Noname manuscript No.
(will be inserted by the editor)

A New Method of Moments for Latent Variable Models

Matteo Ruffini · Marta Casanellas · Ricard Gavaldà

Received: date / Accepted: date

Abstract We present an algorithm for the unsupervised learning of latent variable models based on the method of moments. We give efficient estimates of the moments for two models that are well known e.g. in text mining, the single-topic model and latent Dirichlet allocation, and we provide a tensor decomposition algorithm for the moments that proves to be robust both in theory and in practice. Experiments on synthetic data show that the proposed estimators outperform the existing ones in terms of reconstruction accuracy, and that the proposed tensor decomposition technique achieves the learning accuracy of the state-of-the-art method with significantly smaller running times. We also provide examples of applications to real-world text corpora for both single-topic model and LDA, obtaining meaningful results.

Keywords Spectral Methods · Method of Moments · Latent Variable Models · Topic Modeling

1 Introduction

Latent variable models (LVM) are a wide class of parametric models characterized by the presence of some hidden *unobservable* variables that influence the observable data. A lot of widely used models belong to this class: Gaussian mixtures, latent Dirichlet allocation, naïve Bayes and hidden Markov models and many others. The huge availability of data and the diverse real-world applications, like health-care data mining, text analytics, vision and speech processing, has boosted the need for accurate and fast algorithms to learn models belonging to this class.

For many decades, the standard approach to learn such models has been the Expectation Maximization method (EM) (Dempster et al.), an iterative technique based on the maximum likelihood principle. EM is easy to understand and to implement and can be used for any latent variable model. Nevertheless, EM has many drawbacks: it is known to produce suboptimal results and may be very slow when the model or data dimension grow (Balle et al., 2014).

To overcome these issues a variety of techniques based on the *method of moments* and tensor decomposition have been proposed in the last years (Anandkumar et al., 2014, 2012a,b; Jain and Oh, 2014; Hsu and Kakade, 2013; Balle et al., 2014; Chaganty and Liang, 2013; Song et al., 2011). Generalizing an approach introduced by Pearson (1894), these methods first store in multidimensional tensors the low-order moments of the observable data, and then recover the parameters of the desired model via tensor decomposition techniques. Moment-based algorithms are in general faster than EM as they require a single pass over the data, and have provable guarantees of learning accuracy in polynomial time.

Anandkumar et al. (2014) presented an exhaustive survey showing that the spectral learning of most of the LVMs can be abstracted in two steps: first, given a prescribed LVM, they show how to operate with the data to obtain an empirical estimate of the moments, expressed in the form of symmetric low rank matrices and tensors; in a second step, the moments are decomposed using tensor/matrix decomposition

mruffini@cs.upc.edu

marta.casanellas@upc.edu

gavalda@cs.upc.edu

Universitat Politècnica de Catalunya and BGSMath

techniques, to obtain the unknown parameters of the model. That paper also provides a method to derive such decomposition, the *Tensor Power Method* (TPM). This abstraction allows to split the research on methods of moments in two main non-disjoint areas.

A first area concentrates on methods to retrieve from data empirical estimates of the moments for specific LVMs. Examples are the works by Jain and Oh (2014) for mixture models with categorical observations, by Chaganty and Liang (2013) for mixtures of linear regressions, by Anandkumar et al. (2012b) for naïve Bayes models, and by Hsu et al. (2012) for hidden Markov models. Important examples come from text mining, where the observable data, called *features*, generally coincides with the words appearing in a document, while the hidden variable can be, for example, the topic of the document. A simple model is the *single-topic model*, where each text is assumed to be about only one topic and the probability of a given word appearing in a text depends on the topic of the text; equations to retrieve a moment representation for this model are provided by Anandkumar et al. (2012b) and Zou et al. (2013). An alternative and more complex model is Latent Dirichlet Allocation (LDA) (see Griffiths and Steyvers, 2004; Blei et al., 2003), for which Anandkumar et al. (2012a) provide empirical estimators of the moments.

A second research area focuses on methods to decompose the low-order moments to retrieve the parameters of the model. Literature on tensor decomposition is vast and it is believed to originate from the works by Hitchcock (1927, 1928). The topic received attention in the field of psychometrics (Carroll and Chang, 1970; Harshman, 1970; Tucker, 1966) and chemometrics (Appellof and Davidson, 1981) and extensive surveys are provided by Tomasi and Bro (2006); Kolda and Bader (2009) and Sidiropoulos et al. (2017). These works also present several classical techniques to obtain the so-called *canonical polyadic decomposition* (CPD) of a three dimensional tensor, which expresses it as a linear combination of rank-1 tensors. Kolda and Bader (2009) recall one of the most popular algorithms to obtain the CPD of a generic tensor: *Alternating Least Square* (ALS) (whose origins traces back to Carroll and Chang, 1970; Harshman, 1970). ALS is an heuristic that works by fixing all-but-one of the factors decomposing a tensor, and updating the remaining factor by solving an overdetermined linear least squares problem. ALS is easy to implement and to understand but is known to be prone to local minima, requiring several random restarts to return meaningful results (see the discussion by Kolda and Bader, 2009, pg. 18). A different line of work consists in approaching the tensor decomposition problem with simultaneous Schur decompositions (Van Der Veen and Paulraj, 1996; De Lathauwer et al., 2004; Colombo and Vlassis, 2016) or matrix optimization methods to perform simultaneous diagonalization (Kuleshov et al., 2015). In theory, one could use any of these methods to decompose the three-dimensional tensor containing the third order moments, but the high dimensionality of that tensor makes this approach often computationally unfeasible. On the other hand, one can rely on the specific structure of the whole set of low-order moments by using not only the third, but also the first and the second order moments to explicitly exploit their symmetry properties and obtain the desired model parameters more efficiently and with provable guarantees of global optimality. This is the approach followed by methods of moments.
The reference method here is the *Tensor Power Method* (TPM) from Anandkumar et al. (2014). TPM manipulates the low-order moments to obtain a symmetric orthogonal tensor that is decomposed using a high-dimensional extension of the well-known matrix power method, with an approach proposed by Zhang and Golub (2001) and Kolda (2001). The main issue of this algorithm is its scalability, as its running time grows as k^5 where k is the number of latent components. An alternative, which in general has better dependence on the number of latent factors, consists in using matrix-based techniques and a *simultaneous diagonalization* approach, specializing an approach that can be traced back to Sanchez and Kowalski (1990) and Leurgans et al. (1993). Examples of these methods of moments are provided by Anandkumar et al. (2012b), with an algorithm based on the eigenvectors of a linear operator and by Anandkumar et al. (2012a) that present an alternative method based on the singular vectors of a singular-value decomposition (SVD). These two methods are faster than TPM by far, but they are more sensitive to perturbations on the input data.

A method of moments uses a set of N samples to empirically estimate the moments and then uses tensor decomposition methods to recover the model parameters from the decompositions of the estimated, perturbed, moments. The decomposition methods listed above differ from each other on how perturbations of the input moments affect the final results. A method of moments can thus be improved by modifying two distinct points; one is acting on the procedures to estimate the moments, by reducing their dependence on the sample size and get more accurate estimates with less data. Another is by working on the decomposition algorithms, reducing their sensitivity to input perturbations and improving their performance. This paper presents improvements in both directions; in particular:

– We present improved estimators of the moments for the single-topic model and latent Dirichlet allocation. These estimators modify the ones presented by Zou et al. (2013) increasing the robustness and the stability with respect to the noise, as experimentally shown in Section 4. Also, we present a novel theorem that relates the sample accuracy of the proposed estimates to the sample size and to the lengths of the documents.
– We provide a new algorithm (named SVTD, ***S**ingular **V**alue based **T**ensor **D**ecomposition*) to decompose the retrieved low-rank symmetric tensors of the moments. This method is alternative to the ones presented in the cited literature, and is based on the singular values of a SVD, which are known to be stable under random perturbations (unlike the singular vectors, as shown by Stewart and Sun 1990). Our algorithm is simple to implement and understand, is deterministic and based only on standard matrix operations rather than tensor ones. Experimental results (see Section 4) show that SVTD outperforms existing matrix-based methods from Anandkumar et al. (2012a,b) in terms of reconstruction accuracy, is an order of magnitude faster than TPM, and uses far less memory. In Section 3.1 we outline in more detail the differences between the presented method and the existing techniques.
– Finally, we test SVTD on real-world text corpora, both for single-topic model and LDA, with satisfactory and meaningful results.

The outline of the paper is the following: Section 2 contains the proposed estimators of the moments and a sample complexity bound; Section 3 contains the proposed decomposition algorithm; Section 4 tests the presented methods on both synthetic and real-world data; Section 5 concludes the paper outlining possible future developments and applications.

2 Moments Estimation for Latent Variable Models

2.1 An Improved Estimator for the Single-Topic Model

In this section we recall the single-topic model for which we introduce a new estimator of the moments; we then provide a sample accuracy bound for this estimator and we compare it with existing methods.

We consider a corpus of N text documents and a set of k topics; each document is deemed to belong to only one topic. The vocabulary appearing in the corpus is constituted of n words, from which it is immediate to label all the words of the vocabulary with a number between 1 and n. The generative process works as follows:

– First, a (hidden) topic $Y \in \{1, ..., k\}$ is drawn, according to a given probability distribution; we define, for any $j \in \{1, ..., k\}$ the probability of drawing the topic j as follows:

$$\omega_j := \mathbb{P}(Y = j), \text{ and } \omega := (\omega_1, \dots, \omega_k)^\top.$$

– Once the topic has been chosen, all the words of the documents are generated according to a multinomial distribution; for each $i \in \{1, ..., n\}$, $\mu_{i,j}$ will be the probability of generating word i under topic j:

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j}, \text{ and } M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

Also we will denote by μ_i the set of columns of M: $M = [\mu_1|, ..., |\mu_k]$. It is a common practice to identify a topic with the probability distribution of the words under that topic, i.e. with the columns $\mu_1, \dots, \mu_k$ of M.

