Advances in Tensor Analysis with Applications in Text Mining

Elaheh Sobhani

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE ALPES

Préparée dans le cadre d'une cotutelle entre **l'Université Grenoble Alpes et l'Université de Technologie de Sharif**

Spécialité : **Signal, Image, Parole, Télécommunication (SIPT)**

Arrêté ministériel : le 6 janvier 2005 - 25 mai 2016

Présentée par

Elaheh SOBHANI

Thèse dirigée par **Pierre COMON** et **Christian JUTTEN**

et codirigée par **Massoud BABAIE-ZADEH**

préparée au sein des **Laboratoires Grenoble Images Parole Signal Automatique (GIPSA) et Digital Signal Processing (DSP)**

dans **les École Doctorale d'Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS) et le Département d'Ingénierie Électrique**

Advances in Tensor Analysis with Applications in Text Mining

Thèse soutenue publiquement le **10/03/2022**,

devant le jury composé de :

M. Laurent ALBERA
Professeur, Université de Rennes I, Président

M. Eric MOREAU
Professeur, Université de Toulon Var, Rapporteur

M. Reza SAMENI
Professeur, Université d'Emory, USA, Rapporteur

M. Farrokh MARVASTI
Professeur, Université de Technologie de Sharif, Iran, Examinateur

Mme. Sepideh HAJIPOUR
Maitre de conférences, Université de Technologie de Sharif, Iran, Examinatrice

M. Pierre COMON
Directeur de Recherche, CNRS, Université Grenoble Alpes, Directeur de thèse

M. Christian JUTTEN
Professeur, Université Grenoble Alpes, Co-Directeur de thèse

M. Massoud BABAIE-ZADEH
Professeur, Université de Technologie de Sharif, Iran, Co-Directeur de thèse

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisors, Prof. Pierre Comon, Prof. Christian Jutten and Prof. Massoud Babaie-Zadeh, and for their continuous support of my Ph.D. study and research, for their endless patience, motivation, enthusiasm, and immense knowledge.

The hardworking, smart, cordial Pierre: please accept my heartfelt thanks for your enlightening guidance which gave direction to the work, and your constant support and impeccable expertise during all these years. The large-hearted, caring, welcoming, supportive Christian: I will always remember, and hopefully learn from your kind patronage, not only in my studies, but also in all aspects of my life. The adroit, supportive, expert Massoud: I am honored to work with you for several years of my life, and forever grateful to you for your continued advice and inspiration, and introducing and recommending me to the people of GIPSA-Lab which brought me the opportunity of working there. I could not have ever imagined having better advisors and mentors for my Ph.D. study.

Beside my advisors, I would like to thank the rest of my thesis committee: Prof. Farrokh Marvasti, Prof. Laurent Albera, Prof. Eric Moreau, Dr. Reza Sameni and Dr. Sepideh Hajipour, for their encouragement, insightful comments, deep questions and candid critical insights.

I would also like to thank my always interested, encouraging and enthusiastic friends and colleagues in both laboratories, GIPSA ad DSP.

Last but not least, my sincere gratitude goes to my eternal life-coaches: my parents Mohammadhadi Sobhani and Mahnaz Ghazi, for their love and encouragement, without whom I would never have enjoyed so many oppor-

tunities. Finally, I wish to thank my brother Alireza Sobhani and his nice wife Maryam Fazeli, for all their spiritual support throughout my life and my study.

Abstract

Tensors or multi-way arrays are useful tools to identify unknown quantities thanks to the uniqueness of their decomposition. Tensor decompositions have been widely applied to obtain unknown components with physical meanings in many applications such as medical image and signal processing, hyperspectral images analysis, chemometrics, etc.

In this thesis, we investigate the application of tensor decomposition for probability estimations, which are required for some targeted data/text mining tasks such as unsupervised clustering of data/documents. Besides criticizing the existing tensor decomposition algorithms for probability estimations, we propose to apply some proper constrained tensor decompositions, which result in more reliable and accurate estimations. Moreover, we introduce an algorithm for constrained tensor decomposition, called Simple Forward-Backward Splitting (SFBS), which is based on Proximal Minimization. SFBS performs better than state-of-the-art in decomposing noisy tensors while computationally less expensive.

In addition, to evaluate the performance of tensor decomposition algorithms, we introduce an index that we name CorrIndex, which provides interpretable performance bounds, while keeping computational complexity to a reasonable level. Furthermore, we propose a method of moment estimation (standard averaging), which estimates the second and third order moments, with the same performance of state-of-the-art, but based on a much simpler concept, *i.e.* weighted averaging. Moreover, standard averaging performs better in small dimensions, and provides some advantages in terms of computational complexity.

Keywords— Tensor decomposition, Text mining, Unsupervised clustering, Hidden/latent variable, Third order moments, Forward-Backward Splitting, Proximal operator, Performance index, Permutation and scale ambiguity

Résumé

Les tenseurs - ou les tableaux multi-indices - sont des outils utiles pour identifier des quantités inconnues grâce à l'unicité de leur décomposition. Les décompositions tensorielles ont été largement utilisées pour obtenir des composantes inconnues ayant une signification physique dans de nombreuses applications, telles que le traitement d'images et de signaux médicaux, l'analyse d'images hyperspectrales, la chimiométrie, etc.

Dans cette thèse, nous étudions l'application de la décomposition tensorielle pour l'estimation de probabilités, qui sont nécessaires pour certaines tâches ciblées de fouille de données/textes telles que le groupement non supervisé de données/documents. Au delà de l'analyse critique des algorithmes de décomposition tensorielle utilisés pour l'estimation de probabilités, nous proposons des décompositions tensorielles sous des contraintes appropriées, qui donnent lieu à des estimations plus fiables et plus précises. De plus, nous introduisons un algorithme de décomposition tensorielle sous contraintes, appelé Simple Forward-Backward Splitting (SFBS), qui est basé sur une approche de minimisation proximale. SFBS est plus performant que l'état de l'art dans la décomposition des tenseurs bruités tout en étant moins coûteux en calcul.

En outre, pour évaluer la performance des algorithmes de décomposition tensorielle, nous introduisons un indice appelé CorrIndex, qui fournit des limites de performance interprétables, tout en maintenant la complexité de calcul à un niveau raisonnable. De plus, nous proposons une méthode d'estimation des moments (moyenne standard), qui estime les moments d'ordre deux et trois, avec les mêmes performances que l'état de l'art, mais basée sur un concept beaucoup plus simple, *i.e.* la moyenne pondérée. De plus, le calcul de la moyenne standard est plus performant dans les petites dimensions et présente certains avantages en termes de complexité de calcul.

Mots clés— Décomposition tensorielle, Fouille de texte, Clustering non-supervisé, Variable cachée/latente, Moment de troisième ordre, Fractionnement avant-arrière, Opérateur proximal, Indice de performance, Ambiguïté d'échelle et de permutation

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

ADMM Alternating Direction Method of Multipliers

ALS Alternating Least Squares

AO Alternating Optimization

AO-ADMM Alternating Optimization-Alternating Direction Method of Multipliers

APG Alternating Proximal Gradient

BCD Block Coordinate Descent

BC-VMFB Block Coordinate Variable Metric Forward- Backward

BSS Blind Source Separation

CP Canonical Polyadic or Candecomp/Parafac

DOA Direction Of Arrival

EM Expectation Maximization

FastNTF-APG Fast Non-negative Tensor Factorization-Alternating Proximal Gradient

FISTA Fast Iterative Shrinkage Thresholding Algorithm

HMM Hidden Markov Model

HOSVD Higher Order Singular Value Decomposition

IR Information Retrieval

JAD Joint Approximate Diagonalization

KL Kurdyka-Lojasiewicz

LDA Latent Dirichlet Allocation

LSA Latent Semantic Analysis

lsc lower semi-continuous

LVM Latent Variable Models

ML Maximum Likelihood

MLSVD Multi-Linear Singular Value Decomposition

MRLSA Multi-Relational Latent Semantic Analysis

MWM Maximum Weighted Matching

NLP Natural Language Processing

NMF Non-negative Matrix Factorization

NTF Non-negative Tensor Factorization

PALM Proximal Alternating Linearized Minimization

PILSA Polarity Inducing Latent Semantic Analysis

PLSA Probabilistic Latent Semantic Analysis

pLSI probabilistic Latent Semantic Indexing

SFBS Simple Forward-Backward Splitting

S-HOPM Symmetric Higher-Order Power Method

SNR Signal to Noise Ratio

SS-HOPM Shifted Symmetric Higher-Order Power Method

SVD Singular Value Decomposition

SVM Support Vector Machine

SVTD Singular Value based Tensor Decomposition

List of Symbols

$\otimes$	Tensor/outer product
$\bullet_k$	Contraction mode-k
$\boxtimes$	Kronecker product
$\odot$	Khatri-Rao product
$\|.\|_p$	norm p
∇f	The gradient of a differentiable function
∂f	The sub-gradient of a function f
β	The Lipschitz constant of the gradient of the differentiable function h in Forward-Backward Splitting theorem
Γ_0	The class of lower semi-continuous functions
γ_e	mapping from $\mathcal{L}$ to Ω
δ	Dirac delta function
ϵ_0	The gap between $\widehat{\boldsymbol{A}}$ and $\boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$
ϵ_1	Comon index
ϵ_2	Moreau-Macchi index
ϵ_3	Amari index
$\boldsymbol{\lambda}$	Coefficient vector
$\boldsymbol{\Lambda}$	A diagonal matrix with unit modulus entries
σ	An optimal permutation
$\varphi(k)$	Probability of topic k
$\boldsymbol{\varphi}^K$	The vector of probabilities of dimension K: $[\varphi(1), \varphi(2), \ldots, \varphi(K)]^T$
$\widehat{\boldsymbol{\varphi}}$	The estimation of $\boldsymbol{\varphi}$

$\Phi(k)$	The cumulative distribution of hidden variables: $\Phi(k) = \text{Prob}(h \leq k)$
$\boldsymbol{\Phi}$	The vector of cumulative distribution of topics: $[\Phi(1), \Phi(2), \ldots, \Phi(K)$
Ω	A dictionary
$\mathbb{1}_N$	A vector of ones of dimension N
$\boldsymbol{A}^{(n)}$	mode-n loading matrix (factor)
$\boldsymbol{a}_k$	The vector of conditional probabilities of dimension D: $[f_k(1), f_k(2), \ldots, f_k(D)]^T$
$\boldsymbol{A}^{D\times K}$	The matrix of conditional probabilities: $[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_K]$
$\widehat{\boldsymbol{A}}$	The estimation of $\boldsymbol{A}$
$\boldsymbol{b}_n$	The bag-of-words of document n
$\boldsymbol{C}$	Correlation matrix
d	word index in a dictionary
D	Number of multi-view variables (words) in a dictionary
$\text{dom}(f)$	The domain of a function f
$\mathbb{E}$	Expectation
$f_k(d)$	Conditional probability of word d when the topic is k
$F_k(d)$	The cumulative distribution of words: $\sum_{i=1}^{d} f_k(i)$
$\boldsymbol{F}^{D\times K}$	The matrix of cumulative distribution of words
f_η^h	The hard-thresholding function with a threshold η
f_η^s	The soft-thresholding function with a threshold η
$\boldsymbol{\mathcal{G}}$	Core tensor of $\boldsymbol{\mathcal{T}}$
Gr(f)	The graph of a function f
h	Hidden variable (topic)
$\mathcal{H}$	Set of topics
$\boldsymbol{I}_N$	The identity matrix of size N
$i_{\mathcal{S}(.)}$	The indicator function of a closed convex set $\mathcal{S}$
krank{.}	Kruskal rank

K	Number of hidden variables (topics)
ℓ_q	norm q
ℓ	word index in a document
L	Document length
$\mathcal{L}$	The set of document lengths
L_n	The number of words in document n
N_c	The size of a corpus (a set of texts)
N	Tensor order
$\text{prox}_f(\boldsymbol{x})$	The proximity operator of a function f at point $\boldsymbol{x}$
$\text{proj}_{\mathcal{S}}(\boldsymbol{x})$	The projection of a vector $\boldsymbol{x}$ onto a closed convex set $\mathcal{S}$
Prob(.)	Probability
$\boldsymbol{P}$	Second order moments
$\boldsymbol{P}_e$	Empirical second order moments
Perm(N)	The set of permutations of N elements
$\boldsymbol{P}_p$	The estimation of a permutation matrix
$\boldsymbol{P}_\sigma$	The permutation matrix associated with the σ
R	Tensor rank
R°	Expected rank
$\mathcal{S}^P$	The simplex set of dimension P
$\boldsymbol{t}_{i_1,i_2,\ldots,i_{n-1},:,i_{n+1},\ldots,i_N}$	fiber mode-n of tensor $\boldsymbol{\mathcal{T}}$, where ":" denotes a MATLAB notation which means considering all values of way n
$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$	CP decomposition of tensor $\boldsymbol{\mathcal{T}}$
$\boldsymbol{\mathcal{T}}^{(n)}$	mode-n unfolding matrix of tensor $\boldsymbol{\mathcal{T}}$
$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$	Tucker decomposition of $\boldsymbol{\mathcal{T}}$
$\boldsymbol{\mathcal{T}}$	Third order moments
$\boldsymbol{\mathcal{T}}_e$	Empirical third order moments
$\boldsymbol{x}_i$	Multi-view variable (word)
$\boldsymbol{X}$	A matrix consists of all loading factors together

1 Introduction

Considering vectors and matrices as one-way and two-way arrays, respectively, the concept can be extended to *tensors* as multi-way arrays which can have more than two dimensions. Although vectors/matrices can be seen as tensors of order one or two, tensors usually refer to an array with more than two ways [Com14, SDLF+17].

As tensors are useful tools to identify unknown quantities due to the uniqueness of their decompositions, they have appeared in many applications so far, such as Blind Source Separation (BSS) based on data cumulants [Com14], component analysis in chemistry [Bro98], estimation and localization of sources (Direction Of Arrival (DOA)) [RCM+16] and medical image and signal processing [BAC+14], to name a few.

