mathbb{R}^{d_i \times k} \;\; \forall i \in [l]$ and define $\lambda^{(t)} = (\lambda_1^{(t)}, \dots, \lambda_k^{(t)})^\top \in \mathbb{R}^k$.

In principal, one can estimate $\{\lambda^{(t)}, \{A_i^{(t)}\}_{i=1}^{l}\}_{t=1}^{n}$ by decomposing each of the tensors $\mathcal{T}^{(t)}$ separately. However, such an approach will not take advantage of the structure in the underlying dynamics of the parameters. For example, a natural assumption in many applications is that the parameters $\lambda^{(t)}, A_i^{(t)}$ evolve smoothly over time. Here, we incorporate such a belief into our estimation of $\{\lambda^{(t)}, \{A_i^{(t)}\}_{i=1}^{l}\}_{t=1}^{n}$ by introducing a Gaussian process prior over the parameters.

We now develop a Bayesian probabilistic model to approach the described problem. First, each entry of the noise tensor $N^{(t)}$ is taken to be independent and identically distributed (i.i.d) with a normal distribution, $\mathcal{N}(0, \sigma_N^2)$.

Denote $a_{ij}^{(\cdot)} \in \mathbb{R}^{d_i \times n}$ the matrix with column t being $a_{ij}^{(t)}$ and similarly for $\lambda^{(\cdot)} \in \mathbb{R}^{k \times n}$. We assume that the prior over the parameters of the tensors $\mathcal{T}^{(t)}$ is given by

$$\begin{aligned} \lambda^{(\cdot)} &\sim \mathcal{MN}\left(y_\lambda, U_\lambda, V_\lambda\right) \quad U_\lambda \in \mathbb{R}^{k \times k}, \; V_\lambda \in \mathbb{R}^{n \times n} \\ a_{ij}^{(\cdot)} &\sim \mathcal{MN}\left(y_{ij}^{(\cdot)}, U_{a_{ij}}, V_{a_{ij}}\right) \quad j \in [k], \; U_{a_{ij}} \in \mathbb{R}^{d_i \times d_i}, \; V_{a_{ij}} \in \mathbb{R}^{n \times n} \end{aligned} \tag{39}$$

where $\mathcal{MN}(M, U, V)$ is the matrix normal distribution[3], which we analyse more thoroughly in Appendix B. This approach using Gaussian processes allows us to model the correlation between the elements on each random matrix $a_{ij}^{(\cdot)}$ using kernels or some other estimators for the covariance matrices $\{U_\lambda, V_\lambda, \{U_{a_{ij}}, V_{a_{ij}}\}_{i,j=1}^{l,k}\}$.

Note that our approach is powerful in the sense that it doesn't impose any restrictions on the dimension where the correlation is applied being discrete, unlike other methods like discrete Markov chains, which would provide simplier statistical models.

[3] The probability function for the random matrix X $(n \times p)$ that follows the matrix normal distribution $\mathcal{MN}_{n,p}(\mathbf{M}, \mathbf{U}, \mathbf{V})$ has the form:

$$p\left(\mathbf{X} \mid \mathbf{M}, \mathbf{U}, \mathbf{V}\right) = \frac{\exp\left(-\frac{1}{2}\operatorname{tr}\left[\mathbf{V}^{-1}(\mathbf{X}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{X}-\mathbf{M})\right]\right)}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{p/2}}$$

where tr denotes trace, $|\cdot|$ denotes the determinant and M is $n \times p$, U is $n \times n$ and V is $p \times p$.

Assuming that each entry in the noise tensor $N^{(t)} \in \mathbb{R}^{d_1 \times \cdots \times d_l}$ is normally distributed, $N^{(t)}_{i_1,\ldots,i_l} \sim \mathcal{N}(0, \sigma_N^2)$, we can express each element $\mathcal{T}^{(t)}_{i_1,\ldots,i_l}$ as a product of $l+1$ k-dimensional vectors:

$$\mathcal{T}^{(t)}_{i_1,\ldots,i_l} \approx < \lambda^{(t)}, a^{(t)}_{1,:,i_1}, \ldots, a^{(t)}_{l,:,i_l} >=: \sum_{j=1}^{k} \lambda^{(t)}_j a^{(t)}_{1,j,i_1} \cdots a^{(t)}_{l,j,i_l} \tag{40}$$

where $a^{(t)}_{i,:,j} = (a^{(t)}_{i,1,j}, \ldots, a^{(t)}_{i,k,j})^\top \in \mathbb{R}^k$, $a^{(t)}_{i,j,r}$ is the r^{th} element of the vector $a^{(t)}_{i,j} \in \mathbb{R}^{d_i}$ and $\approx$ is used to express the randomness added by $\mathcal{N}(0, \sigma_N^2)$. Note that we can think of each $a^{(\cdot)}_i$ as a tensor in $\mathbb{R}^{k \times d_i \times n}$ $\forall i \in [l]$, where we use a colon (:) on one dimension to denote that we keep all the elements on that mode.

Equivalently, we may consider

$$\mathcal{T}^{(t)}_{i_1,\ldots,i_l} | \lambda^{(t)}, \{a^{(t)}_{ij}\}^{l,k}_{i,j=1} \sim \mathcal{N}\left(< \lambda^{(t)}, a^{(t)}_{1,:,i_1}, \ldots, a^{(t)}_{l,:,i_l} >, \sigma_N^2 \right) \tag{41}$$

to express that we can recover a noisy version of the tensor $\mathcal{T}^{(t)}$ given $\{\lambda^{(t)}, \{a^{(t)}_{ij}\}^{l,k}_{i,j=1}\}$ or, what is the same, we can get an estimation of each element $\mathcal{T}^{(t)}_{i_1,\ldots,i_l}$ given $\{\lambda^{(t)}, \{a^{(t)}_{ij}\}^{l,k}_{i,j=1}\}$.

We now define the composite tensor of order $l+1$,

$$\mathcal{T} = \sum_{t=1}^{n} \mathcal{T}^{(t)} \otimes e_t \tag{42}$$

Our final goal is to recover the rank one components that estimate the tensors $\mathcal{T}^{(t)}$, $\forall t \in [n]$.

5.1 Statistical approach for recovering the tensor

We now aim to compute the posterior distribution of the above model, $p(\Theta|X)$, where Θ represents all the parameters of the model and X the observed evidence (the tensor). We first use Bayes rule to get $p(\Theta|X) = \frac{p(X|\Theta)p(\Theta)}{p(X)} \propto p(X|\Theta)p(\Theta) = Likelyhood \times Prior$, where $\propto$ denotes proportionality. Our goal is to maximize the posterior distribution and, as the logarithm is a continuous and monotonically increasing function, we can equivalently maximize $log(p(\Theta|X)) \propto log(Likelyhood) + log(Prior)$. Under the above model, the log posterior distribution of the data $\mathcal{T}$ is given by

$$
\begin{aligned}
log\ p\left(\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k} | \mathcal{T}\right) &\propto \underbrace{log\ p\left(\mathcal{T}|\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k}\right)}_{\text{(A1)}} + \underbrace{log\ p\left(\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k}\right)}_{\text{(A2)}} \\
= -\sum_{t=1}^{n} \sum_{i_1,\ldots,i_l=1}^{d_1,\ldots,d_l} & \frac{\left(\mathcal{T}_{i_1,\ldots,i_l}^{(t)} - < \lambda(t), a_{1,:,i_1}^{(t)}, \ldots, a_{l,:,i_l}^{(t)} >\right)^2}{2\sigma_N^2} - \frac{n(d_1 + \cdots + d_l)\log(\sigma_N^2)}{2} \\
- \frac{1}{2}\text{tr}\left[V_\lambda^{-1}\left(\lambda^{(\cdot)} - y_\lambda^{(\cdot)}\right)^\top U_\lambda^{-1}\left(\lambda^{(\cdot)} - y_\lambda^{(\cdot)}\right)\right] & - \frac{n}{2} log|U_\lambda| - \frac{k}{2} log|V_\lambda| \\
- \sum_{i=1}^{l}\sum_{j=1}^{k}\left(\frac{1}{2}\text{tr}\left[V_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)} - y_{ij}^{(\cdot)}\right)^\top U_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)} - y_{ij}^{(\cdot)}\right)\right] - \frac{n}{2} log|U_{a_{ij}}| - \frac{d_i}{2} log|V_{a_{ij}}|\right) & \\
+ const &
\end{aligned}
\tag{43}
$$