A practical encoding of the words in a document consists of identifying each word with an $n-$tuple $x \in \mathbb{R}^n$, defined as:

$$(x)_h := \begin{cases} 1 & \text{if the word is } h, \\ 0 & \text{otherwise.} \end{cases}$$

In fact, if x_j is the jth word of a document of c words, we can define a vector X whose coordinate h represents the number of times the word h has appeared in the document: $X := \sum_{j=1}^{c} x_j$. It is common to call X a *bag of words* representation of a document. We can see that, if the topic is j, each coordinate of X is distributed as a binomial distribution with parameter c and $\mu_{i,j}$:

$$Distr(X_i | Y = j) = B(c, \mu_{i,j}).$$

We now consider a corpus of N documents; for each document $i \in \{1, ..., N\}$ we assume knowing the word-count vector $X^{(i)}$ as above, and the total number of words in the document: $c_i = \sum_{j=1}^{n} (X^{(i)})_j$. These are the only variables that we assume are known, while all the parameters of the model, i.e. the pair (M, ω), and the hidden topic of each document are supposed to be unknown.

Remark 1 Recovering the model parameters is a useful step to infer the hidden topic of each document in a corpus. In fact, given a set of parameters (M, ω), and a document X, if Y is the hidden topic of X we can calculate

$$\mathbb{P}(Y = j|X) = \frac{\mathbb{P}(X|Y = j)\,\omega_j}{\sum_{i=1}^{k} \mathbb{P}(X|Y = i)\,\omega_i}, \tag{1}$$

and assign X to the topic that maximizes that probability.

The following theorem is a variation of Propositions 3 and 4 by Zou et al. (2013) and relates the observable moments of the known variables with the unknowns (M, ω). It provides three estimators: $\tilde{M}_1 \in \mathbb{R}^n$, $\tilde{M}_2 \in \mathbb{R}^{n \times n}$ and $\tilde{M}_3 \in \mathbb{R}^{n \times n \times n}$ converging to the symmetric low rank tensors that will be used to retrieve the model parameters.[1]

Theorem 1 Fix a value $N \in \mathbb{N}$, and let $X^{(1)}, ..., X^{(N)}$ be N sample documents generated according to a single-topic model with parameters (M, ω). Define the vector $\tilde{M}_1 \in \mathbb{R}^n$, and the symmetric tensors $\tilde{M}_2 \in \mathbb{R}^{n \times n}$ and $\tilde{M}_3 \in \mathbb{R}^{n \times n \times n}$ whose entries are as follows, for $h \le l \le m$:

$$\begin{aligned}
(\tilde{M}_1)_h :=& \frac{\sum_{i=1}^{N} (X^{(i)})_h}{\sum_{i=1}^{N} c_i}, \\
(\tilde{M}_2)_{h,l} :=& \frac{\sum_{i=1}^{N} (X^{(i)})_h (X^{(i)} - \chi_{h=l})_l}{\sum_{i=1}^{N} (c_i - 1) c_i}, \\
(\tilde{M}_3)_{h,l,m} :=& \chi_{h<l<m} \frac{\sum_{i=1}^{N} (X^{(i)})_h (X^{(i)})_l (X^{(i)})_m}{\sum_{i=1}^{N} (c_i - 2)(c_i - 1) c_i} \\
&+ \chi_{(h=l<m)\vee(h<l=m)} \frac{\sum_{i=1}^{N} (X^{(i)})_h (X^{(i)} - 1)_l (X^{(i)})_m}{\sum_{i=1}^{N} (c_i - 2)(c_i - 1) c_i} \\
&+ \chi_{h=l=m} \frac{\sum_{i=1}^{N} (X^{(i)})_h (X^{(i)} - 1)_l (X^{(i)} - 2)_m}{\sum_{i=1}^{N} (c_i - 2)(c_i - 1) c_i},
\end{aligned}$$

where χ is the indicator function and c_i is the number of words appearing in document i. We have:

$$\mathbb{E}[\tilde{M}_1] = M_1 := \sum_{i=1}^{k} \omega_i \mu_i, \qquad \mathbb{E}[\tilde{M}_2] = M_2 := \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i, \qquad \mathbb{E}[\tilde{M}_3] = M_3 := \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i \otimes \mu_i,$$

where $\otimes$ denotes the tensor product between vectors.

Notice that, because M_2 and M_3 are symmetric, the previous theorem defines all the entries of these tensors. Given a sample, we are able to calculate the three estimators $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$. Theorem 1 allows to express observable moments in the form of a symmetric tensor. By construction it is immediate to see that both M_2 and M_3 have symmetric-rank less than or equal to k, and that $\lim_{N\to\infty} \tilde{M}_i = M_i$ for $i = 1, 2, 3$.

The following theorem provides a sample accuracy bound for the estimators of Theorem 1.

Theorem 2 Let $\tilde{M}_2$ and $\tilde{M}_3$ be the empirical estimates of M_2 and M_3 obtained using Theorem 1, and let $(c_1, ..., c_N)$ be a vector containing the lengths of the various documents of the corpus. Define

$$C_1 = \sum_{i=1}^{N} c_i, \quad C_2 = \sum_{i=1}^{N} (c_i - 1) c_i, \quad C_3 = \sum_{i=1}^{N} (c_i - 2)(c_i - 1) c_i,$$

$$W_2^{(N)} = \frac{\sum_{i=1}^{N} (c_i (c_i - 1))^2}{C_2^2}, \quad W_3^{(N)} = \frac{\sum_{i=1}^{N} (c_i (c_i - 1)(c_i - 2))^2}{C_3^2}.$$

[1] The proofs of all the results are provided in the Appendix.

Then, for any pair $\epsilon > 0$ and $0 \leq \delta < 1$, such that

$$\sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + \sqrt{\log(\frac{1}{\delta})}\frac{\max_j(c_j)\sqrt{C_1}}{C_2} < \epsilon,$$

it holds that

$$\mathbb{P}(||M_2 - \tilde{M}_2||_F < \epsilon) > 1 - \delta.$$

Also, for any pair $\epsilon > 0$ and $0 \leq \delta < 1$, such that

$$\sqrt{W_3^{(N)}(1 - ||M_3||_F^2)} + \sqrt{\log(\frac{1}{\delta})}\frac{\max_j(c_j(c_j-1))\sqrt{C_1}}{C_3} < \epsilon,$$

it holds that

$$P(||M_3 - \tilde{M}_3||_F < \epsilon) > 1 - \delta.$$

Remark 2 We briefly comment on the meaning of the theorem by focusing on M_2 (similar arguments hold for M_3) and analyzing the case where all the documents have the same length c ($c_i = c$ for all i) and c is asumed to be somewhat large (so $c \simeq c - 1$). Then the bound simplifies to:

$$\sqrt{\frac{1}{N}(1 - ||M_2||_F^2)} + \sqrt{\frac{1}{Nc}\log(\frac{1}{\delta})}.$$

It is interesting to notice that the worst-case accuracy of the bound is $\epsilon = O(1/\sqrt{N})$. Also, when c increases, the bound becomes smaller, with a clear limitation: if we have very few documents (N small), even if they are very long (large c), it is impossible to accurately learn the model, as in particular we may not even see all the topics.

Remark 3 (Alternative estimates of the moments) The simplest estimation of the low-order moments for the single-topic model is provided by Anandkumar et al. (2012b), who, for each document $i \in \{1, ..., N\}$ consider three randomly selected words, $x_1^{(i)}, x_2^{(i)}, x_3^{(i)}$, and then show that

$$\frac{\sum_{i=1}^N (x_1^{(i)})_h}{N} \xrightarrow[N\to\infty]{} (M_1)_h, \quad \frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l}{N} \xrightarrow[N\to\infty]{} (M_2)_{h,l},$$

$$\frac{\sum_{i=1}^N (x_1^{(i)})_h (x_2^{(i)})_l (x_3^{(i)})_m}{N} \xrightarrow[N\to\infty]{} (M_3)_{h,l,m}.$$

This method only uses three words per document, and has mainly illustrative purposes, as noticed by the authors. A method more similar to the one proposed in Theorem 1 is described by Zou et al. (2013). Both the method by Zou et al. (2013) and that in Theorem 1 average the estimators with the document lengths, taking into consideration all the words in each document. However, in the method by Zou et al. (2013), the averaging is done for each document and then they are averaged together with the same weight; for example, Zou et al. (2013) calculate the off-diagonal entries of $\tilde{M}_2$ as follows:

$$(\tilde{M}_2)_{h,l} := \frac{1}{N}\sum_{i=1}^N \frac{(X^{(i)})_h (X^{(i)})_l}{(c_i - 1)c_i}.$$

This calculation is a simple average of many estimators giving the same weight to all documents, namely $1/N$. Instead, in Theorem 1, we propose the following different formula:

$$(\tilde{M}_2)_{h,l} := \frac{\sum_{i=1}^N (X^{(i)})_h (X^{(i)})_l}{\sum_i (c_i - 1)c_i} = \sum_{i=1}^N \frac{(X^{(i)})_h (X^{(i)})_l}{(c_i - 1)c_i}\frac{(c_i - 1)c_i}{\sum_j (c_j - 1)c_j}.$$

We can see that here we perform a weighted average, where the weight of the sample i is $\frac{(c_i-1)c_i}{\sum_j (c_j-1)c_j}$, giving in practice more weight to longer documents, which are supposed to be the most reliable; we will experimentally see in Section 4 that the proposed approach is less sensitive to noise, leading to improved results. If all the documents have the same length, the two estimates will produce the same number.

2.2 Extension to Latent Dirichlet Allocation

In this section we extend the results of the previous section to a more complex latent variable model, the latent Dirichlet allocation.