In addition, the importance of tensors has been revealed in many tasks of machine learning including classification [SDLF+17], data fusion [CMDL+15] and unsupervised clustering [AGH+14]. The latter has been widely appeared in data/text mining [BNJ03, AGH+14, RCG18], and is performed by the estimation of some statistical parameters in mixture models [AGH+14].

Throughout this thesis, we consider a particular mixture model, called single-topic model [BNJ03], which can be used to describe a hidden relation among observed data (see Section 3.2 and Fig. 3.1 for more detailed definition of single-topic model). In a single-topic model, a set of multi-view (observed) variables are generated according to their corresponding hidden (latent or non-observed) variable. For instance, a corpus of documents can be described by a single-topic model: the topic of each document is seen as a hidden variable and the words of that particular document are its multi-view

variables generated according to its topic. Note that single-topic models are not limited to describe a corpus of documents, it can also be useful in modeling social networks [AGH+14] and movie recommendation systems [BNJ03].

It has been shown that whenever observed data can be modeled by a single-topic model, some statistical parameters of the data are related to the tensor of moments [AHK12b]. To be more precise, the probability of the hidden variable (topic) and the conditional probability of multi-view variables (words) can be estimated via the decomposition of the tensor of third order moments, which is a method of moments [Pea94]. As mentioned before, by employing these estimated probabilities, the targeted data/text mining task (unsupervised clustering) can be performed by means of Bayes' law [RCG18].

The main goal of this thesis is to investigate the effect of the accuracy of tensor decompositions on estimated probabilities. In other words, we aim at answering the question of which type of tensor decomposition is more appropriate for probability estimation. In addition, exploring the drawbacks of the employed tensor decompositions for the above mentioned purpose in the literature and trying to propose proper solutions are other goals of this thesis.

The outcomes of the thesis are as follows:

(i) Regarding the estimation of the tensor of third order moments, we investigate the effect of moment estimators on the accuracy of probability estimations and the required size of corpus. Moreover, we propose an estimator which performs as well as the best estimator in the state-of-the-art but based on a much simpler concept, *i.e.* weighted averaging. In addition, our moment estimator performs slightly better for small dimension tensors and provides some advantages in terms of computational complexity.

(ii) Concerning tensor decomposition algorithms, we show experimentally that the probabilities estimated by algorithms employed in the literature are not always acceptable as probability values. More importantly,

the employed tensor decomposition algorithms are not robust to additive noise and suffer dramatically from rounding errors, especially when working with a real corpus.

Nevertheless, we propose to consider proper constraints (such as non-negativity or the simplex set) in tensor decompositions, which guarantees reliable results for the estimated probabilities, and provides much more accurate estimations. Moreover, these kinds of decompositions are more robust to additive noise and perform better on real text datasets.

In addition, we introduce an algorithm for constrained tensor decompositions based on the proximal concept, called Simple Forward-Backward Splitting (SFBS), which performs better than other constrained tensor decompositions especially in noisy scenarios and converges faster. Further, we discuss how one can apply SFBS with sparsity constraints, which is needed for other applications such as co-clustering [PS11]. Last but not least, due to utilizing a proximal regularization and the proximal concept, SFBS has a complete convergence guarantee even with non-convex constraints.

(iii) Due to permutation and scaling ambiguities in tensor decompositions, it is not straightforward to evaluate the performance of tensor decomposition algorithms. In this thesis, we introduce a performance index that we coin CorrIndex, which is invariant to permutation and scaling ambiguities. Contrary to existing methods in the literature, CorrIndex is more reliable in ill-conditioned cases, and also provides interpretable performance bounds, while keeping computational complexity to a reasonable level.

This thesis is organized as follows. In Chapter 2, we review some preliminaries about tensors and their decompositions. Chapter 3, explains the relation between tensor decompositions and the data mining problem, which is in the focus of attention of this thesis. Chapter 4 and Chapter 5, which

are devoted to the main contributions of this thesis, describe performance indices (including CorrIndex) and tensor decomposition algorithms (including SFBS), respectively. Finally, Chapter 6 concludes the thesis and mentions some perspectives.

Notation Throughout the thesis, arrays of numbers will be printed in boldface. More precisely, one-way and two-way arrays will be denoted in bold lowercase and bold uppercase, respectively, *e.g.* $\boldsymbol{v}$ and $\boldsymbol{M}$. Arrays with more than two indices will be denoted by bold calligraphic symbols, such as $\boldsymbol{\mathcal{A}}$. Sets and spaces will be noted in script font, like $\mathcal{S}$. Entries of arrays $\boldsymbol{v}$, $\boldsymbol{M}$ and $\boldsymbol{\mathcal{A}}$ will be noted v_i, M_{ij} and A_{ijk}, without bold font (since they are indeed scalar numbers).

2 Tensor

Contents

2.1 Introduction

In this chapter, we bring the required notations and preliminaries in this report about the tensor and its decompositions. Moreover, the challenges and limitations of tensor decompositions in practice will be discussed along with the relevant methods to face these difficulties.

2.2 TENSOR: NOTATIONS AND PRELIMINARIES

Tensor Tensors can be considered as a multi-linear maps from a vector space to another one [Com14], or they are simply multi-way (multi-dimensional or multi-index) numerical arrays [CZPA09, CMDL+15], or hypermatrices [Lim13]. The *order* of an array refers to the number of its ways [Com00]. Vectors and matrices are one-way and two-way arrays, respectively, but usually tensor refers to an array with three or more ways [SDLF+17].

Products

- Tensor/outer product ($\otimes$): It can be explained easily by considering an example. The tensor/outer product of a 3^{rd} order tensor, $\boldsymbol{\mathcal{A}}$, with entries A_{ijk} and a 4^{th} order tensor, $\boldsymbol{\mathcal{B}}$, with entries B_{lmnp} is a 7^{th} order tensor like $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$ whose components are $C_{ijklmnp} = A_{ijk}B_{lmnp}$. The orders add up under the tensor product.

- Contraction ($\bullet_k$): Unlike the tensor product, which increases the order, the contraction decreases the sum of orders by 2 [Com14]. $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \bullet_k \boldsymbol{\mathcal{B}}$ indicates the contraction product over mode-k between two tensors, $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, which is a summing up over the production of their mode-k entries. Note that if the order of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ are D_1 and D_2, $\boldsymbol{\mathcal{C}}$ is a tensor of order $D_1 + D_2 - 2$. For instance, in the previous examples of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, the contraction product over mode-2 is defined as follows:

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \underset{2}{\bullet} \boldsymbol{\mathcal{B}} \rightarrow C_{iklnp} = \sum_j A_{ijk}B_{ljnp} \tag{2.1}$$

  However, the contraction product between a matrix and a tensor of higher dimension is treated based on the order of putting them. To be more precise, $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{A}} \bullet_k \boldsymbol{M}$ is a contraction between the mode-k of $\boldsymbol{\mathcal{A}}$ and the first matrix index of $\boldsymbol{M}$, but $\boldsymbol{\mathcal{Q}} = \boldsymbol{P} \bullet_k \boldsymbol{\mathcal{A}}$ is a contraction between the mode-k of $\boldsymbol{\mathcal{A}}$ and the second matrix index of $\boldsymbol{P}$. This permits the compatibility with the usual matrix-matrix product.

In the following, two examples of the contraction product over the mode-3 of a 3rd order tensor, $\boldsymbol{\mathcal{A}}$, and two other matrices, namely $\boldsymbol{M}$ and $\boldsymbol{P}$, can be found:

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{A}} \underset{3}{\bullet} \boldsymbol{M} \rightarrow G_{ijt} = \sum_{k} A_{ijk} M_{kt} \tag{2.2}$$

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{P} \underset{3}{\bullet} \boldsymbol{\mathcal{A}} \rightarrow Q_{ijs} = \sum_{k} A_{ijk} P_{sk} \tag{2.3}$$

- Kronecker product ($\boxtimes$): Assume that $\boldsymbol{A}$ and $\boldsymbol{B}$ are two matrices of size $I_1 \times I_2$ and $I_3 \times I_4$ respectively. Their *Kronecker* product is a matrix $\boldsymbol{C}$ of size $I_1 I_3 \times I_2 I_4$ defined by:

$$\boldsymbol{C} = \boldsymbol{A} \boxtimes \boldsymbol{B} \overset{\text{def}}{=} \begin{bmatrix} A_{11}\boldsymbol{B} & A_{12}\boldsymbol{B} & \dots & A_{1I_2}\boldsymbol{B} \\ A_{21}\boldsymbol{B} & A_{2I_2}\boldsymbol{B} & \dots & A_{2I_2}\boldsymbol{B} \\ \dots & \dots & \dots & \dots \\ A_{I_11}\boldsymbol{B} & A_{I_12}\boldsymbol{B} & \dots & A_{I_1I_2}\boldsymbol{B} \end{bmatrix} \tag{2.4}$$

- Khatri-Rao product ($\odot$): This product is actually a column-wise Kronecker product, and it should be performed between two matrices with the same number of columns. If $\boldsymbol{A}^{I_1 \times I_2} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_{I_2}]$ and $\boldsymbol{B}^{I_3 \times I_2} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_{I_2}]$, where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are the i^{th} columns of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively, then:

$$\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B} \overset{\text{def}}{=} [\boldsymbol{a}_1 \boxtimes \boldsymbol{b}_1, \boldsymbol{a}_2 \boxtimes \boldsymbol{b}_2, \dots, \boldsymbol{a}_{I_2} \boxtimes \boldsymbol{b}_{I_2}] \tag{2.5}$$

Unfolding or matricizing For convenience, tensors are sometimes transformed into matrices [Com14]. This transformation is called *unfolding* or *flattering* and can be done in each mode. The resulting matrix in mode n is called the *mode-n unfolding* [Com14] and is denoted by $\boldsymbol{\mathcal{T}}^{(n)}$.

Unfolding can be executed in several manners, but in this thesis we shall use the one, which returns matrices with the same number of rows as in the chosen mode. The obtained vector by fixing all indices except one is called the fiber of tensor $\boldsymbol{\mathcal{T}}^{I_1 \times I_2 \times \dots \times I_N}$, *e.g.* the mode-$n$ fiber is

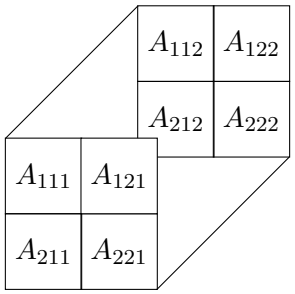

Figure 2.1: The tensor $\boldsymbol{\mathcal{A}}$ of dimension $2 \times 2 \times 2$.

$\boldsymbol{t}_{i_1,i_2,\ldots,i_{n-1},:,i_{n+1},\ldots,i_N}$. Mode-$n$ unfolding, *i.e.* $\boldsymbol{\mathcal{T}}^{(n)}$, is a matrix obtained by concatenating fibers mode-n in order of increasing indices [CMDL+15].

The following example expresses the unfolding of the tensor $\boldsymbol{\mathcal{A}}$ of dimension $2 \times 2 \times 2$, depicted in Fig. 2.1, in its three modes:

$$\boldsymbol{A}^{(1)} = \begin{bmatrix} A_{111} & A_{121} & A_{112} & A_{122} \\ A_{211} & A_{221} & A_{212} & A_{222} \end{bmatrix}$$

$$\boldsymbol{A}^{(2)} = \begin{bmatrix} A_{111} & A_{211} & A_{112} & A_{212} \\ A_{121} & A_{221} & A_{122} & A_{222} \end{bmatrix}$$

$$\boldsymbol{A}^{(3)} = \begin{bmatrix} A_{111} & A_{211} & A_{121} & A_{221} \\ A_{112} & A_{212} & A_{122} & A_{222} \end{bmatrix}$$

2.3 Exact tensor decomposition

In noiseless scenarios, an exact tensor decomposition is feasible. An advantage of exact *tensor* decomposition over exact *matrix* decomposition is the uniqueness of the decomposition under a mild condition (cf. Section 2.3.1). Recall that the decomposition of a matrix as a summation of rank-1 matrices is not unique, for example if $\boldsymbol{M}_{I\times J} = \boldsymbol{A}_{I\times K}\boldsymbol{B}_{J\times K}^T = \sum_k^K \boldsymbol{a}_k\boldsymbol{b}_k^T$, then $\boldsymbol{M} = (\boldsymbol{AQ})(\boldsymbol{Q}^{-1}\boldsymbol{B}^T)$ for any invertible matrix $\boldsymbol{Q}$ ($\boldsymbol{QQ}^{-1} = \boldsymbol{I}$).

2.3.1 Canonical Polyadic (CP) tensor decomposition

A decomposable tensor of order N is a tensor product of N vectors [Com14], *i.e.*, $\boldsymbol{\mathcal{D}} = \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(2)} \ldots \otimes \boldsymbol{a}^{(N)}$. According to [Hit27], any tensor can be written as a linear combination of a finite number of decomposable tensors,

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \lambda_r \boldsymbol{\mathcal{D}}(r), \tag{2.6}$$

where $\boldsymbol{\mathcal{T}}$ is a tensor of order N, and $\boldsymbol{\mathcal{D}}(r) = \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{a}_r^{(2)} \ldots \otimes \boldsymbol{a}_r^{(N)}$. The decomposition described in (2.6) is called the *Polyadic decomposition* [Hit27], and if it is unique, it is called the *Canonical Polyadic (CP)* decomposition or also *CANDECOMP* or *PARAFAC* [Com14]. In compact form, (2.6) can be represented as $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, in which $\boldsymbol{\lambda}$ is a *coefficient vector* of size R containing the values of λ_r and $\boldsymbol{a}_r^{(1)}, \boldsymbol{a}_r^{(2)}, \ldots \boldsymbol{a}_r^{(N)}$ are the r^{th} columns of N *loading matrices (factors)* $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$, respectively. The N matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$ are called mode-1, mode-2, ..., mode-N loading matrices, respectively, since their columns are responsible for the construction of the first, second, ..., N^{th} dimension of $\boldsymbol{\mathcal{T}}$ [CMDL$^+$15].

Rank For each tensor, the minimum value of R for which (2.6) holds is called the tensor rank [Com14]. Therefore, the rank of a decomposable tensor is one.