where

$$
\begin{aligned}
(A1) &= log\ p\left(\mathcal{T}|\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k}\right) = log\ p\left(\sum_{t=1}^{n} \mathcal{T}^{(t)} \otimes e_t | \lambda^{(t)}, \{a_{ij}^{(t)}\}_{i,j=1}^{l,k}\right) \\
&= log\left(\prod_{t=1}^{n}\prod_{i_1,\ldots,i_l=1}^{d_1,\ldots,d_l} p\left(\mathcal{T}_{i_1,\ldots,i_l}^{(t)} | \lambda^{(t)}, \{a_{ij}^{(t)}\}_{i,j=1}^{l,k}\right)\right) = \sum_{t=1}^{n}\sum_{i_1,\ldots,i_l=1}^{d_1,\ldots,d_l} log\ p\left(\mathcal{T}_{i_1,\ldots,i_l}^{(t)} | \lambda^{(t)}, \{a_{ij}^{(t)}\}_{i,j=1}^{l,k}\right) \\
&= -\sum_{t=1}^{n} \sum_{i_1,\ldots,i_l=1}^{d_1,\ldots,d_l} \frac{\left(\mathcal{T}_{i_1,\ldots,i_l}^{(t)} - < \lambda^{(t)}, a_{1,:,i_1}^{(t)}, \ldots, a_{l,:,i_l}^{(t)} >\right)^2}{2\sigma_N^2} - \frac{n(d_1 + \cdots + d_l)\log(\sigma_N^2)}{2} + const_1 \\
(A2) &= log\ p\left(\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k}\}\right) = log\ p\left(\lambda^{(\cdot)}\right) + \sum_{i=1}^{l}\sum_{j=1}^{k} log\ p\left(a_{ij}^{(\cdot)}\right) \\
&= -\frac{1}{2}\text{tr}\left[V_\lambda^{-1}\left(\lambda^{(\cdot)} - y_\lambda^{(\cdot)}\right)^\top U_\lambda^{-1}\left(\lambda^{(\cdot)} - y_\lambda^{(\cdot)}\right)\right] - \frac{n}{2} log|U_\lambda| - \frac{k}{2} log|V_\lambda| \\
&\quad - \sum_{i=1}^{l}\sum_{j=1}^{k}\left(\frac{1}{2}\text{tr}\left[V_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)} - y_{ij}^{(\cdot)}\right)^\top U_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)} - y_{ij}^{(\cdot)}\right)\right] + \frac{n}{2} log|U_{a_{ij}}| + \frac{d_i}{2} log|V_{a_{ij}}|\right) \\
&\quad + const_2
\end{aligned}
$$

and $\lambda^{(\cdot)}$, $a_{ij}^{(\cdot)}$ are independent for all $i \neq j$.

Note that we have strongly used the normality described in Equation (41) and in the Gaussian priors defined in Equation (39). Now, under a fixed value of σ_N^2, maximizing the log-posterior with respect to $\{\lambda^{(\cdot)}, \{a_{ij}^{(\cdot)}\}_{i,j=1}^{l,k}\}$ is equivalent to minimizing the following sum:

$$\sum_{t=1}^{n}\sum_{i_1,\ldots,i_l=1}^{d_1,\ldots,d_l}\frac{\left(\mathcal{T}^{(t)}_{i_1,\ldots,i_l}- < \lambda^{(t)}, a^{(t)}_{1,:,i_1},\ldots,a^{(t)}_{l,:,i_l} >\right)^2}{\sigma_N^2}$$
$$+\operatorname{tr}\left[V_\lambda^{-1}\left(\lambda^{(\cdot)}-y_\lambda^{(\cdot)}\right)^\top U_\lambda^{-1}\left(\lambda^{(\cdot)}-y_\lambda^{(\cdot)}\right)\right]+n\,log|U_\lambda|+k\,log|V_\lambda|$$
$$+\sum_{i=1}^{l}\sum_{j=1}^{k}\left(\operatorname{tr}\left[V_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)}-y_{ij}^{(\cdot)}\right)^\top U_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)}-y_{ij}^{(\cdot)}\right)\right]+n\,log|U_{a_{ij}}|+d_i\,log|V_{a_{ij}}|\right) \tag{44}$$

or, equivalently, the following expression

$$\rho\left\|\mathcal{T}-\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\lambda_j^{(t)}a_{1,j,:}^{(t)}\otimes\cdots\otimes a_{l,j,:}^{(t)}\right)\otimes\boldsymbol{e}_t\right\|_F^2+\operatorname{tr}\left[V_\lambda^{-1}\left(\lambda^{(\cdot)}-y_\lambda^{(\cdot)}\right)^\top U_\lambda^{-1}\left(\lambda^{(\cdot)}-y_\lambda^{(\cdot)}\right)\right]$$
$$+\sum_{i=1}^{l}\sum_{j=1}^{k}\left(\operatorname{tr}\left[V_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)}-y_{ij}^{(\cdot)}\right)^\top U_{a_{ij}}^{-1}\left(a_{ij}^{(\cdot)}-y_{ij}^{(\cdot)}\right)\right]+n\,log|U_{a_{ij}}|+d_i\,log|V_{a_{ij}}|\right)$$
$$+n\,log|U_\lambda|+k\,log|V_\lambda| \tag{45}$$

where $\rho=\frac{1}{\sigma_N^2}$ and $\|\cdot\|_F$ represents the Frobenius norm[4].

Corollari 5.1. *The optimization problem obtained in Definition 4.1 is a particular case of the above model. Specifically, it is the case for $n=1$, where the recoverd tensor is symmetric and the priors are deterministic instead of following a random process. A regularization term has also been appended.*

5.2 Discussion of the model

We have described a Bayesian approach for recovering a tensor with some correlations given by a Gaussian process. What we have described as a correlated

[4]Given $A\in\mathbb{R}^{m\times n}$ the Frobenius (or Hilbert-Schmidt) norm can be defined in various ways:

$$\|A\|_F:=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}=\sqrt{\operatorname{trace}(A^*A)}=\sqrt{\sum_{i=1}^{\min\{m,\,n\}}\sigma_i^2}$$

where A^* denotes the conjugate transpose of A, σ_i are the singular values of A, and the trace function is used.

In the case of $\mathcal{T}\in\mathbb{R}^{d_1\times\cdots\times d_r}$, the definition of the Frobenius norm can be easily extended to

$$\|\mathcal{T}\|_F:=\sqrt{\sum_{i_1=1}^{d_1}\cdots\sum_{i_r=1}^{d_r}|\mathcal{T}_{i_1,\ldots,i_r}|^2}$$

component can be understood as a temporal component but it can also be used to represent, for example, different clusters of nodes in a graph.

It is important to remark that we have imposed the correlation to rely on all the tensors $\mathcal{T}^{(t)}$ and not only on a specific subset of them, an assumption that is not normally used for time, where only past events are used to model the future. However, we believe this simplification might still work well in many applications and it can easily be extended to model time in a more precise way.

We can also observe that while our approach may suit a large number of methods, it presents both computational and storage limitations. Storing a high dimensional tensor can present a memory challange, but also making any kind of calculations with it can be computationally expensive.

Note that another difficulty may lay in calculating the inverse of the kernels V_λ, U_λ, $V_{a_{ij}}$ and $U_{a_{ij}}$, as calculating the inverse of a matrix is, in general, a computationally expensive problem (cubical complexity) which can be unpractical when dealing with high dimensionalities. On the other hand, it is relevant to mention that in most of the applications it would suffice to calculate the inverse of the kernels only once.

5.3 Applications

5.3.1 Temporal topic modeling

We have seen in Section 4 how tensor methods can be used to learn Latent Dirichlet Allocation (LDA). We now face the problem of learning topic models along a time component and we aim to provide a new algorithm for this problem using the statistical framework derived in Section 5.1.

Much research has been done on how to learn topic models over time. Most of the methods make variations of the sucessful Latent Dirichlet Allocation model and adapt it to add a time component. Some methods such as the ones presented in (Blei and Lafferty, 2006) and (Wang et al., 2012) assume some of the hyper parameters of a similar to LDA model follow a Gaussian process. Other relevant references are (Wang and McCallum, 2006), (Hong et al., 2011) and (Dubey et al., 2013).

We assume that we have a corpus with a temporal component and we want to find the most significant topics at each time (for example, newspaper articles for which we want to get the most significant topics for each year). We may just learn independent LDA models at each time, but this approach wouldn't take advantage of the existing data from other years. Indeed, it makes sense to assume that the information provided by close-by years may help find more significant topics.

Temporal LDA using a Gaussian prior

We start by making the assumption that topic models recovered using LDA which belong to close-by times will have correlated third order moments, reason why we design an algorithm for learning topic models by assuming that the whitened third order moment at a time t, the tensor $\mathcal{T}^{(t)}$, has a Gaussian correlation with the whitened third order moment at another time $m \neq t$, $\mathcal{T}^{(m)}$.

In particular, we approach the problem by focusing on a specific case of the model described in Section 5.1. Given data from n different times, we aim to learn a reduced and whitened version $\mathcal{T} = \sum_{t=1}^{n} \mathcal{T}^{(t)} \otimes e_n$ of the composite third order moment tensor of the LDA model, $\widehat{\mathcal{M}} = \sum_{t=1}^{n} \widehat{\mathcal{M}_3}^{(t)} \otimes e_t$, where $\widehat{\mathcal{M}_3}^{(t)}$ is the empirical third order moment provided in Lemma 4.1 and is calculated using only documents that belong to time t.

We will base our approach on the hypothesis that the eigenvectors of each $\mathcal{T}^{(t)}$ follow a Gaussian process:

$$\mathcal{T}^{(t)} = \sum_{j=1}^{k} v_j^{(t)} \otimes v_j^{(t)} \otimes v_j^{(t)} + N^{(t)}, \tag{46}$$

where

$$v_j^{(\cdot)} \sim \mathcal{MN}\left(y_j^{(\cdot)}, Id, K\right) \quad j \in [k],\ K \in \mathbb{R}^{n \times n} \tag{47}$$

and the t^{th} column of $v_j^{(\cdot)}$, $v_j^{(t)} \in \mathbb{R}^k$, forms one of the k rank-one components (or eigenvectors) that estimate the whitened third order moment at time t, $\mathcal{T}^{(t)} = \sum_{j=1}^{k} v_j^{(t)} \otimes v_j^{(t)} \otimes v_j^{(t)} + N^{(t)}$, $y_j^{(\cdot)}$ is the expected value of $v_j^{(\cdot)}$ and $N^{(t)}$ is some Gaussian noise.