The obvious criticism of the single-topic model is that each document can deal with a unique topic, a hypothesis that is commonly considered unrealistic. To overcome this issue, more complex models have been introduced, one of them being latent Dirichlet allocation (LDA) (Griffiths and Steyvers, 2004; Blei et al., 2003). In its simplest form, LDA assumes that each document deals with a multitude of topics, in proportions that are governed by the outcome of a Dirichlet distribution. More precisely, considering our text corpus with N documents with a vocabulary of n words, the generative process for each text is the following:

- First a vector of topic proportions is drawn from a Dirichlet distribution with parameter $\alpha \in \mathbb{R}_+^k$, $Dir(\alpha)$; we recall that Dirichlet distribution is distributed over the simplex

$$\Delta^{k-1} = \{v \in \mathbb{R}^k : \forall i, v_i \in [0,1],\ and\ \sum v_i = 1\},$$

  and has the following density function, for $h \in \Delta^{k-1}$:

$$\mathbb{P}(h) = \frac{\Gamma(\alpha_0)\prod_{i=1}^k h_i^{\alpha_i - 1}}{\prod_{i=1}^k \Gamma(\alpha_i)}.$$

  Where $\alpha_0 = \sum \alpha_i$. From a practical point of view, this step consists of drawing a vector of parameters $h \in \Delta^{k-1}$ such that h_i represents the proportion of the topic i in the document.
- Once the topic proportions (also named *mixture of topics*) have been chosen, each word of the document is generated according to the following procedure: First a (hidden) topic of the word, say $Y \in \{1, ..., k\}$, is drawn, according to the probabilities defined by h (so we will have probability h_j of drawing topic j) and then we will generate the word itself according to a multinomial distribution; for each $i \in \{1, ..., n\}$, $\mu_{i,j}$ will be the probability of generating the word i under the topic j:

$$\mathbb{P}(\text{Drawing word } i | Y = j) = \mu_{i,j},\ \ and\ \ M = (\mu_{i,j})_{i,j} \in \mathbb{R}^{n \times k}.$$

Keeping the notation used in the previous section, we will write $x_j^{(i)} \in \mathbb{R}^n$ for the coordinate vector indicating the word at position j in document i, $X^{(i)} = \sum x_j^{(i)}$ for the word-count vector of document i, and $c_i = \sum_{j=1}^n (X^{(i)})_j$ for the number of words in that document. In the case of LDA, the unknown model parameters are the pair (M, α).

As in the case of the single-topic model, we want to manipulate the observable moments in order to obtain a set of symmetric low-rank tensors. The following theorem is an immediate modification of the one presented by Anandkumar et al. (2012a, Lemma 3.2), and relates the observable moments of the known variables with the unknowns (M, α), providing the required representation. The only modification consists of the fact that we have used the estimates of Theorem 1 instead of the standard ones of Remark 3.

Theorem 3 Let $\tilde{M_1}, \tilde{M_2}$ and $\tilde{M_3}$ be the empirical estimates defined in Theorem 1. Define

$$\tilde{M_2^\alpha} := \tilde{M_2} - \frac{\alpha_0}{\alpha_0 + 1} \tilde{M_1} \otimes \tilde{M_1},$$

$$\tilde{M_3^\alpha} := \tilde{M_3} - \frac{\alpha_0}{\alpha_0 + 2}(M_{1,2}) + \frac{2\alpha_0^2}{(\alpha_0 + 2)(\alpha_0 + 1)} \tilde{M_1} \otimes \tilde{M_1} \otimes \tilde{M_1},$$

where $M_{1,2} \in \mathbb{R}^{n \times n \times n}$ is a three-dimensional tensor such that

$$(M_{1,2})_{h,l,m} := ((\tilde{M_2})_{h,l}(\tilde{M_1})_m + (\tilde{M_2})_{l,m}(\tilde{M_1})_h + (\tilde{M_2})_{m,h}(\tilde{M_1})_l).$$

Then

$$\mathbb{E}[\tilde{M_2^\alpha}] = \sum_{i=1}^k \frac{\alpha_i}{(\alpha_0 + 1)\alpha_0} \mu_i \otimes \mu_i = M_2^\alpha,$$

$$\mathbb{E}[\tilde{M_3^\alpha}] = \sum_{i=1}^k \frac{2\alpha_i}{(\alpha_0 + 2)(\alpha_0 + 1)\alpha_0} \mu_i \otimes \mu_i \otimes \mu_i = M_3^\alpha.$$

This technique allows to express the observable moments in the form of a symmetric tensor. Both M_2^α and M_3^α have symmetric-rank less than or equal to k, and so we can use any tensor decomposition algorithm to retrieve from them the unknown model parameters (M, α). This approach has several advantages over traditional methods based on Markov Chain Monte Carlo (MCMC) algorithms, like the one presented by Griffiths and Steyvers (2004). While MCMC methods require several passes over the data, the method presented here only uses one pass to compute the moments described in Theorem 3. This guarantees higher efficiency, which is confirmed by our experiments in Section 4.3. Furthermore, if the tensor decomposition method applied to the moments is robust, and provably recovers (M, α) from the moments, then the proposed approach will be guaranteed to recover the parameters of the model generating the data, guarantees that do not seem to exist for MCMC approaches.

Remark 4 (Inference) Similarly to the single-topic model, one of the main usages of LDA is to infer the mixture of hidden topics of each document in a corpus. Unfortunately, an exact formula to perform this inference is not known, but a number of approximate approaches exist, like Gibbs sampling (Griffiths and Steyvers, 2004; Newman et al., 2009) and Expectation Propagation (Blei et al., 2003). In our case, if we assume knowing the values of model parameters (M, α), we can apply a modified Gibbs sampler to infer the topic mixture for a given text; consider a text, whose words are $x_1, ..., x_c$; then, in LDA, each word x_i is generated by a unique topic Y_{x_i}. Using the equations for Gibbs sampling from Griffiths and Steyvers (2004), if Y_{x_i} is the hidden topic of word x_i and Y_{-x_i} is the set of topic assignment for all the words in the document excluded x_i, it can be shown that

$$\mathbb{P}(Y_{x_i} = j | Y_{-x_i}, x_i) \approx \mu_{x_i,j} \frac{n_{-i,j} + \alpha_i}{c - 1 + \alpha_0}, \tag{2}$$

where $n_{-i,j}$ is the number of words assigned to topic j excluding x_i, c is the total number of words in the document and $\mu_{x_i,j}$ is the probability of drawing the word x_i under topic j. So, given a document, first we have to assign to each word a hidden topic, and then update this assignment word by word in a iterative way, using a Monte-Carlo assignment governed by equation (2). Each iteration updates the number of words assigned to a given topic; after a suitable number of iterations, we can estimate the topic mixture for a given document as the vector $h \in \mathbb{R}^k$ such that $h_j = \frac{n_j + \alpha_i}{c + \alpha_0}$, where n_j is the number of words assigned to topic j.

2.3 Generic Representation of the Moments for Latent Variable Models

In the previous sections we have seen, for the single-topic model and for LDA, how to estimate from data three entities M_1, M_2 and M_3 of the form

$$M_1 = \sum_{i=1}^{k} \omega_i \mu_i, \tag{3}$$

$$M_2 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i, \tag{4}$$

$$M_3 = \sum_{i=1}^{k} \omega_i \mu_i \otimes \mu_i \otimes \mu_i. \tag{5}$$

For several latent variable models, an analogous characterization of the moments to the one presented in Equations (3), (4) and (5) is possible, allowing to manipulate a dataset to obtain via linear transformations an empirical estimate of the triple (M_1, M_2, M_3). Examples are Gaussian mixture models (Hsu and Kakade, 2013; Ge et al., 2015), hidden Markov models (Balle et al., 2014) and mixtures of linear regressions (Chaganty and Liang, 2013). This characterization reduces the problem of learning the parameters of a LVM to that of solving the system formed by Equations (3), (4) and (5), which can in general be assessed using matrix/tensor decomposition techniques. The next section is dedicated to this task.

Remark 5 (Identifiability of the model parameters) It is important to remark that the system of Equations (3), (4) and (5) admits (M, ω) as unique real solution (for generic parameters), which guarantees the identifiability of the model. In fact, Comon et al. (2017) guarantee that if Equation (5) has a unique solution and system (3), (4) and (5) has at least one solution, then also the whole system (3), (4) and (5) admits the same unique solution. Generic uniqueness of the real solutions of (5) is guaranteed by known results on tensor decomposition (Chiantini et al., 2017; Qi et al., 2016), and can be verified using Kruskal's (Kruskal, 1977) criterion (see also Kolda and Bader, 2009, Sec. 3.2).

3 Singular Value based Tensor Decomposition

In this section we present an algorithm to retrieve the parameters of a LVM from the triple (M_1, M_2, M_3). The proposed method only relies on matrix decomposition techniques, is deterministic, scales better in time and memory than TPM, and is more robust than the matrix-based ones described before, as confirmed by the experiments (Section 4).

Our method relies on the observation that M_2 and the slices of M_3 admit a representation in terms of matrix products as

$$M_2 = M\Omega M^\top, \tag{6}$$

$$M_{3,r} = M\Omega^{1/2} diag(m_r)\Omega^{1/2} M^\top, \tag{7}$$

where $\Omega = diag(\omega)$, $M_{3,r} \in \mathbb{R}^{n\times n}$ is the rth slice of M_3 and m_r is the rth row of M. As a first step, it stores the whitened slices of M_3 into a three-dimensional tensor H in $\mathbb{R}^{n\times k\times k}$ and then performs n SVD on the slices of H (belonging to $\mathbb{R}^{k\times k}$) obtaining the rows of M as the singular values of that slices; for this reason we name our method SVTD, ***S**ingular **V**alue based **T**ensor **D**ecomposition*. The key steps of our algorithm (SVTD) are outlined in Algorithm (1).

Algorithm 1 SVTD

Require: M_1, M_2, M_3, and the number of latent states $k \leq n$
1: Decompose M_2 as $M_2 = U_k S_k U_k^\top$ with a SVD truncated at the kth singular vector.
2: Define the whitening matrix $E = U_k S_k^{1/2}$ and calculate its pseudoinverse $E^\dagger = (S_k)^{-1/2} U_k^\top$.
3: Select a feature r and compute $M_{3,r}$
4: Compute O as the left singular vectors of $H_r := E^\dagger M_{3,r} {E^\dagger}^\top$
and m_r as the singular values.
5: **for** $i = 1 \to n$ **do**
6: Compute $H_i := E^\dagger M_{3,i} {E^\dagger}^\top$, where $E^\dagger = (S_k)^{-1/2} U_k^\top$ is the Moore-Penrose pseudo-inverse of E
7: Obtain the ith row of M as the diagonal entries of $O^\top H_i O$
8: **end for**
9: Obtain ω solving $M_1 = M\omega$
10: Return (M, ω)

The constructive proof of the following theorem will explain why SVTD performs a correct retrieval of the desired model parameters.