Uniqueness A sufficient condition of uniqueness of the CP decomposition is as follows [Com21]:

$$2R + N - 1 \leq \sum_{n=1}^{N} \text{krank}\{\boldsymbol{A}^{(n)}\},$$

where $N \geq 3$ and krank$\{.\}$ denotes the *Kruskal*[1] rank of a matrix.

However, a necessary condition can be obtained according to the *expected* rank of a tensor for the uniqueness of its CP decomposition. Based on the

[1]The Kruskal rank of a matrix is defined as the maximum number k such that any k columns of that matrix are linearly independent.

number of knowns and unknowns in (2.6), the expected rank of a tensor is defined as [Com14]:

$$R^{\circ} \triangleq \left\lceil \frac{D}{1 - N + \sum_{i=1}^{N} n_i} \right\rceil,$$

where N is the order of the tensor of dimensions $n_1 \times \ldots \times n_N$ and $D = \prod_{i=1}^{N} n_i$. It has been shown in [COV14] that if $D \leq 15000$ and $R \leq R^{\circ} - 1$ ensures almost surely the uniqueness of decomposition (2.6). As mentioned before, if this polyadic decomposition is unique, it can be called the *Canonical Polyadic (CP)* decomposition [Com14].

Unfolding of CP Note that the mode-n unfolding of the tensor $\boldsymbol{\mathcal{T}}$ can be expressed based on its CP decomposition, $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, as follows:

$$\boldsymbol{\mathcal{T}}^{(n)} = \boldsymbol{A}^{(n)} \text{diag}(\boldsymbol{\lambda}) \Big(\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)} \Big)^{T}, \quad (2.7)$$

where $\text{diag}(\boldsymbol{\lambda})$ is a diagonal matrix by holding $\boldsymbol{\lambda}$ as its diagonal.

2.3.2 Tucker tensor decomposition

For any tensor of order N and of dimensions $n_1 \times n_2 \times \ldots \times n_N$, one can find N loading matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$ of dimension $n_1 \times R_1,\ n_2 \times R_2,\ \ldots,\ n_N \times R_N$ respectively, and a core tensor, $\boldsymbol{\mathcal{G}}$, of dimension $R_1 \times R_2 \times \ldots \times R_N$ (smaller than or equal to dimension of $\boldsymbol{\mathcal{T}}$) as follows:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{A}^{(1)} \underset{1}{\bullet} \boldsymbol{A}^{(2)} \underset{2}{\bullet} \ldots \boldsymbol{A}^{(N)} \underset{N}{\bullet} \boldsymbol{\mathcal{G}}, \quad (2.8)$$

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]. \quad (2.9)$$

If $\boldsymbol{\mathcal{G}}$ in (2.8) is diagonal, then it expresses the CP decomposition, otherwise it is Tucker decomposition proposed by Tucker in 1966 [Tuc66].

Uniqueness The uniqueness condition of the CP decomposition stated in Subsection 2.3.1 ($R \leq R^{\circ} - 1$) can be refined by means of the dimensions

$R_1 \times R_2 \times \ldots \times R_N$ as follows [COV14, Com21]:

$$R \leq \left\lceil \frac{\prod_{i=1}^{N} R_i}{1 - N + \sum_{i=1}^{N} R_i} \right\rceil - 1.$$

In addition, the rank R of the tensor $\boldsymbol{\mathcal{T}}$ and the dimensions $R_1 \times R_2 \times \ldots \times R_N$ are related via the following inequalities [Com21]:

$$\max_i\{R_i\} \leq R \leq \min_i\{\prod_{j \neq i} R_j\}.$$

Even if the minimal dimensions $R_1 \times R_2 \times \ldots \times R_N$ are used in (2.8), Tucker decomposition is not unique. It can be seen easily by replacing each loading factor with its QR decomposition [Com21], for example $\boldsymbol{A}^{(i)} = \boldsymbol{Q}^{(i)}\boldsymbol{R}^{(i)}$, where the matrices $\boldsymbol{R}^{(i)}$ are of dimensions $R_i \times R_i$ and ${\boldsymbol{Q}^{(i)}}^H\boldsymbol{Q}^{(i)} = \boldsymbol{I}$. Therefore, (2.9) can be rewritten as below:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}'}; \boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)}, \ldots, \boldsymbol{Q}^{(N)}]\!], \tag{2.10}$$

where the new core tensor $\boldsymbol{\mathcal{G}'}$ has the following form:

$$\boldsymbol{\mathcal{G}'} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{R}^{(1)}, \boldsymbol{R}^{(2)}, \ldots, \boldsymbol{R}^{(N)}]\!].$$

2.3.3 Multi-Linear/Higher Order Singular Value Decomposition (MLSVD/HOSVD)

Equation (2.10) can be seen as Tucker decomposition of $\boldsymbol{\mathcal{T}}$ under the constraint that all loading factors should be semi-unitary (*i.e.* ${\boldsymbol{Q}^{(i)}}^H\boldsymbol{Q}^{(i)} = \boldsymbol{I}$). This decomposition is also known as Multi-Linear/Higher-Order Singular Value Decomposition (MLSVD/HOSVD) [Com14].

Therefore, any tensor $\boldsymbol{\mathcal{T}}$ admits a decomposition of MLSVD/HOSVD:

$$\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)}, \ldots, \boldsymbol{Q}^{(N)}]\!], \tag{2.11}$$

where ${\boldsymbol{Q}^{(i)}}^H\boldsymbol{Q}^{(i)} = \boldsymbol{I}$ and $\boldsymbol{\mathcal{G}}$ is a core tensor of dimension $R_1 \times R_2 \times \ldots \times R_N$. Moreover, $[R_1,\ R_2,\ \ldots,\ R_N]$ is called the *multi-linear* rank of the tensor $\boldsymbol{\mathcal{T}}$ [DLDMV00].

Calculating exact MLSVD/HOSVD Assume that a multi-linear rank, $[R_1,\ R_2,\ \ldots,\ R_N]$, has been chosen such that an exact MLSVD/HOSVD of $\boldsymbol{\mathcal{T}}$ can be obtained. It can be shown that $\boldsymbol{Q}^{(i)}$ in (2.11) is nothing else but the matrix of left singular vectors of the i^{th} mode unfolding of $\boldsymbol{\mathcal{T}}$, *i.e.* $\boldsymbol{\mathcal{T}}^{(i)}$:

$$\boldsymbol{\mathcal{T}}^{(i)} = \boldsymbol{Q}^{(i)}\boldsymbol{\Sigma}^{(i)}{\boldsymbol{V}^{(i)}}^{H}, \tag{2.12}$$

and $\boldsymbol{\Sigma}^{(i)}$ and $\boldsymbol{V}^{(i)}$ are the diagonal matrix of singular values and the matrix of right singular vectors, respectively [Com21]. In addition, $\boldsymbol{\mathcal{G}}$ in (2.11) can be computed by the means of ${\boldsymbol{Q}^{(i)}}^{H}$ as follows:

$$\boldsymbol{\mathcal{G}} = [\![\boldsymbol{\mathcal{T}}; {\boldsymbol{Q}^{(1)}}^{H}, {\boldsymbol{Q}^{(2)}}^{H}, \ldots, {\boldsymbol{Q}^{(N)}}^{H}]\!].$$

2.4 Approximate tensor decomposition

Different types of rank (*e.g.* generic, typical) are discussed in [Com14, Section V.B]. The generic rank is the rank that is encountered with the probability one [CGLM08, CBDC09], when the entries of a tensor are drawn independently according to a continuous probability distribution. In practice, almost always the tensor of data is corrupted by noise, which can be considered as an additive random tensor whose entries are random variables with continuous probability distributions. As a result, the rank of such a noisy tensor is generic [Com14]. This generic rank is much larger than the dimension, hence, its CP decomposition would not be unique.

Since the rank of the non-noisy part is much lower, it is typically preferred to perform a *low-rank* or a *low multi-linear rank* approximation, which not only eliminates partially the additive noise, but also results in less expensive computations. More importantly, the low-rank approximation may then have a unique CP decomposition.

2.4.1 Truncated MLSVD/HOSVD

Like truncated SVD in the matrix case, truncated MLSVD/HOSVD can be obtained by keeping a part of the columns of $\boldsymbol{Q}^{(i)}$ corresponding to the largest

singular values. To be more precise, considering truncated multi-linear rank $R_i^t \leq R_i$, *truncated* MLSVD/HOSVD of $\boldsymbol{\mathcal{T}}$ is its MLSVD/HOSVD by $[R_1^t,\ R_2^t,\ \ldots,\ R_N^t]$.

In addition to being well-posed, truncated MLSVD/HOSVD is also useful as a pre-processing for other decompositions such as CP. These pre-processings include noise reduction (as it is a low multi-linear rank approximation) and dimension reduction (if the core tensor estimated via truncated MLSVD/HOSVD is used for the targeted decomposition).

2.4.2 Low-rank CP approximation

Most CP decomposition algorithms require the rank of tensor in advance. Determining the rank in advance becomes more challenging in noisy scenarios, since the rank is generic. CORCONDIA is a well-known method for tensor rank estimation in terms of CP decomposition [BK03]. In addition, as mentioned in Subsection 2.4.1, truncated MLSVD/HOSVD provides some pre-processings for targeted decompositions such as CP. Therefore, one can apply low-rank CP approximation on the core tensor estimated via truncated MLSVD/HOSVD, which has a lower rank compared to the primary tensor (due to the noise reduction) and has also lower dimensions, which result in less expensive computations.

However, even if the rank of a tensor is estimated, the low-rank approximation is ill-posed [Com14]. In practice, in order to face this difficulty, some constraints must be added to CP decomposition, as described in the next Section.

2.5 Constrained CP decomposition

Since in practice, data stored in the form of a tensor are usually corrupted by noise, the best rank-R approximation must be estimated. Although low-rank approximation is useful and unavoidable, generally it is ill-posed [HL13, DSL08], since the set of tensors of rank at most R is not closed [Com14].

Therefore, imposing some constraints such as non-negativity [LC09] or angular separation [LC14] in CP decomposition is proposed in the literature to overcome this difficulty.

We shall see the complete formulation of constrained CP decomposition in Section 5.5.1. Here, we bring a general description of some prevalent constraints in this context:

- **Non-negativity:** All the loading matrices and the coefficient vector are supposed to be non-negative even if the tensor to decompose is not completely non-negative due to noise. There is a rich literature on tensor decomposition under this constraint [CMDL$^+$15, JMC18].

- **Simplex:** This constraint usually appears in the models involving a probabilistic analysis [AGH$^+$14] (cf. Section 3.3.1.1). All or a part of the columns/rows of the loading matrices and/or the coefficient vector in (2.6) should belong to a simplex set defined by: $\mathcal{S} \triangleq \{\boldsymbol{x} : \boldsymbol{x} \geq 0, \|\boldsymbol{x}\|_1 = 1\}$

- **Orthogonality:** it can be imposed between the columns of loading matrices or between decomposable tensors, $\boldsymbol{\mathcal{D}}_r$, in (2.6) [Com14,Kol01]. The former constraint is widely used in blind source separation after the standardization stage [CJ10].

- **Coherence:** In order to guarantee the existence of a low-rank approximation, imposing orthogonality is sufficient but not necessary. In fact, imposing a constraint on the coherences of loading matrices is sufficient [SC15, NMC20, LC14]. This constraint limits the coherence, *i.e.*, the minimal angle between the columns of loading matrices [Com14], and this angle is generally much smaller than $\pi/2$. In addition, such an angular constraint often has a physical meaning, especially in antenna array processing [LC14].

- **Sparsity:** In some applications [HLP14], it may be needed to impose sparsity constraints on loading matrices or coefficient vectors. The

exact sparsity constraint is also known as *cardinality* constraint, and defines a non-convex set [HSL16]. The cardinality of a vector can be measured by the ℓ_0 pseudo-norm (or the counting norm), which is the number of its non-zero entries. Since ℓ_0 is a non-convex function, sometimes its convex approximations such as the ℓ_1 norm [PB14], or Smoothed ℓ_0 (SL0) [MBZJ08] are used instead.

2.6 Conclusion

In this chapter, we reviewed some preliminaries about tensor and its decompositions in terms of exact and approximate decompositions. In addition, it was mentioned that imposing a proper constraint on the decomposition would be very efficient in some particular applications such as the one investigated in this thesis, *i.e.* data/text mining. In the next chapter, the relation between data/text mining with constrained tensor decomposition will be discussed in more details.

3 Data mining and tensors

Contents

3.1 Introduction

In this chapter, firstly, we review a particular data model, which can be used to describe a hidden relation among observed data. In addition, a task of data mining (clustering by means of estimated probabilities) based on this model will be described. Secondly, we shall explain an existing tensor approach as well as other points of view to perform this task. We will see the first step in utilizing the tensor approach is *moment estimation*, therefore,

the last section of this chapter is devoted to discuss about different moment estimators and their properties.

3.2 Hidden and multi-view variable models

Most of events, which are in the center of interest for investigation, admit the nature of multi-modality. The multi-modal property permits to observe different aspects of a phenomenon, which is namely multi-modal data. This can be seen in various natural (earthquakes, weather, ...) and human-made (corpus of texts, news, social networks, ...) data [FCC16]. The entities that relate and affect these multi-modal data together are hidden (latent) aspects, which are called hidden (latent) variables [AGH+14]. Multi-view models are useful in the context of multi-modal recordings (such as texts, audios and videos for a particular event), and offer a means to improve the estimation of particular features of latent variables that are present in several recordings. In this respect, they are related to data fusion and data mining (cf. Section 3.3).

Data mining covers a wide range of methods and tools utilized for discovering knowledge from data [HPK11]. In the context of multi-modal data (*e.g.* text, audio and video for the same event), mining can be considered as the estimation of probabilities of the variables [RMD11, AHESK10]. As it will be explained in Section 3.3, these probabilities can be applied for some ultimate tasks such as unsupervised clustering [RCG18] and classification [BNJ03]. Multi-view models are useful tools to represent the relationships in this framework [Whi09].