The matrix $K \in \mathbb{R}^{k \times k}$ can be calculated as a kernel matrix that depends on the time component. Many kernel functions have been derived and finding the most appropriate one for each problem is, in general, a difficult problem. While many of them may be suitable for approaching time series, we suggest a Radial Basis Function Kernel (RFBF kernel) as a starting point. For more information about kernels see (Rasmussen and Williams, 2005).

The RBF kernel on two samples x and x', represented as feature vectors in some input space, is defined as $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma_K^2}\right)$ where $||\mathbf{x} - \mathbf{x}'||^2$ may be recognized as the squared Euclidean distance between the two feature vectors and σ_K is a free parameter. In our case, we first define the number of times we want to learn, n. Then, using constant vectors $w_i = [i \ \dots \ i]^\top \in \mathbb{R}^n$, we define our kernel matrix $K \in \mathbb{R}^{n \times n}$ as the matrix which element in the position (i, j) is $K_{ij} = \exp\left(-\frac{||w_i - w_j'||^2}{2\sigma_K^2}\right)$.

Note that this kernel gives more correlation to LDA models that are close in time, and this correlation decreases exponentially as the time difference increases. We can use this model to model the correlation between the data that belong to different times.

The model

Our new approach could be understood as trying to reproduce a multiple LDA model where the Dirichlet priors are correlated between them. A valid graphical model for this approach, assuming only two times were given, t_1 and t_2, could be the one given in Figure 4.

Unfortunately, we can't derive a statistical model for recovering the parameters of n LDA models using tensor methods and a Gaussian prior for the Dirichlet priors due to the whitening step to reduce dimensionality.

A true statistical model for recovering the composite whitened tensor $\mathcal{T}$ could be designed by imposing that the Gaussian prior is on the word-topic matrix instead that on the eigenvectors of the whitened tensors $\mathcal{T}^{(t)}$.

In this case, looking at Section 4.6 we can see that instead of adding a Gaussian normalization to our optimization function (from the assumption that the eigenvectors of the whitened tensors follow a Gaussian process) we would have to add a normalization for a different distribution.

More precisely, from Equation (32) we obtain that ${W^{(t)}}^\top \beta^{(t)} = V^{(t)}$, where $W^{(t)}$ is the whitening matrix at time t and $V^{(t)} \in \mathbb{R}^{k\times k}$ is the matrix whose i^{th} column is the i^{th} eigenvector of the whitened third order moment $\mathcal{T}^{(t)}$. This means we would have to calculate the distribution of the random matrices $V_i^{(\cdot)} := v_i^{(\cdot)} = \left(W_i^{(\cdot)}\right)^\top \beta_i^{(\cdot)}$. Even if we assume that $\beta_i^{(\cdot)}$ is normally distributed, $\left(W_i^{(\cdot)}\right)^\top \beta_i^{(\cdot)}$ is the product of two non-independent random variables that seems unapproachable.

We shall remember that $W^{(t)} = U^{(t)}_{M_2,k}\Sigma^{(t)}_{M_2,k}$, with $U^{(t)}_{M_2,k}$ and $\Sigma^{(t)}_{M_2,k}$ the matrix whose columns are the first k singular vectors and singular values of the second order moment $M_2^{(t)}$, respectively. Hence, as $\beta_i^{(t)}$ are the eigenvectors of the third order order moment $\mathcal{M}_3^{(t)}$ (recall Theorem 4.1), $\left(W_i^{(\cdot)}\right)^\top$ and $\beta_i^{(t)}$ are not independent.

Learning topic models over time

We now use Equation (45) to define our optimization problem to learn topic models over time.

Definition 5.1. Our optimization problem is given by

$$\begin{aligned}\underset{v_j^{(\cdot)},\, j\in 1:[k]}{\operatorname{argmin}}\ \rho &\left\| \sum_{t=1}^{n} \left(\frac{1}{n_X} \sum_{x\in X} \mathcal{T}_x^{(t)} - \sum_{j=1}^{k} \otimes^3 v_j^{(t)} \right) \otimes e_t \right\|_F^2 \\ &+ 3\sum_{j=1}^{k} \operatorname{tr}\left[K^{-1} \left(v_j^{(\cdot)} - y_j^{(\cdot)} \right)^\top \left(v_j^{(\cdot)} - y_j^{(\cdot)} \right) \right] + \theta \sum_{t=1}^{n} \left\| \sum_{j=1}^{k} \otimes^3 v_i^{(t)} \right\|_F^2 \end{aligned} \tag{48}$$

where $v_i^{(\cdot)}$ are the unknown components to be estimated, and $\rho, \theta > 0$ are some fixed parameters. As for the matrices $y_j^{(\cdot)}$, they are theoretically the expectation of

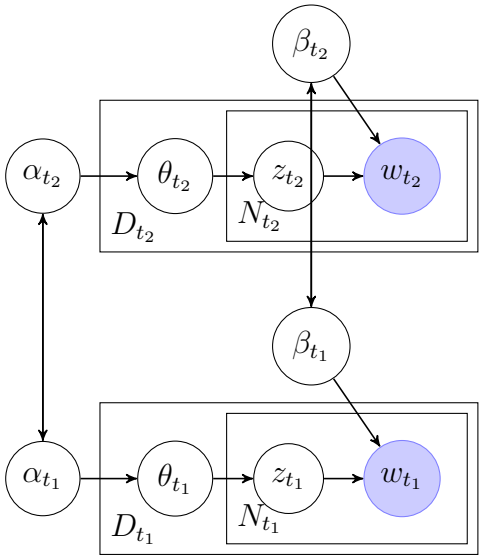


Figure 4: Graphical model representation of a multitime LDA with correlated Dirichlet priors for only two times t_1 and t_2.

the matrices $v_j^{(\cdot)}$, reason why we will solve the problem by using a two step iterative algorithm where $y_j^{(t)}$ will always be approximated by the mean of the estimated neighbooring points of $v_j^{(t)}$ at the previous iteration, $\frac{v_j^{(t-1)}+v_j^{(t+1)}}{2}$ if $t \neq 1, n$, and $v_j^{(2)}$ or $v_j^{(n-1)}$ if $t = 1$ or $t = n$, respectively. The algorithm will first calculate $\{v_j^{(\cdot)}\}_{j=1}^k$ by reconstructing each of the tensors without any Gaussian correlation, and will follow as described above.

In order to encourage orthogonality between the eigenvectors of the three-order tensors on each time $t \in [n]$ and similarly to Definition 4.1, we add the extra term $\theta \sum_{t=1}^n \left\| \sum_{j=1}^k \otimes^3 v_i^{(t)} \right\|_F^2$, where the parameter $\theta > 0$ is a weight.

Define $V^{(\cdot)} := \sum_{t=1}^n V^{(t)} \otimes e_i$, where $V^{(t)} = \left[v_1^{(t)} | \dots | v_k^{(t)} \right]$.

Lemma 5.1. *The optimization problem given in Definition 5.1 is equivalent to minimizing a loss function* $L(V^{(\cdot)}) := \frac{1}{n_x} \sum_{x \in X} L_x(V^{(\cdot)})$, *where* $L_x(V^{(\cdot)})$ *is the loss function evaluated at the document* x, *and is given by*

$$\begin{aligned} L_x(V^{(\cdot)}) :=& (\rho + \theta) \left\| \sum_{t=1}^n \left(\sum_{j=1}^k \otimes^3 v_j^{(t)} \right) \otimes e_t \right\|_F^2 - 2\rho \sum_{t=1}^n \left\langle \sum_{j=1}^k \otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)} \right\rangle \\ &+ 3 \sum_{j=1}^k \operatorname{tr} \left[K^{-1} \left(v_j^{(\cdot)} - y_j^{(\cdot)} \right)^\top \left(v_j^{(\cdot)} - y_j^{(\cdot)} \right) \right] \end{aligned} \tag{49}$$

Proof. We have

$$\begin{aligned}
&\rho\left\|\sum_{t=1}^{n}\left(\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x^{(t)}-\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\|_F^2+\theta\sum_{t=1}^{n}\left\|\frac{1}{n_x}\sum_{j=1}^{k}\otimes^3 v_i^{(t)}\right\|_F^2\\
&=\rho\left\|\sum_{t=1}^{n}\left(\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x^{(t)}\right)\otimes e_t-\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\|_F^2+\theta\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_i^{(t)}\right)\otimes e_t\right\|_F^2\\
&=\rho\left\|\sum_{t=1}^{n}\left(\frac{1}{n_x}\sum_{x\in X}\mathcal{T}_x^{(t)}\right)\otimes e_t\right\|_F^2-2\rho\left\langle\sum_{t=1}^{n}\left(\sum_{x\in X}\mathcal{T}_x^{(t)}\right)\otimes e_t,\ \sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\rangle\\
&\quad+\rho\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\|_F^2+\theta\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_i^{(t)}\right)\otimes e_t\right\|_F^2\\
&=(\rho+\theta)\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_i^{(t)}\right)\otimes e_t\right\|_F^2-2\rho\sum_{t=1}^{n}\left\langle\sum_{j=1}^{k}\otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)}\right\rangle+\rho\left\|\sum_{t=1}^{n}\left(\sum_{x\in X}\mathcal{T}_x^{(t)}\right)\otimes e_t\right\|_F^2
\end{aligned}\tag{50}$$