Theorem 4 Let r be the feature selected at Step 3 of SVTD, and assume $k \leq n$. If all the elements of m_r are distinct and M_2 and M_3 have rank k, then SVTD produces the values of (M, ω) exactly.

Proof As a first step we perform a SVD to $M_2 = U_k S_k U_k^\top$, where $U_k \in \mathbb{R}^{n\times k}$ and $S_k \in \mathbb{R}^{k\times k}$ are the matrices of the singular vectors and values truncated at the kth greatest singular value. Then we define the whitening matrix $E = U_k S_k^{1/2} \in \mathbb{R}^{n\times k}$, and for a slice $M_{3,r}$ of M_3, we define $H_r \in \mathbb{R}^{k\times k}$ as

$$H_r = E^\dagger M_{3,r} (E^\top)^\dagger,$$

where $E^\dagger = (S_k)^{-1/2} U_k^\top$ is the Moore-Penrose pseudo-inverse of E. Now, observe that there exists a unique orthogonal $k \times k$ matrix O, such that

$$M\Omega^{1/2} = EO. \tag{8}$$

To see that such a matrix exists, it is enough to observe that the matrix $O = E^\dagger M\Omega^{1/2}$ is orthogonal (which can be seen from equation $M\Omega M^\top = EE^\top$) and fulfills the requested relation. To see that it is unique, assume that that O_1 and O_2 are two orthogonal matrices such that $M\Omega^{1/2} = EO_1 = EO_2$. Then, multiplying by $E^\dagger$ on the left, we obtain $O_1 = E^\dagger EO_1 = E^\dagger EO_2 = O_2$.
Using Equation (8), one gets the following characterization of $M_{3,r}$:

$$M_{3,r} = M\Omega^{1/2} diag(m_r)\Omega^{1/2} M^\top = EOdiag(m_r)O^\top E^\top,$$

from which it follows that

$$H_r = Odiag(m_r)O^\top.$$

Now, one gets the rth row of M as the *singular values* of H_r. Repeating these steps for all the $i \in \{1, ..., n\}$ will provide the full matrix M. In order to avoid ordering issues with the columns of the retrieved M, one can use the same matrix O to diagonalize all the matrices H_i, as O is uniquely determined, because the elements of m_r are distinct. So, compute O as the singular vectors of H_r, for a certain feature r, and re-use it for all the other features. The subsequent estimations of ω is straightforward, and can be obtained by solving the linear system $M_1 = M\omega$. □

Remark 6 (On the generality of the algorithm) Like other methods of moments, our algorithm requires the number of topics to be smaller than the vocabulary size, $k \leq n$, which is in general a realistic requirement: in topic modeling applications one usually has thousands of words and just few tens of topics. Also, we remark that during the construction of the algorithm we have not made any hypotheses on the probability distribution of the data; instead, we have required the matrix M to be full rank and, in addition, to have at least one feature r with different conditional expectations on the various topics. Also, we do not need to know in advance what this feature is, as Remark 7 will explain. This last requirement is not present in the other matrix-based methods, as they rely on a randomized matrix to guarantee the uniqueness of the results (see Section 3.1); but the reconstruction accuracy of these methods is significantly worse than that of SVTD, as we will show experimentally. Finding a deterministic method that joins the scalability properties of simultaneous diagonalization methods without requiring this separation condition is an interesting open problem.

Remark 7 (On the selection of feature r) The initial steps of the algorithm require the isolation of a feature r to compute the matrix O. While theoretically we could select any feature r such that all elements of $m_r = (\mu_{r,1}, ..., \mu_{r,k})$ are distinct, it is clear that, if matrices M_1, M_2 and M_3 are subject to perturbations, the results obtained by the algorithm might vary, depending on the selected feature r. However, known results from matrix perturbation theory (see for example Stewart and Sun 1990), indicate that the matrix O (which contains the singular vectors of H_r) is more sensitive to random perturbations when the minimum difference between the entries of $m_r = (\mu_{r,1}, ..., \mu_{r,k})$ is small. This fact suggests that a possible heuristic to choose the feature r (and hence a reliable matrix O), is to repeat the steps 3 and 4 of the algorithm, isolating different features and selecting the one that maximizes the quantity $\min_{i \neq j}(|\mu_{r,i} - \mu_{r,j}|)$. With this heuristic, a user can run SVTD without any previous knowledge on the feature to extract. Its cost is that of performing n times a $k \times k$ SVD, therefore $O(nk^3)$, and dominated by the costs of other parts of the algorithm as discussed next.

Remark 8 (Complexity analysis) We start analyzing the time complexity. Using randomized SVD techniques (see Halko et al., 2011), step 1 can be carried out with a total of $O(n^2k)$ steps, the SVD on $E^\dagger M_{3,r}(E^\top)^\dagger$ requires $O(k^3)$ steps while step 6 requires $O(n^2k)$ steps for matrix multiplication for each one of the n iterations. So the overall complexity of the Algorithm 1 is

$$O(n^2k + k^3 + n^3k).$$

We need to add the additional cost of the feature-selection method outlined in Remark 7, whose cost is $O(nk^3)$. Since $n \geq k$, this cost is dominated by the larger $O(n^3k)$ component.

It is important to highlight that the general implementation given in Algorithm 1 has mainly a descriptive purpose. For a specific LVM, optimized implementations may exists. For the single-topic model, for example, the method can be implemented without ever computing explicitly the tensor M_3, instead computing in one step its whitened slices H_r, with a complexity of $O(Nnk)$, and then performing the subsequent diagonalizations in $O(n^2k^2)$ time. Computation can be further accelerated by exploiting the sparsity of the data, that is, using sparse matrix arithmetic techniques. We also remark that the algorithm is trivially parallelizable: Assuming that we have m machines on which to parallelize steps 5, 6, 7 of the algorithm and the feature selection task, we can reduce the total running time to $O(n^2k + k^3 + n^3k/m)$.

Regarding memory, notice that we never use the full tensor M_3, but only its rth slice, for the selected feature r. This means that only that slice has to be computed and stored, and the memory complexity of the algorithm is $O(n^2)$.

These complexity requirements are comparable to those of the methods from Anandkumar et al. (2012a,b); however, those methods are randomized, with nontrivial variance in their output, so they may require several runs of the full algorithm in order to provide accurate results. The tensor power method from Anandkumar et al. (2014) has in general higher cost. It is an iterative technique, with a number of iterations difficult to bound a priori; the authors suggest that accuracy ϵ can be reached with $O(k^{5+\delta}(\log(k) + \log\log(1/\epsilon)))$ operations, among iterations, random restarts and actual matrix operations, higher than our $O(k^3)$. To this time we need to add the time to get the $k \times k \times k$ tensor from the sample, which is not trivial for many LVMs.

3.1 Alternative Algorithms

Mathematically speaking, SVTD is a method to solve the system of equations (3),(4) and (5) and find the unknown model parameters (M, ω). In theory, as Kruskal's criterion guarantees that M_3 has a unique CP decomposition, one could simply try to directly decompose the third order moment, using for example one of the general-purpose tensor decomposition methods described by Tomasi and Bro (2006); Kolda and Bader (2009); Sidiropoulos et al. (2017). However this approach is impractical as M_3 may be too large (for instance in the topic modeling example, where n is quite high). Conversely, explicitly relying on the full set of equations (3),(4) and (5) allows to find the pair (M, ω) that exactly solves those equations — a guarantee that may not be present in general-purpose tensor decomposition algorithms — while working with small, low dimensional tensors/matrices. For example, the whitening step that SVTD performs at step 4 of algorithm 1, reduces the slices of M_3 to a small $k \times k$ matrix, which dramatically reduces the complexity of the algorithm. Methods of moments in the literature other than SVTD all exploit the specific structure of Equations (3),(4) and (5) to efficiently find an optimal solution. In the rest of this section we briefly present their characteristics — providing additional details with respect to what already mentioned in the introduction — and compare them with SVTD.

The tensor power method (TPM) is described by Anandkumar et al. (2014). In that algorithm, the idea is to find a matrix W (for example, the pseudoinverse of the whitening matrix E) such that $W^\top M_2 W = I$, and use it to whiten the tensor M_3, getting a $k \times k \times k$ tensor T, from whose robust eigenvectors it is possible to retrieve the model parameters. To get the set of the robust eigenvectors, TPM uses a three-dimensional extension of the well-known matrix power method. While very robust, the implementation of this method may be complex for someone who is not familiar with tensors. In addition, it is an iterative method, and so it requires a tuning of the hyperconvergence parameters, which might require many trial-and-error tests. These practical considerations, together with its higher time and memory complexity outlined in Remark 8, are drawbacks compared to matrix methods.