Multi-view models or Latent Variable Models (LVM) refer to a vast range of models in which one or some hidden variables exist [RCG18]. These models originate in a simple model for describing documents, called *unigram*, in which words of each document are drawn independently according to a single multinomial distribution [BNJ03].

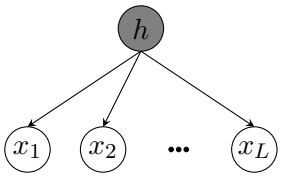

Figure 3.1: The mixture of unigrams or single-topic model with its multi-view variables (x_i) and their corresponding hidden variable (h).

In this work, we assume the graphical model depicted in Fig. 3.1 as the multi-view model. In this model, a set of multi-view (observed) variables, $\{x_1, x_2, \ldots, x_L\}$ are represented that are generated according to a conditional multi-nomial distribution conditioned to their corresponding hidden (latent or non-observed) variable h. This model is called the *mixture of unigrams* [BNJ03], since it can be obtained by augmenting unigram models with a random discrete hidden variable h, as the *topic* of each document.

The model in Fig. 3.1 is also known as the *single-topic* model [RCG18, AGH$^+$14], since the multi-view variables (words in a corpus of documents) are generated according to a *single* hidden variable (topic in each document). However, there are also other more general models such as probabilistic Latent Semantic Indexing (pLSI) [Hof99] and Latent Dirichlet Allocation (LDA) [BNJ03], which attempt to correspond multi-view variables to several (not only one) hidden variables by means of a mixture of weights or Dirichlet distribution on a set of hidden variables. These models as well as other types of LVM such as Gaussian mixture and Hidden Markov Model (HMM) [Jor04, AHK12b] are out of the scope of this dissertation.

Employing a model as in Fig. 3.1 is very common in several tasks of data (e.g. text) mining [RCG18, AGH$^+$14, BNJ03]. In Section 3.3, these applications will be explained in brief.

3.3 Data mining with hidden and multi-view variable models

The daily increasing amount of online available textual data have raised many challenging tasks such as unsupervised learning or clustering, which can be investigated by text mining [HPK11]. Text mining as a special case of data mining is the center of attention of this thesis. Although the graphical model of Fig. 3.1 can be used for a wide range of learning problems [Mur12], in this section, we customize the formulation for the particular problem of text mining. The following explanation describes how one can encode textual data (words) into numerical vectors, hence applying matrix/tensor analysis on text become feasible. In addition, the required pre-processing steps to extract keywords from each text are briefly explained in Section 5.6.3, yet these steps are out of scope of this thesis.

Let L be the number of words in a given document, $\boldsymbol{x}_\ell$ the observed words, $\ell \in \mathcal{L} = \{1, 2, \ldots, L\}$, and h a topic encoded into a discrete variable taking K possible integer values, say from $\mathcal{H} = \{1, 2, \ldots, K\}$ with probability $\varphi(k) = \text{Prob}(h = k)$. All words belong to a known encoded dictionary $\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D\}$ of cardinality D. In other words, we can consider a mapping γ_e (generally not injective) from $\mathcal{L}$ to $\{1, 2, \ldots, D\}$ such that $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)}$. In the context of text mining, D would be the number of words and K the number of topics. The number of documents, N_c, is generally larger than D.

Besides the probability of topics, the conditional probability of each word given the topic is also an important parameter in text mining tasks, *e.g.* unsupervised clustering of documents, as explained in the sequel. We denote the conditional probability of each word $\boldsymbol{u}_d$ of the dictionary Ω, given a particular topic, $h = k$, by $f_k(d) = \text{Prob}(\boldsymbol{x} = \boldsymbol{u}_d | h = k)$. Therefore, according to the simplifying assumptions described in Section 3.3.1.1, the joint distribution of $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L]$ can be written as:

$$p_{\boldsymbol{X}}(\boldsymbol{u}_{\gamma_e(1)}, \ldots, \boldsymbol{u}_{\gamma_e(L)}) = \sum_{k=1}^{K} \varphi(k)\, f_k(\gamma_e(1)) \ldots f_k(\gamma_e(L)), \qquad (3.1)$$

Table 3.1: List of required notations and definitions

Notation	Description
L	Number of words
$\boldsymbol{x}_\ell$ $\ell \in \mathcal{L} = \{1, 2, \ldots, L\}$	Observed words
$\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d, \ldots, \boldsymbol{u}_D\}$	Encoded dictionary
γ_e	mapping $\boldsymbol{x}_\ell$ to $\boldsymbol{u}_d$
$\mathcal{H}$	Set of K topics
K	Number of topics
D	Dictionary size
N_c	Corpus size
$\varphi(k)$	Probability of topic k
$f_k(d)$	Probability of $\boldsymbol{u}_d$ condition to topic k
$\boldsymbol{a}_k = [f_k(1), f_k(2), \ldots, f_k(D)]^T$	Conditional probabilities of words to topic k
$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_K]$	Conditional probability of words

which is referred to as the *naive Bayes* model. The main task is to estimate the quantities appearing in the right-hand side of (3.1) from realizations of $\boldsymbol{X}$.

The term $f_k(\gamma_e(\ell))$ in (3.1) represents the conditional dependence of the word $\boldsymbol{x}_\ell$ in a document to its topic k. In other words, the number of occurrences of a particular word in a document with a particular topic can be completely different from its occurrence in another document with another topic. For example, the word "win" is much more probable in a sport document than a document about cooking. On the other hand, the word "meat" may occur more in the latter document. For the convenience, above notations are listed in Table 3.1.

Estimating the probabilities in (3.1) is the first step in a variety of machine learning and data mining applications, some of which are described below.

Unsupervised clustering [RCG18] In the following, as an example, it is explained how one can employ the estimated probabilities in order to cluster documents in an unsupervised way. However, this approach is applicable for other kinds of data admitting the graphical model in Fig. 3.1.

Assume that $\varphi(k)$ and $f_k(\gamma_e(\ell))$ in (3.1) for $k = 1, \ldots, K$ and $\ell = 1, \ldots, L$ have been recovered. The hidden topic, h, of a document, say `text`, can be inferred according to the following conditional probability:

$$\text{Prob}(h = k|\texttt{text}) = \frac{\text{Prob}(\texttt{text}|h = k)\varphi(k)}{\sum_{k=1}^{K} \text{Prob}(\texttt{text}|h = k)\varphi(k)}, \tag{3.2}$$

and $\text{Prob}(\texttt{text}|h = k)$, according to the simplifying assumptions described in Section 3.3.1.1, can be calculated as follows:

$$\begin{aligned}\text{Prob}(\texttt{text}|h = k) &= \text{Prob}(\boldsymbol{u}_{\gamma_e(1)}, \ldots, \boldsymbol{u}_{\gamma_e(\ell)}, \ldots, \boldsymbol{u}_{\gamma_e(L)}|h = k) \\ &= \prod_{\ell=1}^{L} \text{Prob}(\boldsymbol{u}_{\gamma_e(\ell)}|h = k) \\ &= \prod_{\ell=1}^{L} f_k(\gamma_e(\ell))\end{aligned}$$

In fact, the document `text` is assigned to the topic k, which maximizes $\text{Prob}(h = k|\texttt{text})$. Note that the equality in (3.2) is Bayes' law.

Document classification [BNJ03] The choice of feature is very important in some classification methods. In the document classification problem, although the individual words of all documents can be treated as a rich feature set, it is enormously large [Joa98].

A solution would be a dimensionality reduction, which can be done by considering the estimated conditional probabilities in (3.1). In other words, $f_k(d)$ can be viewed as the d^{th} feature of a document titled by topic k. Moreover, since d refers to the index of a word in the assumed dictionary, the dimensionality reduction can be done, for instance, by keeping just two words of each document, which have the highest conditional probabilities.

After extracting features, document classification can be performed by applying a proper method such as Support Vector Machine (SVM) [SC08].

Movie recommendation [BNJ03] This application can be performed on a data set of users, which indicates their preferred movies. A portion of indicated movies are used to train the recommendation system, while a preferred movie of each user is held out for testing. The goal is to predict the held-out movie for each user, which can be viewed as a recommended movie in practice.

Considering users as the hidden variable and their preferred movies as the multi-view variables, one can assign the held-out movie by estimating the associated probabilities in (3.1), and then by following the procedure described in the above mentioned application of unsupervised clustering.

3.3.1 Tensor approach

In this section, the relation between the left-hand side of (3.1) and moments will be explained. In addition, two main tensor approaches for estimating the probabilities on the right-hand side of (3.1) will be described. As these methods utilize moments to estimate probabilities, they are considered as the *method of moments* [Pea94].

3.3.1.1 Method of moments based on tensor decomposition

As mentioned before, the main goal is to estimate the probability of the hidden variable (topic) and the conditional probability of multi-view variables (words). In this section, we show how these probabilities can be estimated by means of moments and what is the role of tensor decomposition in this vein. To this end, some assumptions are required that are listed below [AGH$^+$14, AHK12a]:

- conditional probabilities do not depend on the order of words (exchangablility), *e.g.*

$$\text{Prob}(\boldsymbol{u}_{\gamma_e(p)}, \boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{u}_{\gamma_e(r)}|h) = \text{Prob}(\boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{u}_{\gamma_e(r)}, \boldsymbol{u}_{\gamma_e(p)}|h)$$

- words are conditionally independent given the topic, *i.e.*

$$(\boldsymbol{u}_{\gamma_e(p)}|h) \perp\!\!\!\perp (\boldsymbol{u}_{\gamma_e(q)}|h)$$

- words have the same conditional distribution given the topic, *i.e.*

$$(\boldsymbol{u}_{\gamma_e(p)}|h) \sim (\boldsymbol{u}_{\gamma_e(q)}|h).$$

Note that although these assumptions cannot be thoroughly satisfied in real textual data, such *simplifying* assumptions are required to identify probabilities through a tensor approach. In this thesis, we consider dictionaries and texts that satisfy these assumptions.

In the sequel, the second and third order moments will be needed:

$$\boldsymbol{P} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q\}, \tag{3.3}$$

$$\boldsymbol{\mathcal{T}} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q \otimes \boldsymbol{x}_r\}, \tag{3.4}$$

where $\boldsymbol{P}$ is a $D \times D$ symmetric matrix and $\boldsymbol{\mathcal{T}}$ a $D \times D \times D$ symmetric tensor. These moments do not depend on $\{p, q, r\}$ provided these three integers are all different, which ensures the conditional independence assumed in (3.1). But note that $\{\gamma_e(p), \gamma_e(q), \gamma_e(r)\}$ may not be different because γ_e is not injective.

$\boldsymbol{x}_\ell$ is encoded to $\boldsymbol{u}_d$, and as in [AGH$^+$14], $\boldsymbol{u}_d$ is chosen as the columns of the $D \times D$ identity matrix. For example, $\boldsymbol{u}_d$ for the i^{th} word in the dictionary is as follows:

$$\boldsymbol{u}_{\{d=i\}}(j) = \begin{cases} 0 & if \quad j \neq i \\ 1 & if \quad j = i \end{cases}, \quad 1 \leq j \leq D.$$

Because of this choice for $\boldsymbol{u}_d$, these moments exhibit the following relations:

$$\boldsymbol{P} = \sum_{k=1}^{K} \varphi_k \; \boldsymbol{a}_k \otimes \boldsymbol{a}_k, \tag{3.5}$$

$$\boldsymbol{\mathcal{T}} = \sum_{k=1}^{K} \varphi_k \; \boldsymbol{a}_k \otimes \boldsymbol{a}_k \otimes \boldsymbol{a}_k, \tag{3.6}$$

where $\boldsymbol{a}_k$ constructs the k^{th} column of a matrix, say matrix $\boldsymbol{A}$. Note that $\boldsymbol{a}_k$ contains the values of $f_k(d)$ for all d and each k. We provide the proof of (3.5) here, which is not in the literature:

Proof.

$$\begin{aligned}\boldsymbol{P} &\overset{\text{def}}{=} \mathbb{E}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q\} \overset{a}{=} \mathbb{E}_h\Big\{\mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p|h\} \otimes \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_q|h\}\Big\} \\ &\overset{b}{=} \sum_{k=1}^{K} \varphi_k \big(\sum_{d=1}^{D} f_k(d)\boldsymbol{u}_d\big) \otimes \big(\sum_{d'=1}^{D} f_k(d')\boldsymbol{u}_{d'}\big) \\ &\overset{c}{=} \sum_{k=1}^{K} \varphi_k \ \boldsymbol{a}_k \otimes \boldsymbol{a}_k,\end{aligned}$$

where

- a: the separation of $\mathbb{E}$ over $(\boldsymbol{x}, h)$ and using the conditional independency of $\boldsymbol{x}_p, \boldsymbol{x}_q$ given h;
- b: the definition of $\mathbb{E}$;
- c: the specific encoding of $\boldsymbol{u}_d$ into the columns of the $D \times D$ identity matrix.

The proof of (3.6) is exactly similar. □

Rewriting (3.1) as (3.3) and (3.4) reveals the nice property that arrays $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ are actually the *joint probabilities* of observations, *i.e.*, $P_{ij} = \text{Prob}\{\boldsymbol{x}_p = \boldsymbol{u}_i, \boldsymbol{x}_q = \boldsymbol{u}_j\}$ and $\mathcal{T}_{ijk} = \text{Prob}\{\boldsymbol{x}_p = \boldsymbol{u}_i, \boldsymbol{x}_q = \boldsymbol{u}_j, \boldsymbol{x}_r = \boldsymbol{u}_k\}$.

Once the empirical approximation of the moments in (3.3) and (3.4) is obtained, the probability of the hidden variable (topic) and the conditional probabilities of multi-view variables (words), *i.e.* $\boldsymbol{\varphi}$ and $\boldsymbol{A}$, can be estimated via tensor decomposition according to (3.6). As we shall see in Chapter 5, this decomposition may be either developed in a constrained manner (under non-negative constraints as described in Section 5.5.2.1 or under simplex set constraints as marked out in Section 5.5.2.2) or be handled in an unconstrained manner as will be expressed in Section 5.4. In addition, it may be performed with or without utilizing the second order moments matrix, $\boldsymbol{P}$. The advantages and disadvantages of each method are investigated experimentally in Section 5.6.2.