Now, since $\rho\left\|\sum_{t=1}^{n}\left(\sum_{x\in X}\mathcal{T}_x^{(t)}\right)\otimes e_t\right\|_F^2$ is a constant, the optimization problem in *Definition* 5.1 becomes

$$\begin{aligned}
&(\rho+\theta)\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\|_F^2-2\rho\sum_{t=1}^{n}\left\langle\sum_{j=1}^{k}\otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)}\right\rangle\\
&+3\sum_{j=1}^{k}\operatorname{tr}\left[\left(v_j^{(\cdot)}-y_j^{(\cdot)}\right)^\top U^{-1}\left(v_j^{(\cdot)}-y_j^{(\cdot)}\right)\right]\\
&=\frac{1}{n_x}\sum_{x\in X}\left((\rho+\theta)\left\|\sum_{t=1}^{n}\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\otimes e_t\right\|_F^2-2\rho\sum_{t=1}^{n}\left\langle\sum_{j=1}^{k}\otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)}\right\rangle\right.\\
&\left.+3\sum_{j=1}^{k}\operatorname{tr}\left[\left(v_j^{(\cdot)}-y_j^{(\cdot)}\right)^\top U^{-1}\left(v_j^{(\cdot)}-y_j^{(\cdot)}\right)\right]\right)=\frac{1}{n_x}\sum_{x\in X}L_x(V^{(\cdot)})
\end{aligned}\tag{51}$$

□

Lemma 5.2. *The partial derivatives of $L_x(V^{(\cdot)})$ are:*

$$\begin{aligned}
\frac{\partial L_x(V^{(\cdot)})}{\partial v_j^{(\cdot)}}=\sum_{t=1}^{m}6\left((1+\theta)\sum_{p=1}^{k}\langle v_p,v_j\rangle^2 v_p-\sum_{i=1}^{k}\left(v_j^\top\mathcal{T}_{x,(:,:,i)}v_j\right)\otimes e_i\right)\otimes e_t\\
+v_j^{(\cdot)}\left(K^{-1}\right)^\top+v_j^{(\cdot)}K^{-1}
\end{aligned}\tag{52}$$

Proof. We first calculate the derivative of the first two terms of the sum in Equation (49). As $\forall i,j\quad v_i^{(m)}$ is a constant with respect to $v_j^{(t)}\ \forall m\neq t$, it suffices to calculate the derivative of $(\rho+\theta)\left\|\left(\sum_{j=1}^{k}\otimes^3 v_j^{(t)}\right)\right\|_F^2-2\rho\left\langle\sum_{j=1}^{k}\otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)}\right\rangle$ with

respect to $v^{(t)}$ for each $t \in [n]$, which we have already calculated in Lemma 4.3. The new derivatives are:

$$\begin{aligned}
&\frac{\partial}{\partial v_j^{(\cdot)}} \left((\rho+\theta) \left\| \sum_{t=1}^{n} \left(\sum_{j=1}^{k} \otimes^3 v_j^{(t)} \right) \otimes e_t \right\|_F^2 - 2\rho \sum_{t=1}^{n} \left\langle \sum_{j=1}^{k} \otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)} \right\rangle \right) \\
&= \sum_{t=1}^{n} \frac{\partial}{\partial v_j^{(t)}} \left((\rho+\theta) \left\| \left(\sum_{j=1}^{k} \otimes^3 v_j^{(t)} \right) \right\|_F^2 - 2\rho \left\langle \sum_{j=1}^{k} \otimes^3 v_j^{(t)},\ \mathcal{T}_x^{(t)} \right\rangle \right) \otimes e_t \\
&= \sum_{t=1}^{m} 6 \left((1+\theta) \sum_{p=1}^{k} \langle v_p, v_j \rangle^2 v_p - \sum_{i=1}^{k} \left(v_j^\top \mathcal{T}_{x,(:,:,i)} v_j \right) \otimes e_i \right) \otimes e_t
\end{aligned} \tag{53}$$

We can calculate the derivative of the last term in Equation (49) by reducing the problem to one of one dimensional calculus in the following way:

$$\begin{aligned}
&\frac{\partial}{\partial v_i^{(\cdot)}} \left(3 \sum_{j=1}^{k} \operatorname{tr} \left[K^{-1} \left(v_j^{(\cdot)} - y_j^{(\cdot)} \right)^\top v_j^{(\cdot)} - y_j^{(\cdot)} \right] \right) = \\
&\qquad 3 \frac{\partial}{\partial v_i^{(\cdot)}} \left(\operatorname{tr} \left[\left(v_i^{(\cdot)} - y_i^{(\cdot)} \right) K^{-1} \left(v_i^{(\cdot)} - y_i^{(\cdot)} \right)^\top \right] \right)
\end{aligned} \tag{54}$$

where

$$\begin{aligned}
&\operatorname{tr} \left[\left(v_i^{(\cdot)} - y_i^{(\cdot)} \right) K^{-1} \left(v_i^{(\cdot)} - y_i^{(\cdot)} \right)^\top \right] = \sum_{j=1}^{k} \left[\left(v_i^{(\cdot)} - y_i^{(\cdot)} \right) K^{-1} \left(v_i^{(\cdot)} - y_i^{(\cdot)} \right)^\top \right]_{jj} \\
&= \sum_{j=1}^{k} \sum_{m=1}^{n} \left(v_{ij}^{(m)} - y_{ij}^{(m)} \right) \left(K^{-1} \left(v_i^{(\cdot)} - y_i^{(\cdot)} \right)^\top \right)_{mj} \\
&= \sum_{j=1}^{k} \sum_{m=1}^{n} \left(v_{ij}^{(m)} - y_{ij}^{(m)} \right) \sum_{r=1}^{n} K_{mr}^{-1} \left(v_{ij}^{(r)} - y_{ij}^{(r)} \right)^\top \\
&= \sum_{j=1}^{k} \sum_{m=1}^{n} \sum_{r=1}^{n} \left(v_{ij}^{(m)} - y_{ij}^{(m)} \right) K_{mr}^{-1} \left(v_{ij}^{(r)} - y_{ij}^{(r)} \right)^\top
\end{aligned} \tag{55}$$

However, in this case the closed form $\frac{\partial}{\partial X} \operatorname{tr} \left(X A X^\top \right) = X A^\top + X A$ from (Petersen and Pedersen, 2012) allows us to find the same result in a faster way:

$$\frac{\partial}{\partial v_i^{(\cdot)}} \left(3 \sum_{j=1}^{k} \operatorname{tr} \left[K^{-1} \left(v_j^{(\cdot)} \right)^\top v_j^{(\cdot)} \right] \right) = v_i^{(\cdot)} \left(K^{-1} \right)^\top + v_i^{(\cdot)} K^{-1} \tag{56}$$

Putting together the two terms, we finally obtain:

$$\frac{\partial L_x(V^{(\cdot)})}{\partial v_j^{(\cdot)}} = \sum_{t=1}^{m} 6 \left((1+\theta) \sum_{p=1}^{k} \langle v_p, v_j \rangle^2 v_p - \sum_{i=1}^{k} \left(v_j^\top \mathcal{T}_{x,(:,:,i)} v_j \right) \otimes e_i \right) \otimes e_t + v_j^{(\cdot)} \left(K^{-1} \right)^\top + v_j^{(\cdot)} K^{-1} \tag{57}$$

□

Last, using the derivative of the optimization function we get the iterative update for the stochastic gradient descent

$$\tilde{v}_{i,(n+1)}^{(\cdot)} \leftarrow \tilde{v}_{i,(n)} - \alpha_n \frac{\partial L_x(V^{(\cdot)})}{\partial v_i^{(\cdot)}} \Big|_{\tilde{V}_{(n)}^{(\cdot)}}, \ \forall i \in [k] \tag{58}$$

where α_n is the learning rate at the n^{th} iteration, the derivatives are calculated for each document x and $\tilde{v}_{i,(n)}^{(\cdot)}$ contains the learnt eigenvectors i at iteration n.

The final method is summarized in Algorithm 1.

5.3.2 Temporal collaborative filtering

In Section 3.4 we have reviewed how LDA can be used for collaborative filtering. Now we aim to analyse how our previous model would work for temporal collaborative filtering.

Typical collaborative filtering methods can be categorized in two classes: neighborhood methods and factorization methods. While, in general, factor-based algorithms are considered more effective than those based on neighborhoods, these two classes are often complementary and the best results might be obtained by blending them.

Assume we aim to solve the movie recommendation problem (in this section we will talk about movie recommendations, but we could equivalently be talking about general items instead). Given a user, and their rating for some movies, we aim to approximate what rating would give the user to other movies that he or she has not rated yet but that have already been rated by other users. For each user a_i and each movie b_j we observe a rating r_{ij}. Each instance of the data is a tuple (a_i, b_j, r_{ij}). Assuming there are n users and m movies, we can organize this tuples into a sparse matrix $R \in \mathbb{R}^{n \times m}$ by using a_i and b_j as the indexes of R: $R_{a_i,b_j} = r_{ij}$. From now one we will denote R_{a_i,b_j} as R_{ij}.