Matrix methods (Anandkumar et al., 2012a,b), are technically more similar to the method presented here, and they are two variations of the same approach, one (Anandkumar et al., 2012b) using eigenvectors, and the other (Anandkumar et al., 2012a) using singular vectors. These methods take a random vector $\eta \in \mathbb{R}^n$, and observe that the matrix $M_3(\eta)$, defined as

$$(M_3(\eta))_{i,j} := \sum_{l=1}^{n} ((M_3)_{i,j,l}\eta_l)$$

can be decomposed as follows:

$$M_3(\eta) = M\Omega^{1/2} diag((\eta\mu_1^\top, ..., \eta\mu_k^\top))\Omega^{1/2}M^\top.$$

Then, they compute the whitening matrix E and get the matrix

$$H_\eta = E^\dagger M_3(\eta)(E^\dagger)^\top$$

from whose left singular vectors O they retrieve M (up to rescaling and columns reordering) solving the following system of equations:

$$\begin{cases} M\Omega^{1/2} = EO \\ M\omega = M_1 \end{cases}$$

using essentially Equation (8). The introduction of the random vector η has the scope of guaranteeing that the elements of $(\eta\mu_1^\top, ..., \eta\mu_k^\top)$ are almost surely distinct, and so O is unique. So, we can see that there are essentially two main differences: the first is the fact that instead of using a randomized matrix, we fix a specific feature r, choosing the one with the maximum minimum variation between the feature components; in particular, this is the same of saying that we fix η to be the rth coordinate vector, providing a recipe for finding r in Remark 7. In this way, we provide the choice that maximizes stability. The second difference is the fact that we do not retrieve the matrix M from Equation (8), but observing that, if η_i is the ith coordinate vector, then

$$M_3(\eta_i) = M\Omega^{1/2} diag(m_i)\Omega^{1/2}M^\top$$

and so, for each $i = 1, ..., n$, we can find the row m_i of M as the singular values of the various $H_i = E^\dagger M_3(\eta_i)(E^\dagger)^\top$. In this sense, our method relies on the singular *values* of a SVD decomposition and not on the singular vectors. This may explain why, in our experiments (Section 4), we find this approach much more stable than other matrix methods.

4 Experiments

In this section we test the algorithms presented in this paper on synthetic and real-world data.

Experimental setting: All the experiments in this section have been run on a MacBook Pro with a 2.7 GHz Intel Core i5 processor and 8 GB of RAM memory. All the algorithms have been implemented in Python 2.7 (interpreted, not compiled), using the *numpy* (Walt et al., 2011) library for all linear algebra operations, including Singular Value Decomposition, for which we used numpy's non-randomized SVD.[2] SVTD and the methods from Anandkumar et al. (2012a,b, 2014), used in Section 4.2, have been implemented by the authors. For ALS, which is used in Section 4.2 as well, we have used the implementation provided by *scikit-tensor*.[3] The implementation of LDA based on MCMC that we use in Section 4.3 is based on the open-source *LDA* python package.[4] All the implementations written by the authors, as well as the code of the implementations of the competing methods, have been publicly disclosed and are freely accessible.[5]

4.1 Recovering M_2 and M_3

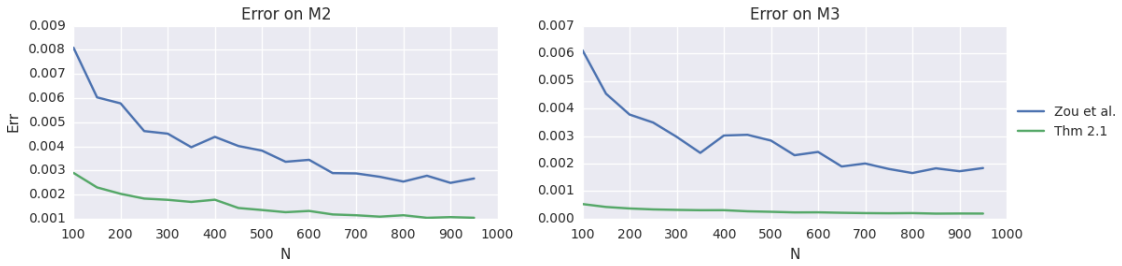


Fig. 1: The $x-$axis of the figures represents the sizes of the synthetic text corpora, while the $y-$axis is Err_2 for the left chart and Err_3 for the right chart. Green lines represent the errors obtained with the method presented in Theorem 1, while blue lines represent the errors obtained with the method by Zou et al. (2013) on the same corpora.

.

In Section 2.1 we described a new estimator to recover the matrix M_2 and tensor M_3 from a sample, comparing it with the methods presented in the state of the art literature from Zou et al. (2013), outlined in Remark 3. In this section we compare, using synthetically generated data, how well the two different methods recover M_2 and M_3 as a function of the sample size. To perform this experiment, we generated a set of 1000 synthetic corpora according to the single-topic model described in Section 2.1, with different sizes (the number of texts for each corpus); the smallest corpus contained 100 texts, the largest 10000; each text contained a random number of words, from a minimum of 3 to a maximum of 100. For each corpus, the values of the unknowns (M, ω) have been randomly generated by sampling a uniform distribution and normalizing both the columns of M and the vector ω to have sum 1. From them we have been able to obtain the theoretical values of M_2 and M_3 using equations (4) and (5) and to compare those values with the one empirically estimated from data using the equations in Theorem 1 for the presented method and the method from Zou et al. (2013) for the competing one. Results appear in Figure 1, where we show how the estimated M_2 and M_3, namely $\tilde{M}_2$ and $\tilde{M}_3$, approach the theoretical values; in particular, the chart presents the errors

$$Err_2 = ||\tilde{M}_2 - M_2||_F \quad \text{and} \quad Err_3 = ||\tilde{M}_3 - M_3||_F$$

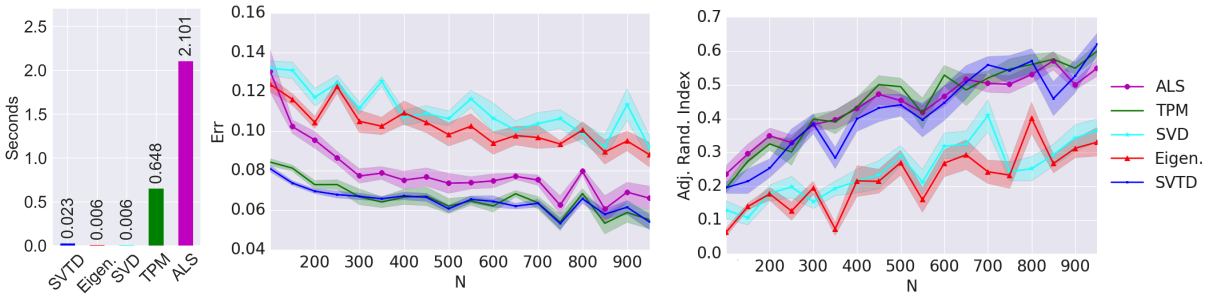
as a function of the sample size N used to find $\tilde{M}_2$ and $\tilde{M}_3$. We can see that the method in Theorem 1 outperforms the state-of-the-art technique; this is because it gives more weight to longer documents, where the signal is stronger, and less to shorter ones, where the signal is noisier.

[2] Link to the code: `https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html`
[3] Link to the code: `https://github.com/mnick/scikit-tensor`
[4] Link to the code: `https://github.com/lda-project/lda`
[5] Link: `https://github.com/mruffini/SpectralMethod.git`

4.2 Recovering the Parameters from a Random Sample.

We now test SVTD on the task of decomposing a tensor of moments obtained from a random sample, comparing its performance with that of existing methods. We fix a dictionary of $n = 100$ words with $k = 5$, and, for several sample sizes comprised between $N = 50$ and $N = 1000$, we generate a random sample X distributed as a single-topic model with parameters[6] (M, ω). For each synthetic sample, we proceed as follows:

- We estimate the values of $\tilde{M}_1, \tilde{M}_2$ and $\tilde{M}_3$ using Theorem 1.
- We retrieve from the estimated $\tilde{M}_1, \tilde{M}_2$ and $\tilde{M}_3$ the pair of unknowns $(\tilde{M}, \tilde{\omega})$ using SVTD as in Algorithm 1. We also generate alternative solutions using the decomposition algorithms from Anandkumar et al. (2014) ("Tensor power method", with 25 random restarts and 20 iterations per restart), from Anandkumar et al. (2012b) ("Eigendecomposition method"), from Anandkumar et al. (2012a) ("SVD method"), and with ALS as described by Kolda and Bader (2009) — where ALS is used to directly find the CP decomposition of $\tilde{M}_3$, with a random initialization and stopping after 250 iterations.
- Each time we generate a solution, we register the time in seconds employed by the various algorithms. For each method, we represent the average time used to recover the parameters over the various runs in Figure 2a.
- For each set of retrieved parameters $(\tilde{M}, \tilde{\omega})$ we calculate the learning error as: $Err = \|\tilde{M} - M\|_F$, where M is the matrix used in the simulations. For each sample size, we repeat the experiment 10 times (each time with newly generated parameters), and we plot in Figure 2b the results in function of N.
- For each run of the experiment and for each of the methods, we use the retrieved parameters to cluster the various documents of the considered corpus, using the MAP assignment described in Remark 1. We compare the clustering accuracy of the various methods using the Adjusted Rand Index (Hubert and Arabie, 1985) of the partition of the data obtained with MAP assignment w.r.t. the one obtained using the true topics. For each sample size, we repeat the experiment 10 times, and we plot in Figure 2c the results of the analysis as a function of N.



(a) (b) (c)

Fig. 2: In Figure 2a the average running times are represented. Figures 2b and 2c contain the analysis of the learning accuracy; in both cases the $x-$axis represents the size of the synthetic text corpora. In 2b the $y-$axis is Err, the reconstruction error for the various tested methods, while in Figure 2c the $y-$axis is the clustering accuracy. The shaded areas represent the variance of the output of the experiments over 10 runs with same sample size.

In Figure 2a, the average running times of the various methods are presented. On average, SVTD is 100 times faster than ALS, 30 times faster than TPM — as a consequence of the better dependence on the number of latent states — and it is slightly slower than other matrix based methods, due to the feature selection process outlined in Remark 7. With respect to accuracy, Figures 2b and 2c show that SVTD and TPM are quite comparable among themselves, being the top performers in terms of learning and clustering accuracy.