3.3.1.2 Diagonalization of two moment matrices

In [AHK12a], a joint diagonalization algorithm is promoted uses two moment matrices, namely $\boldsymbol{P}$ and a matrix obtained by the contraction $\boldsymbol{T}(\boldsymbol{\eta}) = \boldsymbol{\mathcal{T}} \bullet_3 \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is a randomly drawn vector. The idea may seem interesting, but the algorithm unfortunately has never been implemented and tested, according to the available literature.

The Algorithm A of [AHK12a] starts with an SVD of $\boldsymbol{P}$ as $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Then, the matrix $\boldsymbol{B}(\boldsymbol{\eta}) = \boldsymbol{U}^\top \boldsymbol{T}(\boldsymbol{\eta}) \boldsymbol{V} \left(\boldsymbol{U}^\top \boldsymbol{P} \boldsymbol{V}\right)^{-1}$ is computed, as well as its K dominant eigenvectors, $\boldsymbol{\xi}_k$. Then, an estimate of the columns of matrix $\boldsymbol{A}$ (the conditional probabilities of multi-view variables) is given by $\widehat{\boldsymbol{a}}_k = \boldsymbol{U}\boldsymbol{\xi}_k$, up to a scaling factor depending on how $\widehat{\boldsymbol{a}}_k$ are normalized. The "eigenvalues" of $\boldsymbol{\mathcal{T}}$ can be obtained in a second stage by contraction as $\widehat{\varphi}_k = \boldsymbol{\mathcal{T}} \bullet_1 \widehat{\boldsymbol{a}}_k \bullet_2 \widehat{\boldsymbol{a}}_k \bullet_3 \widehat{\boldsymbol{a}}_k$.

3.3.2 Other methods and points-of-view

Expectation Maximization (EM) For decades, the most popular unsupervised and heuristic approach to learn the parameters of a LVM model was Expectation Maximization (EM) [DLR77], which is an iterative local search based on Maximum Likelihood (ML) [AGH$^+$14].

Although EM has several merits such as the ease of understanding, implementing and utility for any LVM, it suffers from slow convergence specially when the model dimension increases and also from suboptimal local results [RCG18]. Moreover, not only it requires users to apply many heuristics to obtain acceptable results, but also for some kinds of models such as Latent trees, it is NP-hard [AGH$^+$14].

In addition, supposing a prior (*e.g.* Dirichlet) is needed in some particular LVM – such as the LDA model [BNJ03] – and some existing EM methods [NMTM00], and increasing the generality of the model is more costly and challenging in terms of unsupervised estimation procedure [AHK12a].

Latent Semantic Analysis (LSA) [DDF$^+$90] Analysis of texts and documents is a great challenge in Natural Language Processing (NLP), Information Retrieval (IR) and information filtering [Hof99].

One of the problems in organizing, searching and understanding such a vast amount of digitalized documents is learning the meaning and the usage of the words with a data-driven model [Ble12]. In other words, one of the big challenges is the transition from the *lexical* level (*i.e.* actual written text) to the *semantical* level (*i.e.* the intention of the writer) [Hof99]. Some issues such as *polysems* (*i.e.* a word with several meanings) and synonyms (*i.e.* different words with similar meanings) make this challenge even harder.

Latent Semantic Analysis (LSA) [DDF$^+$90] is a well-known method, which maps the high dimensional space of the words of the vocabulary into a lower dimensional space, called *latent semantic space*. This method works with a matrix, $\boldsymbol{W}$, whose rows and columns are referred to the documents of the corpus and the words of the vocabulary, respectively. $W(i,j)$ shows how often the j^{th} word occurs in the i^{th} document, and the inner product of two columns of $\boldsymbol{W}$ shows how much those two words are *co-occurred*[1]. LSA computes a low rank approximation of $\boldsymbol{W}$, say $\boldsymbol{X}$, by means of the Singular Value Decomposition (SVD) of $\boldsymbol{W}$. Although the non-zero inner product of the columns of $\boldsymbol{X}$ doesn't depend on the co-occurrence of the corresponding words, it shows some latent relation between them, such as synonym.

As LSA does not distinguish the types of latent relation (*e.g* synonym, antonym, hypernym, hyponym[2], ...), some other algorithms such as Probabilistic LSA (PLSA) [Hof99], Polarity Inducing LSA (PILSA) [YQ12] and Multi-Relational LSA (MRLSA) [CYM13] (which is based on tensor decomposition) were proposed to improve the power of distinction between different

[1]In practice, instead of simply word frequency, *TF-IDF* score [SWY75] is used, which better shows the importance of a word. TF-IDF consists of two factors, TF-IDF = TF $\times$ IDF, where TF expresses the word frequency and IDF shows how much information a words has, *i.e.* it is a common or rare word across all the documents [SWY75].

[2]For instance, "red" is hyponym of "color", and "color" is hypernym of "red", and "red" is co-hyponym of "blue".

types of latent relation.

Unlike the method of moments, which tries to estimate some latent statistics and the parameters of the observed data, LSA and its variants try to measure semantic data relation such as synonym, antonym, hypernym, etc. In addition, LSA and its variants are based on cosine similarity of the raw observed data, whereas the method of moments attempts to estimate parameters using the empirical moments of the observed data. In general, LSA and its variants are applicable to understand the semantical layer from the lexical level, while the method of moments are useful to extract the informative features of each of documents in the corpus.

The approach used in this thesis In this thesis, we focus on the two-phase *tensor decomposition* approach for data mining (*i.e.* estimating the probabilities) with the graphical model in Fig. 3.1, which was described in section 3.3.1.1. In order to make the procedure more clear, we bring a brief review below:

In the first phase, the second and third order moments are approximated by observing the multi-view variables (words). Then, in the second phase, one tries to estimate the probabilities by decomposing or using the tensor of third order sample moments (sometime the second order sample moments is also used).

The more precise the sample moments are approximated in the first phase, the more accurate estimation of the probabilities is obtained in the second phase. We shall see the effect of moments approximation in Section 5.6.2 in practice. In addition, in Section 3.4, some methods of moments approximation are reviewed.

3.4 Moment estimation

As mentioned before, the first step in the procedure of data mining (the probability estimation) by means of tensor decomposition described in Sec-

tion 3.3.1.1 is the moment estimation. In this section, first, two generative processes are expressed, then, some of the methods of the moment estimation are reviewed (Section 3.4.3) and are introduced (Section 3.4.4). Finally, these moment estimators are compared by some simulations in Section 3.4.5.

It is worth first reviewing some notations of Section 3.3:

- L: the number of words in a given document,
- $\boldsymbol{x}_\ell, \ell \in \mathcal{L} = \{1, 2, \ldots, L\}$: the observed words in a document,
- h: a topic encoded into a discrete variable taking K possible integer values, say in $\mathcal{H} = \{1, 2, \ldots, k, \ldots, K\}$ with the probability $\varphi(k) = \text{Prob}(h = k)$
- $\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d, \ldots, \boldsymbol{u}_D\}$: the encoded dictionary of cardinality D
- $f_k(d) = \text{Prob}(x = \boldsymbol{u}_d | h = k)$: the conditional probability of each word $\boldsymbol{u}_d$ of the dictionary Ω, given a particular topic, $h = k$
- $\boldsymbol{a}_k$: a vector of dimension D containing the values of $f_k(d)$ for all d
- $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k, \ldots, \boldsymbol{a}_K]$.

3.4.1 Generative process

In this subsection, we explain how one can generate a set of synthetic encoded words that are related to the same topic (cf. Fig. 3.1), with these two properties simultaneously:

- conditionally independent given the topic
- having the same conditional distribution

In the following, we describe two generative processes, which generate a topic (as hidden variable) and words (as multi-view variables) such that the above mentioned properties will be developed.

3.4.1.1 Generative process based on the cumulative distribution

To generate a data set, we need first to define the distributions $\varphi(k)$ and $f_k(d)$ for all $(k, \boldsymbol{u}_d) \in \mathcal{H} \times \Omega$. The values of $\varphi(k)$ are stored in a K-dimensional vector $\boldsymbol{\varphi}$. Similarly, the values of $f_k(d)$ are stored in a $D \times K$ matrix $\boldsymbol{A}$.

But it is actually useful to use their cumulative distributions, $\Phi(k) = \text{Prob}(h \leq k)$ and $F_k(d)$ with an appropriate encoding, as we shall see. The values of $\Phi(k)$ are stored in a $K \times 1$ vector $\boldsymbol{\Phi}$ obtained from $\boldsymbol{\varphi}$ by cumulated sum of its entries. Similarly, the values of $F_k(d)$ are obtained by cumulated sum of the entries of matrix $\boldsymbol{A}$ over first dimension and stored in a $D \times K$ matrix $\boldsymbol{F}$. Our generative algorithm goes along the following lines:

- draw $z \in [0, 1]$ with uniform distribution, and pick a topic (hidden variable), $h = \Phi^{-1}(z)$, by selecting the first entry in Φ that is larger than z.
- assuming that $h = k$ has been drawn in the first step, then, for each $\ell \in \{1, 2, \ldots, L\}$,
  - draw $z_\ell \in [0, 1]$ with uniform distribution, and pick a word (multi-view variable), $d = \gamma_e(\ell) = F_k^{-1}(z_\ell)$, by selecting the first entry in the k^{th} column of $\boldsymbol{F}$ that is larger than z_ℓ.
  - set $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)} = \boldsymbol{u}_d$.

3.4.1.2 Generative process based on the multinomial distribution

This generative process is stated in [RCG18]. After defining the distributions $\varphi(k)$ and $f_k(d)$ for all $(k, \boldsymbol{u}_d) \in \mathcal{H} \times \Omega$, a topic and words are drawn based on some multinomial distributions with event probabilities $\boldsymbol{\varphi}$ and $\boldsymbol{A}$, respectively. The "multinomial distribution" was firstly introduced by Ronald Fisher in 1925 [Fis25]. Generally speaking, it is an extension of the binomial distribution to an event with more than two outcomes [UC08]. Some required moments of the multinomial distribution are expressed in Remark 1.

The details of this generative algorithm are as follows:

- draw the topic (hidden variable), h, according to a multinomial distribution[3] with K event probabilities of $\boldsymbol{\varphi}^{K\times 1}$.
- assuming that $h = k$ has been drawn in the first step, then, for each $\ell \in \{1, 2, \ldots, L\}$,
  - draw a word (multi-view variable), $d = \gamma_e(\ell)$, according to a multinomial distribution with D event probabilities of $\boldsymbol{a}_k^{D\times 1}$.
  - set $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)} = \boldsymbol{u}_d$.

3.4.2 Moment consistency

Reminding that second and third order moments in (3.3) and (3.4) are defined as follows:

$$\boldsymbol{P} \overset{\text{def}}{=} \mathbb{E}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q\},$$
$$\boldsymbol{\mathcal{T}} \overset{\text{def}}{=} \mathbb{E}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q \otimes \boldsymbol{x}_r\}.$$

As it is shown in Section 3.3.1.1, by encoding $\boldsymbol{x}_\ell$ to $\boldsymbol{u}_d$ and by choosing $\boldsymbol{u}_d$ as the columns of the $D \times D$ identity matrix, the above mentioned moments have the following matrix and tensor structure (mentioned in (3.5) and (3.6)):

$$\boldsymbol{P} = \sum_{k=1}^{K} \varphi_k \; \boldsymbol{a}_k \otimes \boldsymbol{a}_k,$$
$$\boldsymbol{\mathcal{T}} = \sum_{k=1}^{K} \varphi_k \; \boldsymbol{a}_k \otimes \boldsymbol{a}_k \otimes \boldsymbol{a}_k.$$

These moments, using the distributions φ_k and $f_k(d)$, can be considered as *true* moments, while any *sample* estimate of (3.3) and (3.4) should converge to the true moments $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ defined in (3.5) and (3.6). Hence, it is essential to evaluate the *consistency* of the moment estimator in order to make sure that sample moments indeed converge to the true joint probabilities. It is preferred that this convergence occurs by observing the least possible number of realizations of $\boldsymbol{x}_\ell$.

[3] This can be done with the commands `mnrnd` and `numpy.random.multinomial` in MATLAB and Python, respectively.

To this end, first, sample moments are calculated via a moment estimator, then, the difference between true moments ((3.5) and (3.6)) and the estimated ones will be reported in terms of Euclidean norm. Therefore, the moment consistency can be considered as a measure of evaluating the performance of moment estimators, and we shall compare some algorithms of the moment estimation in Section 3.4.5 in terms of moment consistency.

3.4.3 State-of-the-art

3.4.3.1 Simple averaging [AFH+12]

This estimator is simply the averages of large number of realizations (a *corpus* of documents), say N_c, to obtain an acceptable approximation of $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$. In each realization, a random encoded topic, k, is drawn in the way described in the first step of generative processes in Sections 3.4.1.1 and 3.4.1.2. Then, according to the chosen topic, *three* random encoded words (it is assumed that a document consists of at least three words) are drawn, $\{\boldsymbol{x}_p = \boldsymbol{u}_{\gamma_e(p)}, \boldsymbol{x}_q = \boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{x}_r = \boldsymbol{u}_{\gamma_e(r)}\}$, based on the second step of generative processes mentioned in Sections 3.4.1.1 and 3.4.1.2. At the end, by using the following averages, an *empirical* approximation of the second and the third order moments are obtained with sample moments below:

$$\boldsymbol{P}_e = \frac{1}{N_c} \sum_{n=1}^{N_c} \boldsymbol{u}_{\gamma_e(p)} \otimes \boldsymbol{u}_{\gamma_e(q)}, \tag{3.7}$$

$$\boldsymbol{\mathcal{T}}_e = \frac{1}{N_c} \sum_{n=1}^{N_c} \boldsymbol{u}_{\gamma_e(p)} \otimes \boldsymbol{u}_{\gamma_e(q)} \otimes \boldsymbol{u}_{\gamma_e(r)}. \tag{3.8}$$

Note that in this method of estimation, the length of each document (L_n) does not matter, and just three words (can be the first, the middle and the last word) in each document participate in the sample moment estimation. Due to these facts, providing a reasonable moment consistency with simple averaging is very difficult and requires a large size corpus, *i.e.* large N_c in (3.7) and (3.8). We shall show this drawback of simple averaging in practice by some simulations in Section 3.4.5.