Probability Matrix Factorization

One of the representative factor-based methods is Probabilistic Matrix Factorization (PMF) (Salakhutdinov and Mnih, 2011), which assigns a d-dimensional latent feature vector for each user and movie, $u_i \in \mathbb{R}^d$ and $v_j \in \mathbb{R}^d$, respectively, and each rating as the inner product $R_{ij} = u_i^\top v_j$. As d is chosen to be much lower than the

Algorithm 1 Overall approach for learning temporal LDA with k topics.

Input: Observed data: document samples on each timestamp $t \in \{1, \ldots, n\}$

Output: Learned latent variable model (word-topic matrix) and Dirichlet hyperparameters, for each time t

for t in $\{1, \ldots, n\}$ **do**

  Compute the empirical second order moment $\widehat{\mathcal{M}_2^{(t)}}$

  Use $\widehat{\mathcal{M}_2^{(t)}}$ to calculate the whitened empirical third order tensor $\mathcal{T}^{(t)}$

  Compute an orthogonal CP decomposition of rank k of $\mathcal{T}^{(t)}$, $\mathcal{T}^{(t)} \approx \sum_{i=1}^{k} \tilde{v}_i^{(t)} \otimes \tilde{v}_i^{(t)} \otimes \tilde{v}_i^{(t)}$

end for

Initialize $j \leftarrow 0$

Comment: In general, define $\tilde{V}_{(j)}^{(\cdot)}$ as the tensor that approximates $\sum_{i=1}^{k} V^{(t)} \otimes e_t$ at iteration j. Let ϵ be some fixed parameter and α_j the learning rate at iteration j

Initialize $\tilde{V}_{(0)}^{(\cdot)} = \sum_{i=1}^{k} \tilde{v}_i^{(\cdot)} \otimes e_i$,

$Y_{(0)}^{(\cdot)} = \sum_{i=1}^{k} \left[\tilde{v}_i^{(2)} \quad \frac{\tilde{v}_i^{(1)} + \tilde{v}_i^{(3)}}{2} \quad \ldots \quad \frac{\tilde{v}_i^{(m-1)} + \tilde{v}_i^{(m+1)}}{2} \quad \ldots \quad \tilde{v}_i^{(t-1)} \right] \otimes e_i$

while $\left\| \tilde{V}_{(j)}^{(\cdot)} - \tilde{V}_{(j-1)}^{(\cdot)} \right\|_F > \epsilon$ **or** first iteration **do**

  Calculate the derivatives $G_{(j)}$ of the loss function $L(\tilde{V}_{(j)}^{(\cdot)})$ where $Y_{(j)}^{(\cdot)}$ is used as the mean of the Gaussian prior

  Update $\tilde{V}_{(j+1)}^{(\cdot)} \leftarrow \tilde{V}_{(j)}^{(\cdot)} - \alpha_j G_{(j)}$

  Update

  $Y_{(j+1)}^{(\cdot)} = \sum_{i=1}^{k} \left[\tilde{v}_{i,(j)}^{(2)} \quad \frac{\tilde{v}_{i,(j)}^{(1)} + \tilde{v}_{i,(j)}^{(3)}}{2} \quad \ldots \quad \frac{\tilde{v}_{i,(j)}^{(m-1)} + \tilde{v}_{i,(j)}^{(m+1)}}{2} \quad \ldots \quad \tilde{v}_{i,(j)}^{(t-1)} \right] \otimes e_i,$

  (each $\tilde{v}_{i,(j)}^{(l)}$ is obtained from $\tilde{V}_{(j)}^{(\cdot)}$)

  Update $j \leftarrow j + 1$

end while

for t in $\{1, \ldots, n\}$ **do**

  Post-processing: Use $\tilde{V}_{(j+1)}^{(\cdot)}$ to calculate the word-topic matrix $\beta^{(t)}$ and recover the Dirichlet hyperparameters

end for

number of movies in the dataset, this method consists of performing a low rank approximation of the ratings matrix R. It is also equivalent to performing Principal Component Analysis or SVD decomposition of the matrix R.

Formally, the model is defined by the following conditional distribution:

$$p(R|U,V,\alpha) = \prod_{i=1}^{n}\prod_{j=1}^{m}\left[\mathcal{N}\left(R_{ij}|u_i^\top v_j, \alpha^{-1}\right)\right]^{I_{ij}} \tag{59}$$

where $U = [u_1|\dots|u_n] \in \mathbb{R}^{d\times n}$ and $V = [v_1|\dots|v_m] \in \mathbb{R}^{d\times m}$, $\mathcal{N}$ denotes the Gaussian distribution, α is the observation precision and I_{ij} is the indicator that R_{ij} has been observed.

Probability Tensor Factorization

In (Xiong et al., 2010) the idea of PMF is extended to model a time evolving behavior. Unlike PMF, each entry R_{ijk} of a tensor R is refered with the tuple (i,j,k), where each of the three dimensions are used to represent users, movies and time, respectively. Each entry of the third order tensor R is modeled as:

$$R_{ijk} \approx \langle u_i, v_j, w_k\rangle := \sum_{m=1}^{d} u_{im}v_{jm}w_{km} \tag{60}$$

where u_i and v_j follow the previous notation and w_k is the k^{th} column of a matrix $W \in \mathbb{R}^{d\times l}$. Denote $U = [u_1|\dots|u_n] \in \mathbb{R}^{d\times n}$ and $V = [v_1|\dots|v_m] \in \mathbb{R}^{d\times m}$. Equivalently, it follows that

$$R = \sum_{m=1}^{d} \bar{u_i} \otimes \bar{v_i} \otimes \bar{w_i} \tag{61}$$

where $\bar{u_i}, \bar{v_i}$ and $\bar{w_i}$ represent the i^{th} row of the matrices U, V and W, respectively.

If we assume W determines a time component, then each rating does not only depend on how similar a user's preferences and an item's features are, like in PMF, but also on how much they match with the current temporal trends.

In this case, the randomness for each rating is modeled as

$$R_{ijk}|U,V,W \sim \mathcal{N}\left(\langle u_i, v_j, w_k\rangle, \alpha^{-1}\right) \tag{62}$$

with the assumption of the following prior distributions:

$$\begin{aligned} u_i &\sim \mathcal{N}(0, \sigma_U^2 Id) \quad i = [n] \\ v_j &\sim \mathcal{N}(0, \sigma_V^2 Id) \quad j = [m] \\ w_k &\sim \mathcal{N}(w_{k-1}, \sigma_W^2 Id) \quad k = [l] \\ w_0 &\sim \mathcal{N}(\mu_W, \sigma_0^2 Id) \end{aligned} \tag{63}$$

where σ_0, σ_U, σ_V, σ_W are some fixed scalars, $\mu \in \mathbb{R}^d$ is given and Id determines the identity matrix of size $d \times d$.

Finally, the model is recovered by maximizing the log-posterior distribution:

$$log\ p(U, V, W, w_0|R) \propto log\ p(R|U, V, W, w_0) + log\ p(U, V, W, w_0) \tag{64}$$

This model is named Probability Tensor Factorization (PTF) and it can be seen as a slightly modified case of our previously described tensor decomposition with correlated parameters. Unlike for topic modeling, PTF will need to learn many non-symmetrical second order tensors (matrices) assuming there is a Gaussian correlation between them. It is important to note that in this case the correlation is described by a Brownian motion process, reason why the Gaussian correlation will only be determined by the previous times. We could have also followed this approach in the previous sections, as it might adjust better to a temporal approach, but the overall idea of the previously described algorithm will not change and it can easily be adapted.

Another difference between PTF and our framework is the random prior of each rank-one component contained in U and V. While this seems to add a significant difference to the model, it will only add some extra regularization terms to the log-posterior distribution. The only complication arises due to the difficulty of estimating all the parameters in Equation (63), a problem that instead of being approached using (stochastic) gradient descent is solved with a fully Bayesian approach which uses Gibbs Sampling and a Markov Chain Monte Carlo method.

6 A dataset for temporal topic modeling

All kind of datasets are available online and it is not normally difficult to find one for any kind of applications. In the case of topic modeling, it's easy to find an already preprocessed corpus or even a collection of documents in the bag of words format.

Unfortunately, this is not the case for temporal topic modeling. Even though much research has been done on temporal topic modeling, most of the datasets have been directly adapted from the original corpus and kept private, what leads to non-comparable results and a large amount of time spent arrenging and formatting the datasets.

Typical datasets for temporal topic modeling include papers of some specific conferences or magazines (for example, NIPS papers) or its abstracts, personal emails of some of the authors of a paper, tweets obtained using the Twitter API and the transcripts of the State of the Union addresses. We will work with this last dataset.

A transcript of all the State of the Union addresses from 1790 to 2006 can be obtained from the project Gutenberg website [5]. This dataset is provided as a text file and the addresses are separated by three stars "***". In order to use this dataset for LDA, we are interested in formatting it as a bag of words, so that the whole corpus is represented using rows with the format "$x \quad y \quad z$" where x is an index that identifies a document, y is an index that identifies a word of the vocabulary and z is the number of times that word y appears in document x. Only pairs of x, y and z where z is not zero are stored.