We conclude that SVTD provides learning accuracy similar to TPM while having a running time in the order of the more inaccurate matrix-based methods. Also, it is deterministic and requires little

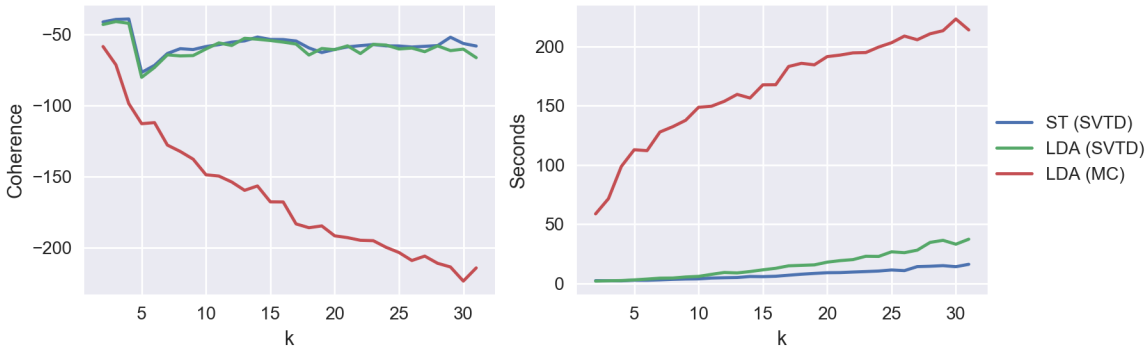
[6] For each sample size, the parameters (M, ω) are randomly generated following the same procedure of the previous experiment.

hyperparameter tuning. ALS has slightly worse performance in terms of learning accuracy and is far slower than the other methods tested.

4.3 Real Data

We now test SVTD on two real-world text corpora: Dante's "*Divina Commedia*" and the *State of the Union* addresses from 1945 to 2005. In both cases, we use SVTD to learn a single-topic model and an LDA, and compare quantitatively and qualitatively their behavior. Additionally, we compare these approaches with a standard method to learn LDA based on Markov chain Monte Carlo (MCMC) approach, described by Griffiths and Steyvers (2004).

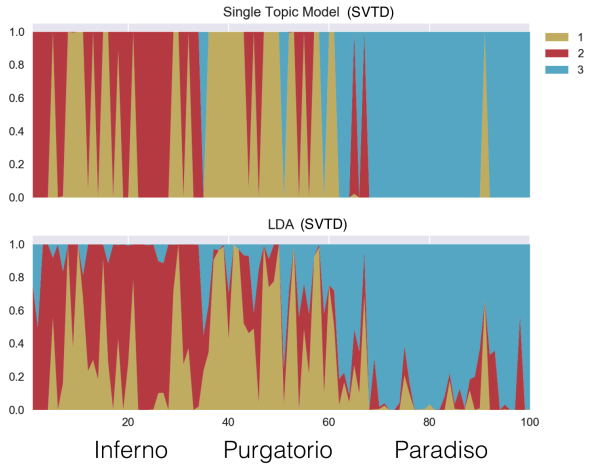
4.3.1 Dante's Divina Commedia



(a)

single-topic model (SVTD)
Topic 1: disse, saprai, morta, torre, maestro *(Transl.) said, will know, dead, tower, master*
Topic 2: vidi, serpente, greve, maestro, sovra *(Transl.) I saw, snake, heavy, master, above*
Topic 3: luce, Cristo, creata, lume, vicine *(Transl.) light, Christ, created, light, near*
LDA (SVTD)
Topic 1: saprai, morta, torre, saggio, disse *(Transl.) will know, dead, tower, wise, said*
Topic 2: vidi, cotai, greve, serpente, maestro *(Transl.) I saw, so many, heavy, snake, master*
Topic 3: luce, creata, vicine, Cristo, lume *(Transl.) light, created, near, Christ, light*

(b)



(c)

Fig. 3: Figure 3a contains the topic coherence (left) of a single-topic model learned by SVTD (ST SVTD), of an LDA learned by SVTD (LDA SVTD) and an LDA learned with MCMC (LDA MC), in function of the number of topics k; top-right figure contains an analysis of the running times. Table 3b contains the most relevant words (together with their English translation) of each topic learned by SVTD setting $k = 3$, both for the single-topic model and for LDA, while Figure 3c, contains the topic assignment for the same two models.

Dante's "Divina Commedia" (Alighieri, 1979) is an Italian epic poem written in the first half of the 14th century.[7] It narrates the imaginary trip of the main character, Dante himself, in the afterlife, guided by Virgilio, the famous Latin poet, and Beatrice, a Florentine woman that inspired most of Dante's works. The storyline represents an allegorical description of death soul's journey towards God according to medieval world view. It begins with Dante's travel trough the "Inferno" (Hell), where damned souls are deemed to eternal punishment according to their sins; the journey then moves to "Purgatorio", a seven-level mountain, where, at each level, a capital sin (sins less serious than those punished in Hell) is allegorically described; here souls are discounting their punishment, before finally moving to "Paradiso", Heaven, which is visited by Dante in the last third of the book. The book is divided into 100 chapters: 34 for Hell, 33 for Purgatory and 33 for Heaven.

We analyze the performance of SVTD on this dataset, using both the single-topic model and LDA (setting $\alpha_0 = 0.2$) and fixing the vocabulary size to $n = 1820$ words, which corresponds to the 70% most frequent ones.[8] Also, we experimentally compare the results of SVTD with a classic approach to learn LDA described by Griffiths and Steyvers (2004) based on MCMC — setting as a stopping criterion 2000 iterations of the sampler (which is the default value provided by the Python package we used). As a first step, we study how the quality of the results depends on the number of topics k requested to the algorithm. To do this, we use the various competing methods to learn a set of model parameters (M, ω), for all the values of k between 2 and 32. For each returned pair of parameters, we evaluate the running time of the algorithm and the average topic coherence across the various topics, displaying the results in Figure 3a ("ST SVTD" and "LDA SVTD" represents the results of the single-topic model and LDA learned with SVTD and "LDA MC" represents the results of LDA learned with MCMC). The *coherence* (Mimno et al., 2011) of a topic in a corpus is a quantitative indicator of the quality of a topic, and evaluates how much pairs of words that are highly probable in a topic tend to co-occur in the texts of the corpus; a good model is expected to generate topics with high coherence scores. Formally, the coherence of a topic μ in a corpus is defined as

$$Coherence(\mu) = \sum_{j=2}^{L} \sum_{i=1}^{j-1} \log \frac{D(w_i, w_j) + 1}{D(w_i)}$$

where $(w_1, ..., w_L)$ is the list of the $L = 20$ most frequent words in topic μ, $D(w_i)$ (resp. $D(w_i, w_j)$) is the count of documents containing the word w_i (resp. w_i and w_j). In Figure 3a, we can see that SVTD outperforms MCMC in terms of speed and coherence. Furthermore, while the performance of SVTD remains good as model grows, the coherence of MCMC degrades strongly.[9] For SVTD, the single-topic model and LDA have similar performance in terms of coherence (left chart), while the running times of LDA are larger (right chart). From the coherence chart, we can see that $k \in \{2, 3, 4\}$ are all good numbers of topics for SVTD; so, we keep $k = 3$ and perform two additional analysis, focusing only on the models learned with SVTD. First, we plot in Figure 3c the topic assignment for the various texts of the corpus, with the single-topic model (top figure) using Equation (1) for the assignment, and for LDA (bottom figure), using the approach described in Remark 4. The $x-$axis represents the chapter of the book: The first 34 are Hell, 35 to 67 are Purgatory, and the last 33 are Heaven, while the areas represent how much each chapter belongs to each of the topics. It is interesting to observe that, both with the LDA and single-topic models, each of the three topics is clearly dominant in one of the three areas of the corpus (Hell, Purgatory and Heaven): topic 1, is clearly dominant in Purgatory's chapters, topic 2 in Hell, while topic 3 dominates Heaven. To see that this makes sense, we print in Table 3b the most relevant words for each topic, for the two models, where the *relevance* (Sievert and Shirley, 2014) is an indicator of how much a word characterizes a topic. Formally, the relevance of a word w with respect to a topic μ and is defined as

$$r(w, \mu) = \lambda \log \mathbb{P}(w|\mu) + (1 - \lambda) \log \frac{\mathbb{P}(w|\mu)}{\mathbb{P}(w)} \quad ,$$

where $\mathbb{P}(w|\mu)$ (resp. $\mathbb{P}(w)$) is the probability of sampling w with topic μ (resp. in the overall corpus) and the weight parameter was set to $\lambda = 0.7$. Again, the topics are similar between the LDA and single-topic models, and are coherent with the topic assignment results. For example, the most relevant words for topic 3 are *Cristo, luce, lume*, meaning Christ and light. The similarity of the topics between the LDA and single-topic

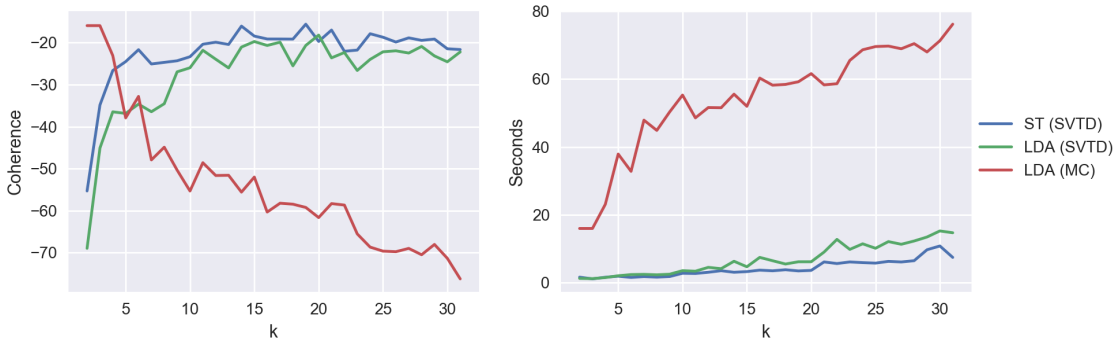
[7] The full text can be found here: `http://www.gutenberg.org/files/1012/1012-0.txt`

[8] We tested the same experiment with different vocabulary sizes — like 80% and 90% of the most frequent words — obtaining with SVTD nearly identical topics.