3.4.3.2 Zou's estimator [ZHPA13]

For the rest of this section, a well-known concept in text mining, called *bag-of-words*, is required, which is a representation of each document. Assume that a document consists of L_n words ($\boldsymbol{x}_\ell, \quad \ell = 1, \ldots, L_n$) that are encoded to $\boldsymbol{u}_d = \boldsymbol{u}_{\gamma_e(\ell)}$. Then, the bag-of-words of such a document is a vector of dimension D, and is defined as follows:

$$\boldsymbol{b}_n \stackrel{\text{def}}{=} \sum_{\ell=1}^{L_n} \boldsymbol{u}_{\gamma_e(\ell)}, \tag{3.9}$$

where $b_n(d)$ shows how many times the d^{th} word appears in the document n.

The Zou's estimator ($\boldsymbol{P}_{\text{Zou}}^{D\times D}, \boldsymbol{\mathcal{T}}_{\text{Zou}}^{D\times D\times D}$) is as follows:

$$P_{\text{Zou}}(i,i) = \frac{1}{N_c}\sum_{n=1}^{N_c} \frac{b_n(i)^2 - b_n(i)}{L_n(L_n-1)} \tag{3.10}$$

$$P_{\text{Zou}}(i,j) = \frac{1}{N_c}\sum_{n=1}^{N_c} \frac{b_n(i) - b_n(j)}{L_n(L_n-1)} \tag{3.11}$$

$$T_{\text{Zou}}(i,i,i) = \frac{1}{N_c}\sum_{n=1}^{N_c} \frac{b_n(i)^3 - 3b_n(i)^2 + 2b_n(i)}{L_n(L_n-1)(L_n-2)} \tag{3.12}$$

$$T_{\text{Zou}}(i,i,j) = \frac{1}{N_c}\sum_{n=1}^{N_c} \frac{b_n(i)^2 b_n(j) - b_n(i)b_n(j)}{L_n(L_n-1)(L_n-2)} \tag{3.13}$$

$$T_{\text{Zou}}(i,j,k) = \frac{1}{N_c}\sum_{n=1}^{N_c} \frac{b_n(i)b_n(j)b_n(k)}{L_n(L_n-1)(L_n-2)}, \tag{3.14}$$

where N_c is the number of documents in the considered corpus. Equations (3.10) to (3.14) express that in spite of simple averaging, Zou's estimator takes into account all the words of a document by utilizing $\boldsymbol{b}_n$, as well as the length of the document, L_n.

3.4.3.3 Ruffini's estimator [RCG18]

With the same assumptions of Zou's estimator, Ruffini's estimators, *i.e.* $\boldsymbol{P}_{\text{Ruffini}}$ and $\boldsymbol{\mathcal{T}}_{\text{Ruffini}}$, are defined as follows ($i < j$):

$$P_{\text{Ruffini}}(i,j) = \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j) - \delta_{i=j})}{\sum_{n=1}^{N_c} L_n(L_n - 1)} \tag{3.15}$$

$$\begin{aligned} T_{\text{Ruffini}}(i,j,k) = {} & \delta_{i<j<k} \frac{\sum_{n=1}^{N_c} b_n(i)b_n(j)b_n(k)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)} \\ & + \delta_{(i=j<k)\vee(i<j=k)} \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j)-1)b_n(k)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)} \\ & + \delta_{i=j=k} \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j)-1)(b_n(k)-2)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)}. \end{aligned} \tag{3.16}$$

By employing bag of words of each document in the corpus, nominators of the above expressions correspond to implementing moments (summation and production of random variables), and their denominators are the coefficients, which make this estimator unbiased.

Note that by computing the upper triangular part of $\boldsymbol{P}_{\text{Ruffini}}$ and $\boldsymbol{\mathcal{T}}_{\text{Ruffini}}$, and then, by setting the lower triangular part identical to the upper side, symmetrization would be done.

Like Zou's estimator, all the words of the documents in a corpus along with the length of each document have been also considered in Ruffini's estimator. However, in Zou's estimator, the averaging is done for each document, and then, all the documents are averaged together with the same weight. But Ruffini's estimator averages all the documents according to a particular weight for each of them, *i.e.* $\frac{L_n(L_n-1)}{\sum_{n=1}^{N_c} L_n(L_n-1)}$ [RCG18]. Therefor, if all the documents have the same length ($L_n = L$), the two estimators will produce the same estimation. Moreover, Ruffini's estimator is less sensitive to the noise [RCG18].

It is shown that Ruffini's estimator is *unbiased* [RCG18]. We make this

proof easier to follow by providing much more details below ($i \neq j$):

$$
\begin{aligned}
&\mathbb{E}\{P_{\text{Ruffini}}(i,j)\} = \mathbb{E}\left\{\frac{\sum_{n=1}^{N_c} b_n(i)b_n(j)}{\sum_{n=1}^{N_c} L_n(L_n-1)}\right\} \\
&= \frac{1}{\sum_{n=1}^{N_c} L_n(L_n-1)} \sum_{n=1}^{N_c} \mathbb{E}\{b_n(i)b_n(j)\} \\
&= \frac{1}{\sum_{n=1}^{N_c} L_n(L_n-1)} \sum_{n=1}^{N_c}\sum_{k=1}^{K} \varphi(k)\mathbb{E}\{b_n(i)b_n(j)|h=k\} \\
&= \frac{1}{\sum_{n=1}^{N_c} L_n(L_n-1)} \sum_{n=1}^{N_c}\sum_{k=1}^{K} \varphi(k)[\underbrace{\text{Cov}(b_n(i)b_n(j)|h=k)}_{I} \\
&+ \underbrace{\mathbb{E}(b_n(i)|h=k)\mathbb{E}(b_n(j)|h=k)}_{II}] \\
&\overset{(*)}{=} \frac{1}{\sum_{n=1}^{N_c} L_n(L_n-1)} \sum_{n=1}^{N_c}\sum_{k=1}^{K} \varphi(k)[\underbrace{-L_n\boldsymbol{a}_k(i)\boldsymbol{a}_k(j)}_{I} + \underbrace{L_n\boldsymbol{a}_k(i)L_n\boldsymbol{a}_k(j)}_{II}] \\
&= \sum_{k=1}^{K} \varphi(k)\boldsymbol{a}_k(i)\boldsymbol{a}_k(j) = P(i,j),
\end{aligned}
$$

where "Cov(.,.)" denotes the *covariance* of two random variables, and the equality $(*)$ is due to the properties of multinomial distribution, which are described in Remark 1.

Remark 1. *Suppose that* $\boldsymbol{y}$ *is a random vector of dimension* D *holding multinomial distribution,* i.e. $\boldsymbol{y} \sim$ *multinomial* $(t, [p_1, \ldots, p_D])$*, where "*t*" is the number of the trials of "*D*" events with probabilities* $[p_1, \ldots, p_D]$*, and of course* $\sum_{d=1}^{D} p_d = 1$*. According to the multinomial distribution [UC08], we have:*

- *mean* $= \mathbb{E}\{y(d)\} = tp_d$,
- *variance* $= \mathbb{E}\{y(d), y(d)\} = tp_d(1-p_d)$,
- *covariance* $= \mathbb{E}\{y(d), y(d')\} = -tp_d p_{d'}$

Algorithm 1 Standard averaging for estimating the second order moments

Input: D, a corpus of N_c documents of length $[L_1, \ldots, L_{N_c}]$

Output: $\boldsymbol{P}_{\text{Stand-Ave}}$

1: $\boldsymbol{P}_{\text{Stand-Ave}} = \text{zeros}(D,D)$
2: **for** $n = 1, 2, \ldots, N_c$ **do**
3: Save the index of words of document "n".
4: $\boldsymbol{P}_{\text{temp}} = \text{zeros}(D,D)$
5: count-pair $= 0$
6: **for** all possible pairs of words, ex. $(\text{word}_1, \text{word}_2)$ **do**
7: $\boldsymbol{P}_{\text{temp}}(\text{word}_1, \text{word}_2) = 1 + \boldsymbol{P}_{\text{temp}}(\text{word}_1, \text{word}_2)$
8: count-pair $= 1 +$ count-pair
9: **end for**
10: $\boldsymbol{P}_{\text{Stand-Ave}} = \boldsymbol{P}_{\text{Stand-Ave}} + \dfrac{L_n}{\sum_{n=1}^{N_c} L_n} \dfrac{\boldsymbol{P}_{\text{temp}}}{\text{count-pair}}$
11: **end for**
12: Symmetrize $\boldsymbol{P}_{\text{Stand-Ave}}$

3.4.4 Proposed: Standard averaging

By taking into account the length and all the words (not only three) of each document, we propose an algorithm, which we coin as *standard averaging*, and can be seen as a weighted averaging based on the length of each document.

Algorithm 1 describes standard averaging in details. Note that symmetrization in line 12 is executed by computing $\dfrac{\boldsymbol{P} + \boldsymbol{P}^T}{2}$. We observed experimentally that this method of symmetrization shows a better performance compared to that of Ruffini's estimator in which the values of the lower triangular part will be set identical to the upper one at the end of the algorithm. The algorithm for the third order moments is similar, except that in line 6, all possible *triples* would be considered. Note that the symmetrization of a third order tensor (cf. line 12) requires averaging over six permutations of indices.

Simulations in Section 3.4.5 show that standard averaging performs better than the most recent estimator, *i.e.* Ruffini's estimator, when the dimension of dictionary, D, is not very large. More details about performances will be given in Section 3.4.5. Moreover, standard averaging provides some advantages in terms of computational complexity:

- standard averaging requires less number of multiplications in the estimation of $\boldsymbol{\mathcal{T}}$,
- the number of required additions in estimating $\boldsymbol{P}$ does not depend on the size of dictionary (D), which is an advantage in case of large dimension dictionaries.

Comparison of computational complexity in terms of number of multiplications and additions are explained in the next paragraph.

Computational complexity By assuming the same method of symmetrization, we can compare the computational complexity of standard averaging and Ruffini's estimator in terms of multiplications and additions. In this calculation, we consider the symmetrization of Ruffini's estimator, *i.e.* setting the lower triangular part identical to the upper one. Note that to symmetrize the output of Algorithm 1 in this way, the symmetric pair of $(\text{word}_1, \text{word}_2)$, *i.e.* $(\text{word}_2, \text{word}_1)$, will be ignored in the set of all possible pairs (line 6 of Algorithm 1). Remind that, as $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ are symmetric, the number of their free parameters are $\frac{D(D+1)}{2}$ and $\binom{D+3-1}{3}$, respectively [Com14].

One can observe that the number of required multiplications for computing $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ in (3.15) and (3.16) for Ruffini's estimator are $(1+N_c)\frac{D(D+1)}{2}+N_c$ and $(1+2N_c)\binom{D+3-1}{3}+2N_c$, respectively. However, these values in standard averaging are $N_c\frac{D(D+1)}{2}+2N_c$ and $N_c\binom{D+3-1}{3}+2N_c$. Therefore, in estimating $\boldsymbol{\mathcal{T}}$, standard averaging costs fewer multiplications, and in estimating $\boldsymbol{P}$, the comparison depends on the values of N_c and D.

The number of additions in computing $\boldsymbol{P}$ in (3.15) for Ruffini's estimator is $(1+\sum_{n=1}^{N_c} L_n)D+3N_c$, while the same quantity for standard averaging is

Table 3.2: Compare the number of required multiplications

Estimator	$\boldsymbol{P}$	$\boldsymbol{\mathcal{T}}$
Ruffini	$(1+N_c)\frac{D(D+1)}{2}+N_c$	$(1+2N_c)\binom{D+2}{3}+2N_c$
standard averaging	$N_c\frac{D(D+1)}{2}+2N_c$	$N_c\binom{D+2}{3}+2N_c$

$2N_c+2\sum_{n=1}^{N_c}\binom{L_n}{2}$. Therefore, the number of additions in standard averaging does not depend on D, which could be preferable for large values of D. These results are also mentioned in Table 3.2.

3.4.5 Simulations

All computer experiments reported in this section have been executed either on a laptop with a processor of 3.1 GHz Intel Core i5, 16 GB RAM or on a PC with a processor of 3.2 GHz Intel Core i5, 8 GB RAM, both running macOS Mojav and MATLAB 2019a.

3.4.5.1 The effect of corpus size on the moment consistency

As mentioned before, simple averaging does not provide acceptable moment consistency and a naive way to compensate this drawback is increasing the size of the corpus (the number of documents or realizations of the topic/hidden variable and words/multi-view variables).

In order to show this challenge in practice, we generated N_c realizations of our synthetic data set, $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)}$, (according to the method explained in Section 3.4.1.2) with the following parameter values: the number of hidden variables (topics), $K = 4$; the number of multi-view variables (words), $D = 8$; the number of realizations for estimating moments (the size of corpus, *i.e.* the number of documents), $N_c = [2^9, 2^{10}, \ldots, 2^{17}]$. The minimum and the maximum length of the generated documents are 3 and 100, respectively.