6.1 Processing the dataset

We use Natural Language Toolkit (NLTK) [6], a powerful Python library for natural language processing. The first step for processing the text file consists of tokenizing the words, so that we obtain an array of words that we can work with. This allows us to filter some unuseful words, characters or numbers that lack any meaning using regular expressions. After, we split the addresses using the "***" characters as the divider between them, downcase all the text and remove stopwords, which are words that are very used in english and that lack any meaning (articles, prepositions...).

Next, following the work in (Wang and McCallum, 2006), because the topics discussed on each address are so diverse and in order to improve the robustness of the discovered topics, we increase the number of documents in the dataset by splitting each transcript into 3-paragraph "documents".

The final result is written in bag-of-words format and is stored in a total of 214 text files, one for each address, and two extra files, one containing the vocabulary (around 22,000 words) and another one which contains the number of documents

[5] http://www.gutenberg.org/ebooks/5050

on each address.

6.2 Publication of the dataset

We make our new dataset, as well as the Python and R code used to create it, publicly available using Github, so that it can be modified and improved in the future. The project can be found at

https://github.com/ourii/Temporal-State-of-the-Union-dataset

[6]http://www.nltk.org/

7 Topic modeling visualization

Representing high dimensional data is in general a difficult problem. Most of the approaches consist on projecting the data to some low dimensional space or visualizing only a subsample of the whole set.

Some research has been focusing on understanding human perception about colors, shapes and size, reason why many research groups have been working on the design of kernels that can assess how distance is perceived by humans between different objects.

Crowd-sourcing is one of the most popular tools to assess human perception. A common aproach is to show users pairs of objects and ask them to evaluate the distance they perceive there is between each pair. Afterwards, the results have to be evaluated and kernels can be built to represent the perceived distance between objects of different colors, shapes and size by using different judgement types. Examples of these kernels, extracted from (Demiralp et al., 2014), are given in Figure 5.

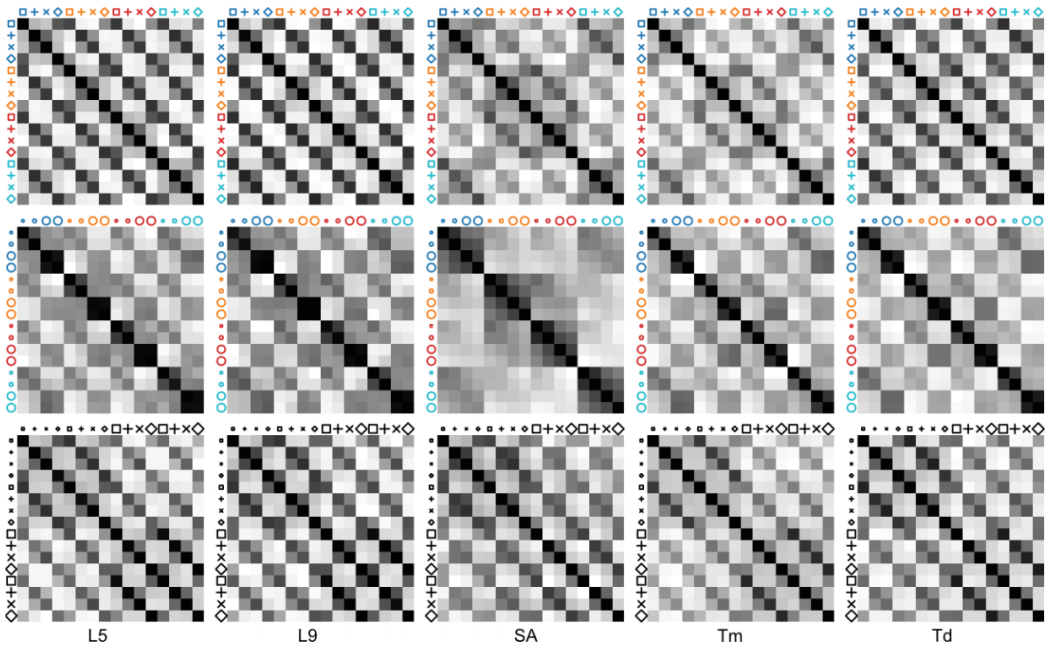


Figure 5: Heat plot of the estimated bivariate kernels between objects of different color and shape.

7.1 Representing topic modeling with parallell coordinates

In the case of topic modeling we are interested in visualizing the word-topic probability matrix, a matrix that contains the probability of each word given each of the recovered topics of a corpus. The word-topic matrix is formally defined as $A \in \mathbb{R}^{V \times H}$ such that $A_{ij} = P(x_i|h_j)$, where h_j is the topic j, x_i is the word i, H is the number of topics and V is the vocabulary size.

We are interested in a visualization that shows how a LDA model is learned using tensor methods. Specifically, we focus on a project that is publicly available on Github[7], which performs a similar process than the one described in Section 4 but using alternating least squares instead of stochastic gradient descent. The algorithm performs learning and inference of topic modeles via method of moments using tensor decomposition on a single machine. Specifically the code first performs a pre-processing which implements a whitening transformation for matrix/tensor orthogonalisation and dimensionality reduction and then runs Alternating Least Squares (ALS) to reconstruct the whitened third order moment. Our goal is to represent the recovered topics on each step of the ALS algorithm and evaluate the convergence.

While an animation seems to be the most logical way to represent each iteration of the algorithm, each step presents a challenging visualization problem. We choose parallell coordinates as our visualization method.

In parallell coordinates we aim to represent a set of points that belong to a n-dimensional space. First, n parallel, equally spaced and vertical lines are drawn. Each of the lines represents one of the n dimensions, and each data point is represented as a set of n vertices that are drown on each of the n lines. This vertices, which represent the coordinates of each point, are joined with the vertices corresponding to the same data point in contiguous dimensions.

In the case of topic modeling, each of the lines (or dimensions) represent one of the learned topics, and vertices represent the probability of each word conditional on the topics (each vertex corresponds to the probability of a word given a specific topic).

We aim to represent the topics found in a dataset of New York Times articles (Lichman, 2013). This dataset contains approximately $102,660$ distinct words, $300,000$ documents and $100,000,000$ words in total.

In order to have a clean visualization we choose to run our topic modeling algorithm with 20 topics (the algorithm will detect up to 20 topics in our corpus), and we choose to represent only 5 of them, which we tag as economy, education, sports, online social media and crime reports. For each of these 5 topics, represented by their most common words, we choose its top 10 most probable words, each of them being a data point of our 20-dimensional space. To facilitate the visualization of the datapoints we use different colors to represent each topic, and each of the words is painted with the color of the topic they belong to.

7.2 Implementation

We implement our parallell coordinates visualization on a website designed with HTML and CSS.

[7] https://github.com/FurongHuang/TensorDecomposition4TopicModeling

7.2.1 Running the algorithm on a web browser

We first considered implementing the topic modeling algorithm on the web browser using JavaScript, but the memory constrains of most web browsers made us discard this possibility. Indeed, before starting to recode the project into JavaScript, we tried to compile the C++ code into JavaScript using Emscripten[8] but memory contrains were too tight for our purposes.

Emscripten is a source-to-source compiler which takes LLVM bitcode as input, generated from the C++ code, and compiles that into JavaScript, which can be run on the web. Unlike C or C++, JavaScript is an interpreted programming language, what usually implies lower performance. One of the advantages of Emscripten is that the generated JavaScript code has always been preprocessed and optimized before running, what makes it, in general, much more efficient than hand-written JavaScript code.

In our case, and as expected, JavaScript code generated by Emscripten run slower than the C++ code (around 10 times slower). However, the main limitation was found in the amount of memory that web browsers can handle, as it wasn't enough to load our training dataset (we were running the code with the same New York Times bag of words dataset described before).

As a conclusion, it was a good decision to not start recoding some several thousand lines of C++ code into JavaScript, as we would probably have eventually run into the same barriers we found using Emscripten. Afterwards, we considered two alternatives: either running an executable file on the browser or just visualizing some previously generated data. As getting results on the New York Time dataset was a time consuming process (it easily takes more than 1 minute using OSX on a Macbook Pro with 2.9 GHz Intel Core i5 and 16 GB 1867 MHz DDR3) we finally decided to just make the visualization using previously generated data.

7.2.2 Visualization using D3

We proceeded to implement our parallell coordinates visualization using JavaScript. As it is one most used and powerful libraries for visualization, we decided to use D3 to represent the parallel coordinates.

D3 [9] (Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards. It is the successor to the earlier Protovis framework.

In order to represent parallell coordinates we use a library of D3 called Parcoords[10]. This library provides a basic parallell coordinates template that we modified to satify our needs and represent our data. In our visualization many iteration steps of the ALS algorithm are represented using a timer function and adding some buttons to pause, stop and resume the animation. Some on-hover interactive efects

[8]https://github.com/kripken/emscripten
[9]https://d3js.org/

that trigger when the mouse is moved to one of the lines or vertices of the plot were also added.