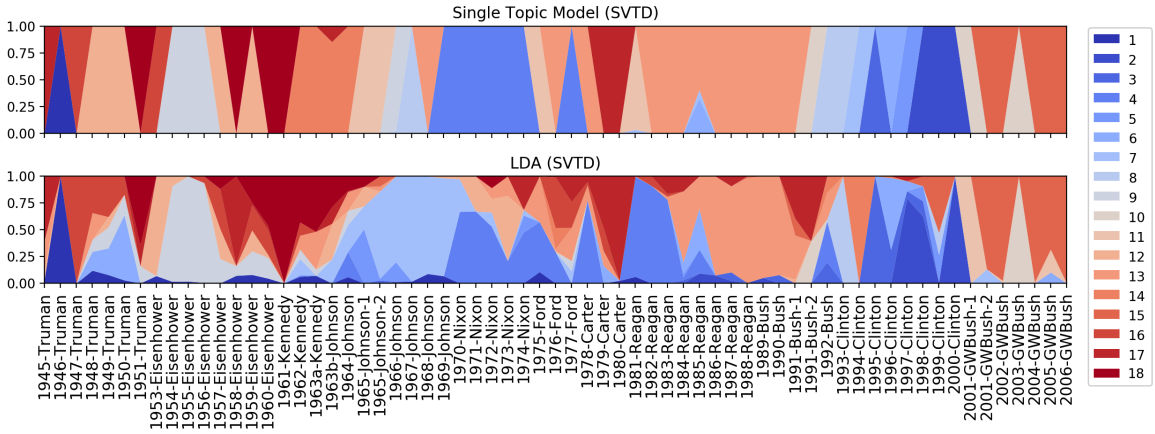
[9] We tried to improve the quality of the topics obtained with MCMC by allowing for further iterations, but the increased running time yielded no improvements in the coherence results.

models is expected from the analysis of the coherence, where the LDA and single-topic models gave very similar scores. An interpretation of this is the fact that Divina Commedia is close to following a single-topic model, with each chapter dealing with a single topic.

4.3.2 State of the Union Addresses



(a)



(b)

Fig. 4: Figure 4a contains the topic coherence (left) of a single-topic model learned by SVTD (ST SVTD), of an LDA learned by SVTD (LDA SVTD) and an LDA learned by MCMC (LDA MC), in function of the number of topics k; top-right figure contains an analysis of the running times. Figure 4b, contains the topic assignment for the two models learned by SVTD setting $k = 18$.

Every year, the president of United States of America presents a speech to a joint session of the United States Congress, where he outlines his governative agenda, the national priorities and legislative projects. We considered the set of $N = 65$ state of the union addresses presented between 1945 and 2005, and we perform the same analysis performed for the Divina Commedia corpus. Again, we keep the 70% most frequent words, obtaining a total vocabulary of $n = 1184$ words, and analyze topic coherence and running times in Figure 4a for SVTD — using LDA ("LDA SVTD") and single-topic model ("ST SVTD") — and MCMC — only for LDA ("LDA MC"). Again, the performance of MCMC degrades as the model increases, while SVTD, both when learning LDA and single-topic models, keeps providing meaningful results, always using smaller running times. From now on we will only focus on the methods leaned by SVTD.

Overall, the single-topic model presents a better coherence and smaller running times than LDA. The coherence score for the single-topic model is maximized when we set $k = 18$, which is also good for LDA. We thus fix $k = 18$ and present in Figure 4b the topic assignment for the two models, where the areas represent

how much each document deals with each topic. Also, Table 1 presents the most relevant words for each topic according to the LDA and single-topic models. The topics make sense and look similar across the two models. However, unlike the Divina Commedia case, they are not identical. For example, topic 8 for the single-topic model looks like a mixture of topic 7 and 8 in LDA. Looking at the topic assignments in Figure 4b, we can see that speeches from the same president share similar topic assignments: for example, topic 15 is dominant for G.W. Bush, and mainly deals with terrorism and Iraq. Cold-war presidents have a strong predominance of topics involving the Soviet Union, space missions and the Vietnam war. The two models are often coherent, with some notable exceptions, like president Kennedy: LDA assigns him to a mixture of topics that properly describe the challenges of cold war (LDA topics: 12, 13, 17, 18), while the single-topic model provides a simpler characterization with speeches assigned to topics 18 (again a cold war topic) and 14 (cold war, with a focus on Europe and foreign politics).

single-topic model (SVTD)	
Topic 1:	dollars, 1947, 1945, estimated, reconversion
Topic 2:	21st, century, affordable, children, Medicare
Topic 3:	class, work, cold, people, worked, working
Topic 4:	years, people, energy, elected, congress, peace
Topic 5:	21st, challenge, children, century, parents
Topic 6:	challenge, children, working, challenges, work
Topic 7:	produced, care, health, kids, people
Topic 8:	Vietnam, plan, recommend, deficit, numbers
Topic 9:	highway, Vietnam, recommend, program, federal
Topic 10:	Hussein, Saddam, intelligence, aids, weapons
Topic 11:	applause, program, government, federal, people
Topic 12:	benefits, democratic, economic, great, life
Topic 13:	federal, government, programs, Hussein, intelligence
Topic 14:	alliance, Atlantic, people, free, Europe
Topic 15:	applause, terrorists, Iraq, Iraqi, terror
Topic 16:	strikes, bargaining, collective, labor, management
Topic 17:	space, soviet, disarmament, military, defense
Topic 18:	disarmament, space, defense, soviets, military

LDA (SVTD)	
Topic 1:	dollars, 1947, 1945, estimated, reconversion
Topic 2:	21st, college, affordable, children, child
Topic 3:	class, cold, worked, cuts, talk
Topic 4:	regulations, plan, government, reducing, inflation
Topic 5:	Vietnam, south, 21st, tonight, principle
Topic 6:	challenge, children, working, work, challenges
Topic 7:	Vietnam, south, tonight, north, conflict
Topic 8:	companies, plan, deficit, invest, care
Topic 9:	highway, maintenance, postal, planning, program
Topic 10:	Hussein, Saddam, seniors, aids, intelligence
Topic 11:	applause, Medicare, Hussein, seniors, Saddam
Topic 12:	controls, demands, labor, study, initiative
Topic 13:	percent, family, people, America, tonight
Topic 14:	produced, care, kids, health, renew
Topic 15:	applause, Iraq, terrorists, terrorist, seniors
Topic 16:	collective, strikes, bargaining, management, labor
Topic 17:	soviet, soviets, military, peace, disarmament
Topic 18:	space, disarmament, civil, defense, Latin

Table 1: The most relevant words for each topic learned by SVTD setting $k = 18$, both for the single-topic model and for LDA.

5 Conclusion and Future Work

We described a simple algorithm to learn latent variable models in polynomial time, together with novel estimators of the moments for the single-topic model and LDA. The proposed estimators are unbiased and outperform those coming from previous literature. The proposed tensor decomposition technique proved to be a top performer in terms of speed and accuracy of the retrieved results. Overall, the methods proposed in this paper provide a robust technique to learn high-dimensional latent variable models with very high efficiency.

The main limitation of our approach —which in fact is a limitation of all methods of moments— is that Theorems 1 and 3 only hold when data is exactly generated by the considered LVM (respectively a single-topic model and LDA); however, real-world datasets are likely to violate model assumptions, as they never exactly follow a specific latent variable model. When data presents outliers, or violates model assumptions, it is not clear what a method of moments will yield. A reasonable theoretical continuation of this work consists in investigating the behavior of our method in this challenging setting, analyzing its robustness to outliers and model misspecification. Experimental evidences confirm that SVTD is able to provide competitive performance on real data, which may indicate its ability to deal with data that violate model hypotheses; however a deeper theoretical study is required to better explore this possibility.

We also aim at providing a perturbation theorem — for which we already have preliminary results — studying the sensitiveness of SVTD to random perturbations.

On the applications side, we are interested in applying this algorithm to learn LVM in the healthcare analytics field, for instance to learn mixture models to perform patient clustering (Ruffini et al., 2017) and construct disease progression models.

Acknowledgements

M. Casanellas is is partially funded by AGAUR project 2017 SGR-932 and MINECO/FEDER project MTM2015-69135-P. R. Gavaldà is partially funded by AGAUR project 2014 SGR-890 (MACDA) and by MINECO projects TIN2014-57226-P (APCOM) and TIN2017-89244-R (MACDA). Both authors are partially funded by MDM-2014-0445.

Appendix

A Proofs for Section 2

A.1 Proof of Theorem 1

Proof We will prove the statements only for

$$\mathbb{E}\left(\frac{\sum_{i=1}^{N}(X^{(i)})_h(X^{(i)})_l}{\sum_{i=1}^{N}(c_i-1)c_i}\right)=\sum_{j=1}^{k}\omega_j\mu_{h,j}\mu_{l,j}.$$

Similar arguments hold for the other equations. It is easy to see, by conditional independence, that

$$E((X^{(i)})_h(X^{(i)})_l)=\sum_{j=1}^{k}\omega_j E((X^{(i)})_h(X^{(i)})_l|Y=j)$$

but the conditioned $(X^{(i)})_h$ and $(X^{(i)})_l$ are components of a multinomial distribution and so

$$\sum_{j=1}^{k}\omega_j E((X^{(i)})_h(X^{(i)})_l|Y=j)=\sum_{j=1}^{k}\omega_j(c_i^2-c_i)\mu_{h,j}\mu_{l,j},$$

which implies the thesis. □

A.2 Proof of Theorem 2

Proof We want to express the elements of the matrix $\tilde{M_2}-M_2$ in an appropriate way and then express a bound using McDiarmid's inequality (McDiarmid 1989 and Lemma 1). We know by construction that, for any $i\in\{1,...,N\}$ it holds that

$$X^{(i)}=\sum_{j=1}^{c_i}x_j^{(i)},$$

where each $x_j^{(i)}$ is the jth word of the document. We thus consider the set of all the words from all the documents:

$$\mathcal{X}=(x_1^{(1)},...,x_{c_1}^{(1)},...,x_1^{(N)},...,x_{c_N}^{(N)}).$$

It is easy to see that $\tilde{M_2}$ can be expressed as a function of $\mathcal{X}$, for all pairs $u,v\in\{1,...,n\}$ we have

$$(\tilde{M_2})_{u,v}(\mathcal{X})=\frac{\sum_{i\neq j}(x_i^{(1)})_u(x_j^{(1)})_v+...+\sum_{i\neq j}(x_i^{(N)})_u(x_j^{(N)})_v}{C_2},$$

where $C_2=\sum_{i=1}^{N}c_i(c_i-1)$. We now define the following function:

$$\Phi(\mathcal{X})=||\tilde{M_2}(\mathcal{X})-M_2||_F$$

and observe that, given two samples differing only by one element

$$\mathcal{X}=(x_1^{(1)},...,x_{c_1}^{(1)},...,x_i^{(l)},...,x_1^{(N)},...,x_{c_N}^{(N)})$$

$$\mathcal{X}'=(x_1^{(1)},...,x_{c_1}^{(1)},...,x_i^{(l)'},...,x_1^{(N)},...,x_{c_N}^{(N)})$$

we obtain the following inequality:

$$|\Phi(\mathcal{X})-\Phi(\mathcal{X}')|\leq||\tilde{M_2}(\mathcal{X})-\tilde{M_2}(\mathcal{X}')||_F=$$

$$=\sqrt{\sum_{u,v=1}^{n}\left(\frac{\sum_{i\neq j}(x_j^{(l)})_u((x_i^{(l)})_v-(x_i^{(l)'})_v)}{C_2}\right)^2}\leq\frac{\sqrt{2}\max_j(c_j)}{C_2}.$$