Lastly, we calculated sample moments with the moment estimators described in Section 3.4.3 and 3.4.4, and then, compared them to the true ones (3.5) and (3.6). Fig. 3.2 reports the discrepancy between both in terms

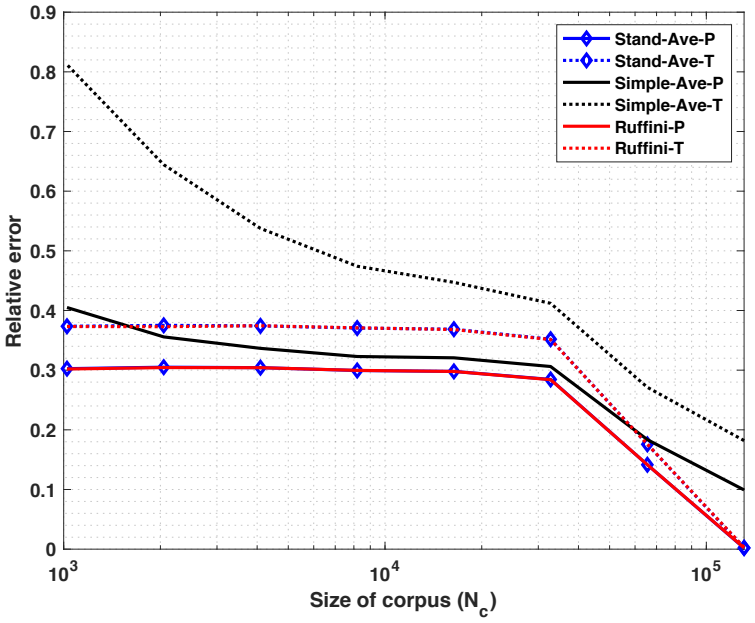

Figure 3.2: The consistency of the moment estimates. The relative error in the estimation of second order moments ($\boldsymbol{P}$) and third order moments ($\boldsymbol{\mathcal{T}}$) are plotted in solid and dotted line, respectively. Note that the plot of standard averaging and Ruffini's estimator are superimposed. In this figure, the larger the size of the corpus, the fewer the number of corpuses, to keep the total block length (and hence the computational load) constant.

of Euclidean norm. As can be seen in Fig. 3.2, simple averaging estimation, in the way described in Section 3.4.3.1, is consistent, if it performs on a corpus of size $N_c > 10000$. Ruffini's estimator and standard averaging have the same relative error (they are superimposed in Fig. 3.2), however, we shall see the difference between the performance of Ruffini's estimator and of standard averaging in Section 3.4.5.2 by investigating the effect of dictionary cardinality.

3.4.5.2 The effect of dictionary cardinality on the moment consistency

In this section, we investigate the effect of dictionary cardinality, D, on the performances of the moment estimators. For all the experiments, we con-

sider a corpus of $N_c = 100$ documents whose length L_n are chosen randomly between a minimum ($L_{\min} = 3$) and a maximum length ($L_{\max} = 100$). The topics and the words of each document are generated according to the generative process described in Section 3.4.1.2. Each experiment averaged over the results of 50 different probabilities $\boldsymbol{\varphi}$ and $\boldsymbol{A}$. In these series of experiments, we either plot the moment consistency versus a range of values of D (the number of words) or K (the number of topics). We shall conclude that standard averaging performs better than Ruffini's estimator, when dictionary cardinality, D, is less than 50.

Figures 3.3 and 3.4 show the relative error in estimating $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ versus a range of values of D, when K is fixed to 10. Since in Fig. 3.3, the difference of standard averaging and Ruffini's estimator is not significantly obvious, we plot the results of standard averaging and Ruffini's estimator separately in Fig. 3.4. For the same reason, in the rest of figures of this section, we just show the results of standard averaging and Ruffini's estimator, since the results of simple averaging is much worse than standard averaging and Ruffini's estimator.

As it can be seen in Figs. 3.3 and 3.4, not only standard averaging is a much more better estimator than simple averaging, but also it performs slightly better than Ruffini's estimator when dictionary cardinality, D, is less than 50.

In order to show that the only critical parameter is dictionary cardinality and not K, we performed two other experiments in Fig. 3.5 and Fig. 3.6. In Fig. 3.5, we set a small value for D ($D = 15$), and plot the relative errors versus a range of values of K, from 10 to 100. On the other hand, in Fig. 3.6, we set a large value for D ($D = 65$), and then, plot the relative error versus the same range of values of K. These two experiments admit the same conclusion as before (standard averaging performs slightly better than Ruffini's when D is not very large), and also show that this conclusion does not depend on the amount of K (although standard averaging performs slightly better in estimating $\boldsymbol{P}$ even for large $D = 65$, the difference is

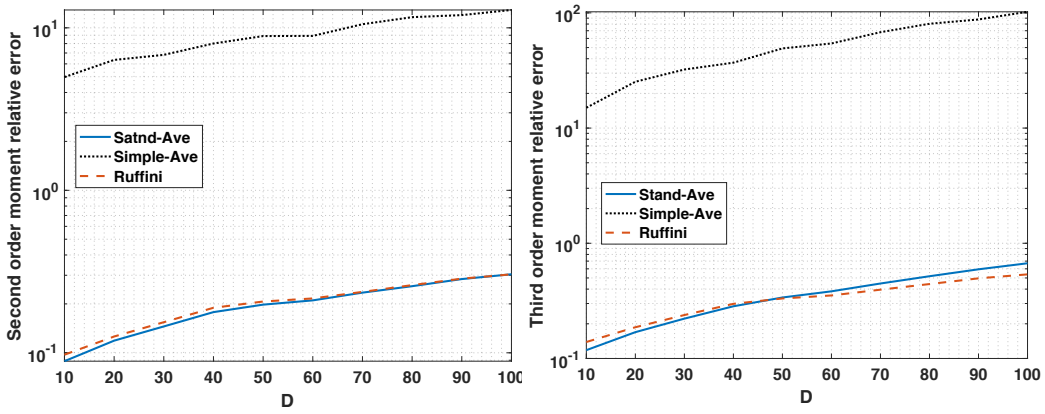

Figure 3.3: Comparison between all mentioned estimators in estimating $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ with $K = 10$, $D = [10, 20, 30, \ldots, 100], N = 100, L_{\min} = 3, L_{\max} = 100$, and averaging over 50 different probabilities $\boldsymbol{\varphi}, \boldsymbol{A}$, Left: The relative error of the second order moments estimation, Right: The relative error of the third order moments estimation.

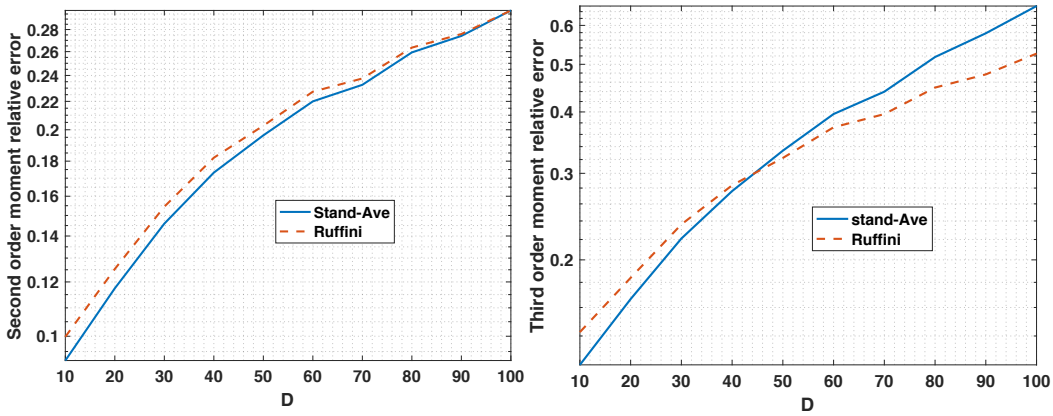

Figure 3.4: A zoom of Fig. 3.3 for showing details.

negligible).

As mentioned before, the length of a document, L, is a critical factor in improving the performance. When dictionary cardinality is small, the influence of document length is more significant, since the related words depending on the topic are much more repetitive in such a document. Ruffini's estimator decreases this influence by the denominator in its formula, *e.g.* by dividing by $\sum_{n=1}^{N_c} L_n(L_n-1)$, while standard averaging is simply a weighted averaging with respect to the length of all documents (*e.g.* by dividing over $\sum_{n=1}^{N_c} L_n$). Therefore, standard averaging is able to better show the effect of the length, L_n, in case of small D.

3.5 Conclusion

In this chapter, we described the hidden and multi-view variable model, which can be fitted to several multi-modal events. Then, we explained a task of data mining, *i.e.* probability estimation, which can be followed to perform an ultimate data mining task such as an unsupervised clustering. In Section 3.3, we described a tensor approach for this probability estimation as well as some other existing methods and points of view. It is mentioned that the first step in utilizing the tensor decomposition to approximate the probabilities of hidden variables and the conditional probabilities of multi-view variables is moment estimation. In Sections 3.4.3 and 3.4.4, we described some existing methods of moment estimation as well as our proposed method, *standard averaging*, along with some simulation comparisons in Section 3.4.5, using a synthetic generative model.

The next two chapters are devoted to the second step of the tensor approach for data/text mining, *i.e.* the constrained tensor decomposition and the performance index needed to evaluate different decomposition algorithms.

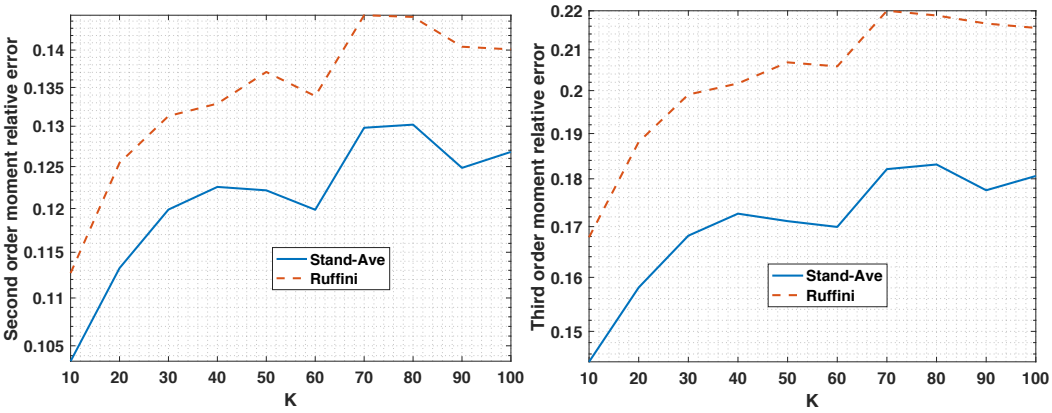

Figure 3.5: Comparison between standard averaging and the Ruffini's estimator in estimating $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ with $D = 15$, $K = [10, 20, 30, \ldots, 100]$, $N_c = 100, L_{\min} = 3, L_{\max} = 100$, and averaging over 50 different probabilities $\boldsymbol{\varphi}, \boldsymbol{A}$, Left: The relative error of the second order moments estimation, Right: The relative error of the third order moments estimation.

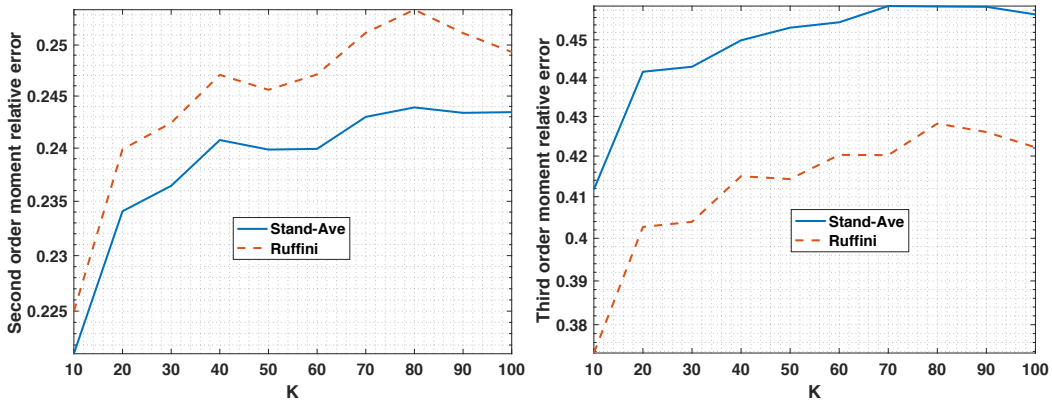

Figure 3.6: Comparison between standard averaging and Ruffini's estimator in estimating $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ with $D = 65$, $K = [10, 20, 30, \ldots, 100]$, $N_c = 100, L_{\min} = 3, L_{\max} = 100$, and averaging over 50 different probabilities $\boldsymbol{\varphi}, \boldsymbol{A}$, Left: The relative error of the second order moments estimation, Right: The relative error of the third order moments estimation.

4 Performance index

Contents

4.1 Introduction

As mentioned in Section 3.3.1, tensor decomposition is the second step in estimating probabilities from observed data. In order to compare different methods of decomposition, a proper performance index is needed to be applied on the estimated coefficient vector and loading matrices. Due to permutation and scaling ambiguities, evaluating these variables is not straightforward. Section 4.2 explains the existing challenges with this issue. In Section 4.3, the problem is formulated and a review of previous methods and measures is provided. The proposed index, *CorrIndex*, is presented in

Section 4.4, and its comparison with other existing measures along with discussions about its advantages are provided in Sections 4.5. The content of this chapter is mainly based on our paper [SCJBZ22], which is published in the journal *Signal Processing*, Elsevier.

4.2 Challenges in measuring performance

Permutation and scaling ambiguities are relevant issues in applications such as tensor decomposition [Com14] and Blind Source Separation (BSS) [CJ10]. Scaling refers to multiplication by a diagonal matrix with non-zero entries, which may be complex in the most general case, and permutation refers to the column permutation of a matrix, which is equivalent to multiplication by a permutation matrix. Firstly, these two ambiguities are inherent in tensor representations, by definition of tensors [Com14]. Secondly, in BSS, statistical independence is not affected by scaling or permutation of the sources [CJ10]. A mixing (or demixing) matrix can then only be estimated up to these ambiguities under the independence assumption. Although it is impossible to eliminate these ambiguities when working with real data sets, where the original parameters are not available, it is feasible to overcome these uncertainties in evaluating algorithm performance on synthetic data sets. Furthermore, reasonable comparisons on synthetic data sets are very helpful to choose adequately an appropriate algorithm to be applied on real data sets. Therefore, in order to report reasonably the performance indices of existing algorithms on synthetic data sets where the desired parameters are accessible, it is important to employ proper methods to measure the performances.

Assume that the original and estimated components have been normalized, then the only remaining ambiguities are the permutation and scaling with complex numbers of unit modulus. The existing approaches to measure the performances of the algorithms of BSS and tensor decomposition can be classified in three main categories: “greedy approaches”, “graph-based

methods" and "invariant indices". Greedy approaches [FIW+20, CLDA09, CKAC14, BBK18] try to assign the most correlated components estimated by an algorithm, and then compute the error of estimation or decomposition. Although most of these methods return back an estimated permutation as well as a performance index, they are not reliable in noisy conditions. In other words, the reported index (which is a criterion to evaluate the performance of algorithms in estimating components) by these kinds of methods depends directly on the manner of computing and analyzing the correlation matrix.