The final interactive visualization can be found on the following link:

http://newport.eecs.uci.edu/anandkumar/Lab/Lab_sub/NewYorkTimes3.html

And its source code has been made publicly available at

https://github.com/ourii/NYTimes_visualization_demo/tree/master

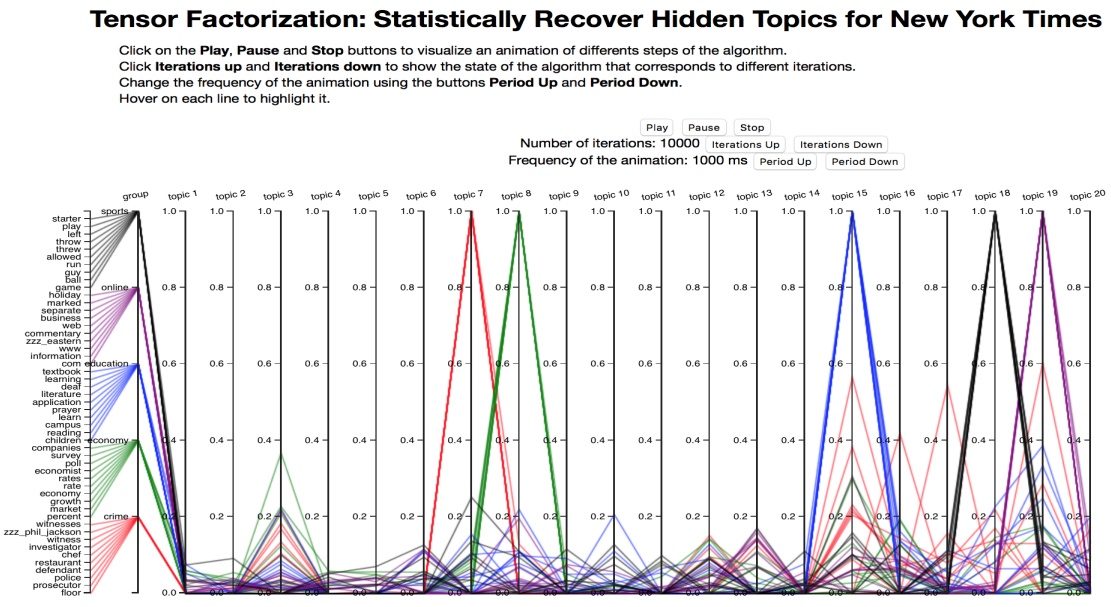
In Figure 6 we aim to provide overview screenshots of our visualization as well as of its interactive characteristics and in Figure 7 we provide screenshots of the representation that aim to show some of its states on a few different number of iterations (which were activated using the Play, Pause and Stop buttons).

Last, it is worth mentioning that the visualization was cited in the issue #104 of the Data Science Weekly newslatter[11].

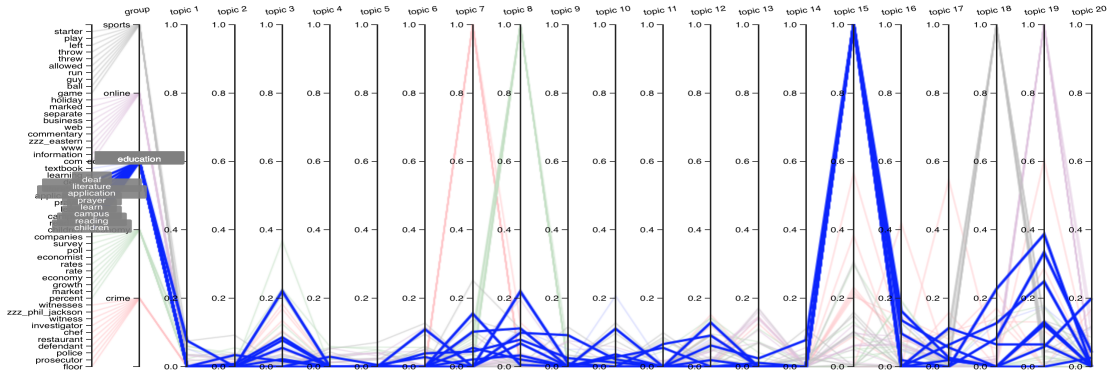
[10]https://syntagmatic.github.io/parallel-coordinates/

[11]http://www.datascienceweekly.org/

Figure 6: Overview of our visualization for topic modeling.

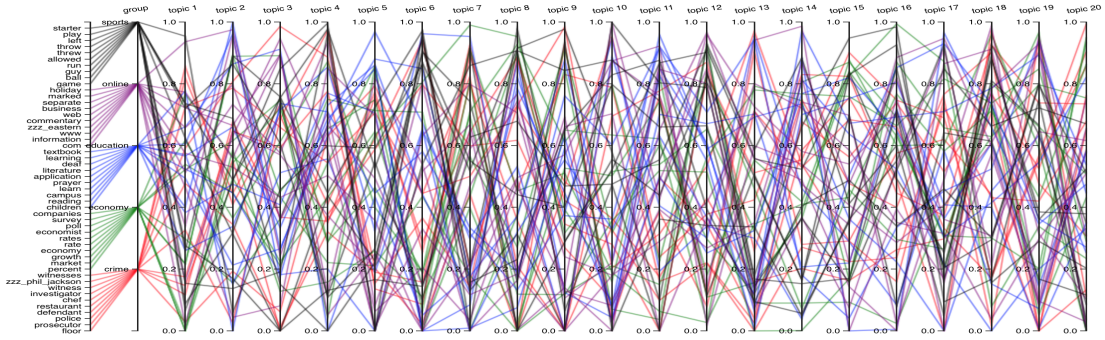


(a) Screenshot of the topic modeling visualization described in Section 7.2.2.

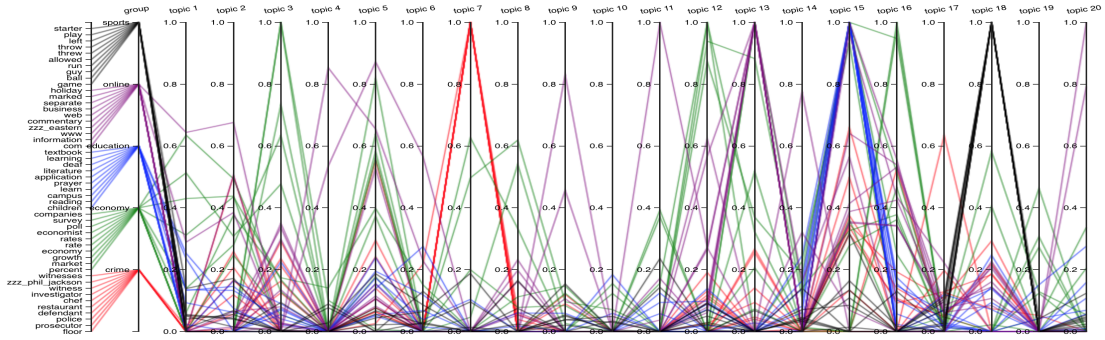


(b) Words and their corresponding topics are highlighted on hover (when the mouse is placed on one of the lines of each word).

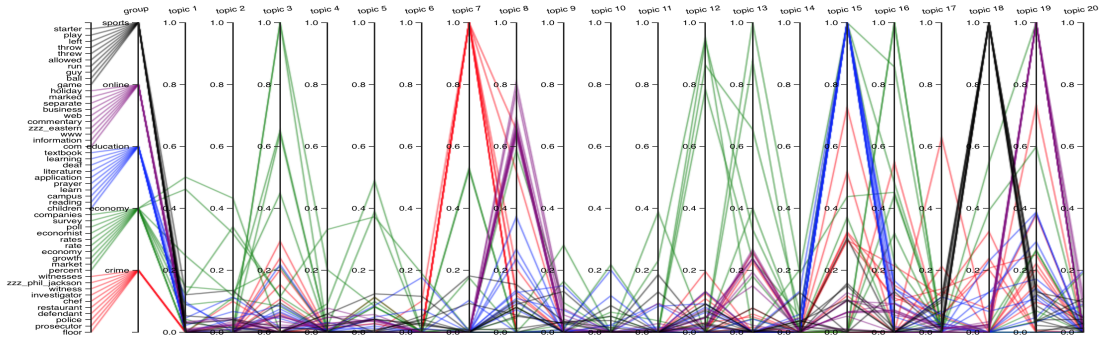
Figure 7: Top words found on 5 of the topics detected by LDA using tensor methods.



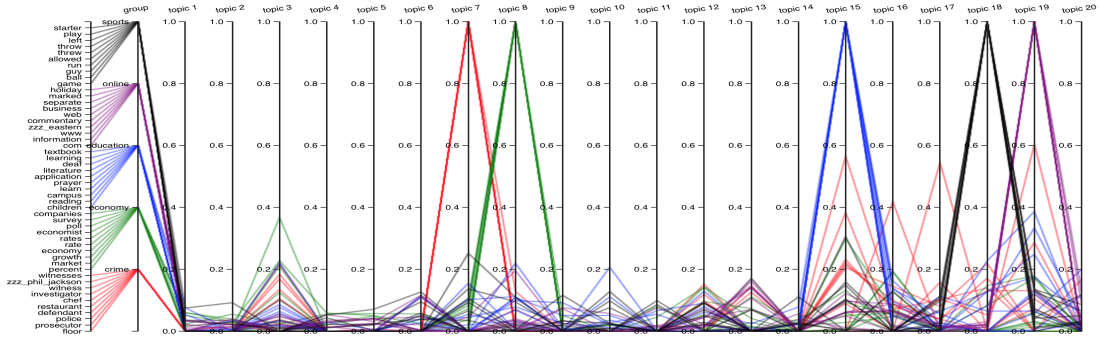
(a) Random initialization before running ALS.