The inequality above enables us to apply McDiarmid's inequality, stating that

$$\mathbb{P}(||\tilde{M_2} - M_2||_F > \mathbb{E}(||\tilde{M_2} - M_2||_F) + \epsilon) \leq e^{-\frac{\epsilon^2 C_2^2}{(\max_j (c_j))^2 C_1}} = e^{-t^2},$$

where we defined

$$t = \frac{\epsilon C_2}{\max_j(c_j)\sqrt{C_1}}.$$

We now provide a bound for $\mathbb{E}(||\tilde{M_2} - M_2||_F)$. We begin observing that $\tilde{M_2} = \sum_{i=1}^N w_i \tilde{M_2}^{(i)}$, where $w_i = \frac{c_i(c_i-1)}{C_2}$ and $\tilde{M_2}^{(i)}$ are independent matrices defined as follows:

$$(\tilde{M_2}^{(i)})_{(u,v)} = \frac{\sum_{l \neq j} (x_l^{(i)})_u (x_j^{(i)})_v}{c_i(c_i-1)}.$$

Notice that, for any i, $\mathbb{E}(\tilde{M_2}^{(i)}) = M_2$. Using Jensen's inequality we have

$$\mathbb{E}(||\tilde{M_2} - M_2||_F) \leq \sqrt{\mathbb{E}(||\tilde{M_2} - M_2||_F^2)}.$$

This last term is equal to

$$\sqrt{\mathbb{E}(||\tilde{M_2}||_F^2) - ||M_2||_F^2} = \sqrt{\sum_{u,v} \mathbb{E}((\sum_{i=1}^N w_i (\tilde{M_2}^{(i)})_{(u,v)})^2) - ||M_2||_F^2} =$$

$$= \sqrt{\sum_{u,v} \mathbb{E}(\sum_{i=1}^N w_i^2 (\tilde{M_2}^{(i)})_{(u,v)}^2) + \sum_{u,v} \mathbb{E}(\sum_{i \neq j} w_j w_i (\tilde{M_2}^{(i)})_{(u,v)} (\tilde{M_2}^{(j)})_{(u,v)}) - ||M_2||_F^2}$$

and using the fact that $\mathbb{E}(\tilde{M_2}^{(i)}_{(u,v)} \tilde{M_2}^{(j)}_{(u,v)}) = (M_2)^2_{(u,v)}$, this equals

$$\sqrt{\sum_{i=1}^N w_i^2 \mathbb{E}(||\tilde{M_2}^{(i)}||_F^2) + \sum_{i \neq j} w_j w_i ||M_2||_F^2 - ||M_2||_F^2}\,.$$

Now using that $||\tilde{M_2}^{(i)}||_F \leq 1$, we can bound this from above by

$$\sqrt{\sum_{i=1}^N w_i^2 + ||M_2||_F^2 (\sum_{i \neq j} w_j w_i - 1)} = \sqrt{\sum_{i=1}^N w_i^2 (1 - ||M_2||_F^2)}.$$

where in the last equality we used the fact that $\sum_{i \neq j} w_j w_i = 1 - \sum_{i=1}^N w_i^2$. So, if we call $W_2^{(N)} = \sum_{i=1}^N w_i^2$, we have $\mathbb{E}(||\tilde{M_2} - M_2||_F) \leq \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)}$, from which we obtain

$$\mathbb{P}(||\tilde{M_2} - M_2||_F > \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + t\frac{\max_j(c_j)\sqrt{C_1}}{C_2}) \leq e^{-t^2}.$$

In conclusion, we can state that if $e^{-t^2} = \delta$ we get, for any $\delta \in (0, 1]$

$$\mathbb{P}(||\tilde{M_2} - M_2||_F > \epsilon) \leq \delta$$

where

$$\epsilon = \sqrt{W_2^{(N)}(1 - ||M_2||_F^2)} + \sqrt{\log(\frac{1}{\delta})}\frac{\max_j(c_j)\sqrt{C_1}}{C_2}.$$

A similar argument works for M_3. □

Lemma 1 (McDiarmid's inequality McDiarmid 1989.) Let $X_1, ..., X_m$ be independent random variables all taking values in the set $\mathcal{C}$. Furthermore, let $f : \mathcal{C}^m \to \mathbb{R}$ be a function of $X_1, ..., X_m$ satisfying for all i and for all $x_1, ..., x_m, x_i' \in \mathcal{C}$,

$$|f(x_1, ..., x_i, ..., x_m) - f(x_1, ..., x_i', ..., x_m)| \leq c_i.$$

Then for all $\epsilon > 0$,

$$\mathbb{P}(f - E[f] \geq \epsilon) \leq exp(-\frac{2\epsilon^2}{\sum_{i=1}^m c_i^2}).$$

References

Alighieri, D. (1979). *La Divina Commedia, a cura di N. Sapegno*. Nuova Italia, Firenze.

Anandkumar, A., Foster, D. P., Hsu, D. J., Kakade, S. M., and Liu, Y.-K. (2012a). A spectral algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 917–925.

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

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012b). A method of moments for mixture models and hidden Markov models. In *Conference on Learning Theory (COLT)*, pages 33.1–33.34.

Appellof, C. J. and Davidson, E. R. (1981). Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056.

Balle, B., Hamilton, W., and Pineau, J. (2014). Methods of moments for learning stochastic languages: unified presentation and empirical comparison. In *International Conference on Machine Learning (ICML)*, pages 1386–1394.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3(Jan):993–1022.

Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319.

Chaganty, A. T. and Liang, P. (2013). Spectral experts for estimating mixtures of linear regressions. In *International Conference on Machine Learning (ICML)*, pages 1040–1048.

Chiantini, L., Ottaviani, G., and Vannieuwenhoven, N. (2017). On generic identifiability of symmetric tensors of subgeneric rank. *Transactions of the American Mathematical Society*, 369(6):4021–4042.

Colombo, N. and Vlassis, N. (2016). Tensor decomposition via joint matrix Schur decomposition. In *International Conference on Machine Learning (ICML)*, pages 2820–2828.

Comon, P., Qi, Y., and Usevich, K. (2017). Identifiability of an x-rank decomposition of polynomial maps. *SIAM Journal on Applied Algebra and Geometry*, 1(1):388–414.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2004). Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition. *SIAM Journal on Matrix Analysis and Applications*, 26(2):295–327.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, 39(1):1–38.

Ge, R., Huang, Q., and Kakade, S. M. (2015). Learning mixtures of Gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM Symposium on Theory Of Computing (STOC)*, pages 761–770.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.

Harshman, R. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189.

Hitchcock, F. L. (1928). Multiple invariants and generalized rank of a p-way matrix or tensor. *Studies in Applied Mathematics*, 7(1-4):39–79.

Hsu, D. and Kakade, S. M. (2013). Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science (ITCS)*, pages 11–20.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Jain, P. and Oh, S. (2014). Learning mixtures of discrete product distributions using spectral decompositions. In *Conference on Learning Theory (COLT)*, pages 824–856.

Kolda, T. G. (2001). Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Kruskal, J. B. (1977). Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95–138.

Kuleshov, V., Chaganty, A., and Liang, P. (2015). Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics (AISTATS)*, pages 507–516.

Leurgans, S., Ross, R., and Abel, R. (1993). A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083.

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. (2011). Optimizing semantic coherence in topic models. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 262–272.

Newman, D., Asuncion, A., Smyth, P., and Welling, M. (2009). Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10(Aug):1801–1828.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110.

Qi, Y., Comon, P., and Lim, L.-H. (2016). Semialgebraic geometry of nonnegative tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 37(4):1556–1580.

Ruffini, M., Gavalda, R., and Limon, E. (2017). Clustering patients with tensor decomposition. In *Machine Learning for Healthcare Conference (MLHC)*, pages 126–146.

Sanchez, E. and Kowalski, B. R. (1990). Tensorial resolution: a direct trilinear decomposition. *Journal of Chemometrics*, 4(1):29–45.

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., and Faloutsos, C. (2017). Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582.

Sievert, C. and Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on Interactive Language Learning, Visualization, and Interfaces (ILLVI)*, pages 63–70.

Song, L., Xing, E. P., and Parikh, A. P. (2011). A spectral algorithm for latent tree graphical models. In *International Conference on Machine Learning (ICML)*, pages 1065–1072.

Stewart, G. and Sun, J.-g. (1990). *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Academic Press.

Tomasi, G. and Bro, R. (2006). A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Van Der Veen, A.-J. and Paulraj, A. (1996). An analytical constant modulus algorithm. *IEEE Transactions on Signal Processing*, 44(5):1136–1155.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Zhang, T. and Golub, G. H. (2001). Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(2):534–550.

Zou, J. Y., Hsu, D. J., Parkes, D. C., and Adams, R. P. (2013). Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2238–2246.