(b) Word-topic distributions after 1,000 iterations.



(c) Word-topic distributions after 5,000 iterations.



(d) Word-topic distributions after 10,000 iterations.

8 Conclusions

This report summarizes part of the work done during a stay of 6 months at the University of California Irvine. We start by reviewing Latent Dirichlet Allocation (LDA) and explain how tensor methods can be used to learn LDA. Next, we discuss and implement a variation of an already existing method for learning LDA using tensor methods, and present a new approach for temporal topic modeling. Finally, we publish a new dataset for temporal topic modeling as well as present an innovative visualization for the recovered word-topic probabilities.

Further research could involve an implementation of Algorithm 1, for which we have already provided a great part of the code in Section 4.7.1. It would be interesting to test how effective the method is and how strongly the hypothesis of Gaussian correlation between the eigenvectors is transmitted to the learnt topics. Note that, to make this experiments, we have already arranged and made publicly available a dataset for temporal topic modeling, as described in Section 6.

It would also be interesting to make more experiments on the effectiveness of tensor methods for learning LDA. While some work, such as Table 1, show already good results in terms of the topics recovered using tensor methods, we believe greater comparison with the topics found using other methods should be made. A comparison with variational methods can easily be done using the code from the LDA-C project: http://www.cs.columbia.edu/~blei/lda-c/.

We have found that our estimations of the eigenvectors of the whitened third order tensor have difficulties to reconstruct the original tensor. Precisely, we have observed that the estimated eigenvectors allow for a good recovery of the tensor only for $k \leq 3$, being k the number of topics. For $k > 3$, the Frobenius norm between the recovered tensor and the whitened empirical third order moment is of the order of many powers of 10 and an inspection of the two tensors doesn't allow for the discovery of many similarities. However, the recovered topics keep being meaningful.

This lack of reconstructing capacity is due to the fact that we try to reconstruct the tensor using only k eigenvectors, that is to say, we reconstruct a rank-k tensor, what seems to obviate some of the complexity of the original tensor. This behaviour is why we believe greater comparison between different methods for learning LDA should be a next step, although we have seen that tensor methods can undoubtedly recover significant topics. Indeed, while a rank-k tensor may not be enough to recover all the complexity of the original empirical whitened third order moment, it might be enough to summarise the main trends of the corpus and work as a dimensionality reduction method.

Last, I would like to add some few words to say how difficult it has been to cover some of the parts of this report. As a totally non initiated to research that I was, I sometimes found research papers (which are the base of this report) to be little explanatory and hard to reproduce, and tensor problems usually require a very meticulous notation that I found difficult to follow at the beginning.

References

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. *arXiv preprint arXiv:1203.0683*, 2012.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

R. Bellman and R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton Legacy Library. Princeton University Press, 1961. URL `https://books.google.es/books?id=POAmAAAAMAAJ`.

D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022, 2003.

C. D. Demiralp, M. S. Bernstein, and J. Heer. Learning perceptual kernels for visualization design. *IEEE transactions on visualization and computer graphics*, 20(12):1933–1942, 2014.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

J. M. Dickey. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *Journal of the American Statistical Association*, 78(383):628–637, 1983.

A. Dubey, A. Hefny, S. Williamson, and E. P. Xing. A nonparametric mixture model for topic modeling over time. In *SDM*, pages 530–538. SIAM, 2013.

B. Everett. *An introduction to latent variable models*. Springer Science & Business Media, 2013.

R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points-online stochastic gradient for tensor decomposition. *arXiv preprint arXiv:1503.02101*, 2015.

L. Hong, B. Dom, S. Gurumurthy, and K. Tsioutsiouliklis. A time-dependent topic model for multiple text streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 832–840. ACM, 2011.

F. Huang, U. Niranjan, M. U. Hakeem, and A. Anandkumar. Online tensor methods for learning latent variable models. *The Journal of Machine Learning Research*, 16:2797–2835, 2015.

H. J. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`. Version 20121115.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, pages 1–8, 2011.

J. TENBERGE. Kruskal polynomial for 2x2x2 arrays and a generalization to 2xnxn arrays. *Psychometrika*, 56(4):631–636, 12 1991. ISSN 0033-3123.

C. Wang, D. Blei, and D. Heckerman. Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*, 2012.

X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.

Wikipedia. Collaborative filtering — Wikipedia, the free encyclopedia, 2016. URL `https://en.wikipedia.org/wiki/Collaborative_filtering`. [Online; accessed 12-May-2016].

L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM, 2010.

Appendix A Latent variable models

Researchers in a wide range of disciplines often seek a better understanding of the relationship of two observed variables by trying to discover whether their relationship can be explained by a third variable. Latent variable models are statatistical models that try to find an explanation to the relation between the observed variables using latent or hidden variables.

Let $X = [X_1 \ \dots \ X_n]$ represent the manifest or observed variables and $Y = [Y_1 \ \dots \ Y_n]$ denote the latent variables. Normally the number of latent variables is much lower than that of the manifest variables, and latent variables models can be interpreted as dimensionality reduction methods.

Latent variables assume that the manifest variables X have a joint probability distribution conditional on the latent variables Y. If we assume both X and Y are continuous random variables and that the density function of Y is g, the unconditional density of X is given by

$$f(X) = \int p(X|Y)g(Y)dY$$

where $p(X|Y)$ is the joint probability distribution of X conditional on Y.

In general, we are interested in estimating the density functions p and g in order to learn the relationships between our variables. The crucial assumption to make in order to solve the problem is that of conditional independence, which states that the observed variables are independent given the values of the latent variables. Equivalently, we can write:

$$p(X|Y) = p_1(x_1|Y) \cdots p_n(X_n|Y)$$

Note that the property of conditional independence means that the relation between the manifest variables is due to the latent variables. It is also normally assumed that we know the distribution of X and Y except some paremeters that we need to infer.

More information on latent variable models can be found in (Everett, 2013).

Appendix B The matrix Normal distribution

The matrix normal distribution is a probability distribution that is a generalization of the multivariate normal distribution to matrix-valued random variables.

The probability density function for a random matrix X $(n \times p)$ that follows the matrix normal distribution $\mathcal{MN}_{n,p}(\mathbf{M}, \mathbf{U}, \mathbf{V})$ has the form:

$$p(\mathbf{X} \mid \mathbf{M}, \mathbf{U}, \mathbf{V}) = \frac{\exp\left(-\frac{1}{2}\operatorname{tr}\left[\mathbf{V}^{-1}(\mathbf{X}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{X}-\mathbf{M})\right]\right)}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{p/2}}$$

where tr denotes trace, $|\cdot|$ denotes the determinant and M is $n \times p$, U is $n \times n$ and V is $p \times p$.

Unlike the multivariate Normal distribution, which produces random vectors, the matrix Normal distribution draws random matrices that depend on three paremeters: one mean matrix $\mathbf{M}$ and two covariance matrices $\mathbf{U}$ and $\mathbf{V}$, which are responsible for the covariances within the elements of each column and row of the random matrices, respectively.

To understand how the matrix Normal distribution works and the role of each of the two covariance matrices we have made a simulation using R, for which we have represented the results in Figure 8.

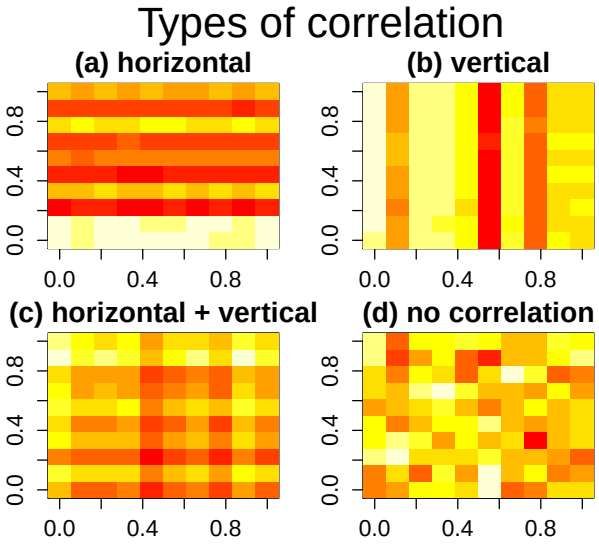


Figure 8: Heat plot of a 10×10 matrix drawn from a Matrix Normal distribution $\mathcal{MN}_{10,10}(\mathbf{0}, \mathbf{A}, \mathbf{B})$ with different covariance matrices $\mathbf{A}$ and $\mathbf{B}$ which infer correlation in different directions between the elements of the random matrix. Given a matrix $\mathbf{V} \in \mathbb{R}^{10\times10}$ determined by $(\mathbf{V})_{\mathbf{ij}} = 1 \; if \; i = j \; \wedge \; 0.95 \; otherwise$, each of the subplots has been drawn from the following distributions: Figure 8 (a) $\mathcal{MN}_{10,10}(\mathbf{0}, \mathbf{Id}, \mathbf{V})$, Figure 8 (b) $\mathcal{MN}_{10,10}(\mathbf{0}, \mathbf{V}, \mathbf{Id})$, Figure 8 (c) $\mathcal{MN}_{10,10}(\mathbf{0}, \mathbf{V}, \mathbf{V})$, Figure 8 (d) $\mathcal{MN}_{10,10}(\mathbf{0}, \mathbf{Id}, \mathbf{Id})$