Graph-based methods [Kuh55, Gal83, Mun57] are originated from the well-known *optimal assignment* problem [DP14], which is itself a particular case of the *optimal transport* problem [PC+19]. Although these kinds of methods have the guarantee to find the optimal permutation, they are computationally expensive (as we shall see, the minimum cost is approximately $8N^3$ flops with a correlation matrix of size $N \times N$), especially when the correlation matrix is large.

However, the viewpoint of a third category, namely invariant indices [Com94, MM94, ACY+96], differs from the latter approaches. These *invariant* indices measure the performance regardless of permutation and scaling, and yield an index that can directly be used to compare algorithms. The reported indices of [Com94, MM94, ACY+96] are invariant to permutation and scaling, and the index of [Com94] provides the guarantee of a zero distance between estimated and original matrices up to column permutation and scaling, when the obtained index is zero. Nevertheless, the index of [Com94] is not bounded from above. More importantly, the upper bounds of indices of [MM94, ACY+96] have not been investigated, and it seems that these bounds are not easy to interpret. In addition, these methods are in the literature of source separation, and the indices introduced therein utilize the inverse (or pseudo-inverse) of the mixing matrix, which may involve an additional computational burden.

In next section, these three kinds of methods for measuring the perfor-

mance along with the problem formulation will be explained.

4.3 State-of-the-art

Let $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N] \in \mathbb{C}^{M\times N}$ and $\widehat{\boldsymbol{A}} = [\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2, \ldots, \hat{\boldsymbol{a}}_N] \in \mathbb{C}^{M\times N}$ be the original and estimated matrices respectively, where $\mathbb{C}^{M\times N}$ stands for the set of M by N complex-valued matrices. Let us denote the set of permutations of N elements by $\mathrm{Perm}(N)$, and denote by $\boldsymbol{P}_\sigma$ the matrix associated with the permutation $\sigma \in \mathrm{Perm}(N)$. If the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized by their L_2 norms, scaling ambiguity reduces to post-multiplication by a diagonal matrix $\boldsymbol{\Lambda}$ with entries of unit modulus.

Assume $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}+\boldsymbol{W}$, where the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized, $\boldsymbol{\Lambda}$ is a diagonal matrix with unit modulus entries and $\boldsymbol{W}$ is the error of estimation $\widehat{\boldsymbol{A}}$, which can be modeled as an additive noise. Since relevant issues in considered applications are permutation and scale ambiguity of *columns* of $\widehat{\boldsymbol{A}}$ comparing to $\boldsymbol{A}$, then right-multiplication of $\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$ is utilized here. More formally, the goal is to measure the gap defined below:

$$\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \min_{\sigma,\boldsymbol{\Lambda}} \|\boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda} - \widehat{\boldsymbol{A}}\|_F^2 \tag{4.1}$$

This gap can be computed with or without estimating permutation σ explicitly. Seeking the optimal permutation σ can be written as the following optimization problem:

$$\underset{\sigma}{\operatorname{argmin}} \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{a}_n - \hat{\boldsymbol{a}}_{\sigma(n)}\|_2^2 = \underset{\sigma}{\operatorname{argmax}} \sum_{n=1}^{N} |\boldsymbol{a}_n^H \hat{\boldsymbol{a}}_{\sigma(n)}|. \tag{4.2}$$

Let $C_{ij} = |\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j|$, and denote by $\boldsymbol{C}$ the matrix whose entries are C_{ij}. Then, if the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized by their L_2 norms, we have $0 \le C_{ij} \le 1$. In the sequel, three main approaches of measuring the distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ appeared in the literature are reviewed.

4.3.1 Methods based on correlation matrix

4.3.1.1 Greedy approach of [FIW+20]

In this approach, $\hat{\boldsymbol{a}}_j$ is assigned to $\boldsymbol{a}_i$ if C_{ij} has the maximum value in the j^{th} column of $\boldsymbol{C}$. This straightforward approach has two drawbacks. On one hand, if two or more maximum values occurred in the same row, a reasonable assignment could not be concluded. This happens for instance in far-field antenna array processing when sources are angularly close, in the presence of noise [SC15]. On the other hand, the delivered index is not reliable, since, even if the index is zero, one cannot guarantee $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$. The following toy numerical example illustrates this problem.

Assume that in an experiment matrix $\boldsymbol{C}$ is:

$$\boldsymbol{C} = \begin{bmatrix} 0.8 & 0.3 & 0.1 \\ 0.85 & 0.9 & 0.5 \\ 0.5 & 0.2 & 0.7 \end{bmatrix}. \tag{4.3}$$

The concluded assignment by this method is $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$, which is obviously not acceptable because column $\boldsymbol{a}_2$ is selected twice. Computing the square error via $\frac{1}{2}\sum_{n=1}^{3} \|\boldsymbol{a}_n - \hat{\boldsymbol{a}}_{\sigma(n)}\|_2^2$ by considering the assumption of normalized $\boldsymbol{a}_n$ and $\hat{\boldsymbol{a}}_{\sigma(n)}$ with respect to L_2 norm and by substituting the values of $|\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j|$ from C_{ij}, one obtains $3 - 0.85 - 0.9 - 0.7 = 0.55$, which is less than the exact error, $3 - 0.8 - 0.9 - 0.7 = 0.60$ (the exact error is given in Section 4.3.3 with the optimal permutation). This example shows that this algorithm outputs a matrix $\boldsymbol{P}_p$ that may not be a permutation.

This index is always *optimistic* since it searches in a set of assignments larger than Perm(N). In fact, if a set $\mathcal{A}$ contains a set $\mathcal{B}$, *i.e.* $\mathcal{B} \subseteq \mathcal{A}$, then

$$\min_{x \in \mathcal{A}} f(x) \leq \min_{x \in \mathcal{B}} f(x) \tag{4.4}$$

for any function $f(x)$. Therefore, the reported error is always smaller than or equal to the exact error based on the optimal assignment.

4.3.1.2 Greedy approach of [CLDA09, CKAC14]

In order to avoid a non-acceptable assignment, after detecting the maximum value of each column of $\boldsymbol{C}$, its row and column can be removed for the rest of the algorithm. In other words, if in j^{th} column of matrix $\boldsymbol{C}$, C_{ij} is the maximum value, then the i^{th} row and j^{th} column of $\boldsymbol{C}$ will be ignored in the search of the next maximum value.

This is a *greedy* approach, since the index depends on the order of choosing the maximum values. For example, if this greedy algorithm is applied on matrix $\boldsymbol{C}$ expressed in (4.3), the resulted assignment will be $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_1)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$ provided that the columns are swept from left to right. However, if the columns are swept in the opposite way, the assignment will be $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_1)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$. Compared to the optimistic index, the error output by this greedy approach by sweeping from left to right, $3-0.85-0.3-0.7 = 1.15$, is larger than the exact error, $3-0.8-0.9-0.7 = 0.60$, while by sweeping from right to left, the reported error equals to the exact error 0.6.

By imposing a column ordering, this greedy approach searches a set of assignments smaller than Perm(N): following (4.4), one can conclude that the error measurement is always pessimistic. Therefore, the reported error is always larger than or equal to the exact error based on the optimal assignment.

4.3.1.3 Score measure [BBK18]

This index, which is also known as *congruence* [Ste09], is customized for tensors and is applied to evaluate the performance of a tensor decomposition in terms of estimating all the loading matrices (defined below) together. Let us explain the permutation ambiguity by means of a tensor decomposition example called Canonical Polyadic (CP) [Com14]. The CP decomposition of a third order tensor of rank 2 admits the following form:

$$\boldsymbol{\mathcal{T}}^{I\times J\times K} = \sum_{r=1}^{2} \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{a}_r^{(2)} \otimes \boldsymbol{a}_r^{(3)}, \tag{4.5}$$

where $\otimes$ denotes the outer (tensor) product, and $\boldsymbol{a}_r^{(1)}, \boldsymbol{a}_r^{(2)}$ and $\boldsymbol{a}_r^{(3)}$ are some vectors of size I, J and K, respectively. Equation (4.5) can be represented in a compact form as $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}]\!]$, where $\boldsymbol{A}^{(i)} = [\boldsymbol{a}_1^{(i)}, \boldsymbol{a}_2^{(i)}]$ is called the mode-i loading matrix of $\boldsymbol{\mathcal{T}}$. Observe that the permuted version of loading matrices, *i.e.* $\boldsymbol{A}_p^{(i)} = [\boldsymbol{a}_2^{(i)}, \boldsymbol{a}_1^{(i)}], i = 1, 2, 3$, results in the same tensor as $\boldsymbol{\mathcal{T}}$ in (4.5).

The score measure of the tensor $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}]\!]$ is calculated based on the correlation matrix $\boldsymbol{C} = \boldsymbol{C}^{(1)} \boxdot \boldsymbol{C}^{(2)} \boxdot \boldsymbol{C}^{(3)}$, where $\boxdot$ is the *Hadamard* product (element-wise product) and $C_{ij}^{(k)} \triangleq |\boldsymbol{a}_i^{(k)H} \hat{\boldsymbol{a}}_j^{(k)}|, k = 1, 2, 3$. This index is also greedy, since the assignment is concluded based on the maximum values of $\boldsymbol{C}$, which have been chosen in a way explained in Section 4.3.1.2, and the corresponding *score* is an average of these selected values.

4.3.2 Methods based on graph matching

The optimal assignment (or optimal transport) problem is an old, well-known and fundamental combinatorial optimization problem [Gal83, Mun57, PC$^+$19]. The first polynomial time algorithm for optimal assignment problems is the "Hungarian method" [Kuh55] also known as "Kuhn-Munkres" [Mun57, TK04], and the complexity of the algorithm is approximately N^4 flops [Mun57]. This algorithm has been employed in [TK04] for an optimal pairing of the sources in BSS.

The optimal assignment problem can also be considered as a special case of Maximum Weighted Matching (MWM), which is a well-known problem in graph theory, for which several polynomial time algorithms exist [Gal83]. The best exact [DP14, DK69, D$^+$59] and approximate [GT89] MWM algorithms cost approximately $8N^3$ and N^2 flops, respectively.

4.3.3 Methods based on optimal permutation

Searching for the optimal permutation σ, *i.e.* for the optimal permutation matrix $\boldsymbol{P}^\star$, described at the beginning of Section 4.3, can be viewed as finding some entries of $\boldsymbol{C}$ such that no pair among them lies in the same row or column, while the sum of these entries is maximum. One can formulate this as the following optimization problem [PC+19]:

$$\boldsymbol{P}^\star = \underset{\boldsymbol{P}_p \in \mathbb{R}_+^{N\times N}}{\operatorname{argmin}} \sum_{i,j} D_{ij} P_{p_{ij}} \quad \text{s.t.} \quad \boldsymbol{P}_p \mathbb{1}_N = \boldsymbol{P}_p^T \mathbb{1}_N = \mathbb{1}_N, \tag{4.6}$$

where $\boldsymbol{D} = -\boldsymbol{C}$, $\mathbb{1}_N$ is a vector of ones of dimension N and the superscript $\star$ denotes the optimal solution. In other words, we look for a *bistochastic* matrix, *i.e.* a square matrix of non-negative real numbers, whose rows and columns have unit L_1 norm [MOA79]. By vectorizing (concatenating columns) $\boldsymbol{P}_p$ and $\boldsymbol{D}$ into vectors $\boldsymbol{d}$ and $\boldsymbol{p}$, (4.6) can be rewritten in the standard form of linear program [PC+19, Sec. 3.1]:

$$\boldsymbol{p}^\star = \underset{\boldsymbol{p}\in\mathbb{R}_+^{N^2}}{\operatorname{argmin}} \boldsymbol{d}^H \boldsymbol{p} \quad \text{s.t.} \quad \boldsymbol{Q}\boldsymbol{p} = \mathbb{1}_{2N}, \tag{4.7}$$

where $\boldsymbol{Q} = [\mathbb{1}_N^T \boxtimes \boldsymbol{I}_N, \boldsymbol{I}_N \boxtimes \mathbb{1}_N^T]^T \in \mathbb{R}^{2N\times N^2}$, $\boldsymbol{I}_N$ and $\boxtimes$ denote the identity matrix of size N and the Kronecker product, respectively. Yet, from Birkhoff's Theorem, the set of bistochastic matrices is a polyhedron[1] whose vertices are permutations [HJ99, Theorem 8.7.1]. On the other hand, a fundamental theorem of linear programming [BT97, Theorem 2.7] states that the minimum of a linear objective in a non-empty polytope (*i.e.* a finite polyhedron) is reached at a vertex of the polytope. This permits to relax the search for a permutation into (4.6) or (4.7): in fact, looking for the best bistochastic matrix will eventually yield a permutation and this is the whole power of this method, which employs linear programming to estimate the permutation.

For example, by employing MWM or the linear program described above, the optimal permutation in experiment (4.3) is the identity matrix.

[1] convex by definition.

Recently, by improving some required computational steps, the running time of implementing linear programming algorithms has been reduced to approximately $q^{2+\frac{1}{6}}$ [CLS21] and $q^{2+\frac{1}{18}}$ [LSZ19] flops, where q is the size of unknown vector in the linear programming problem. Therefore, as $q = N^2$ in (4.7), the lowest complexity to find the optimal permutation of the problem described at the beginning of Section 4.3 by the means of linear programming is approximately N^4 flops. In addition, in [PC+19, KA21], authors proposed sub-optimal solutions, called Auction algorithms, which costs approximately N^2 flops [KA21].

4.3.4 Indices invariant to permutation

Methods described in Section 4.3.1 try first to estimate the permutation, and then measure some distances based on the estimated permutation. As it is seen in Section 4.3.1.1, these methods may actually not return a permutation, and there is no guarantee that $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$ even if the indices they output are zero. Conversely, the algorithms of Section 4.3.2 behave better (in terms of returning optimal permutation) but may become very costly for large values of N.

However, in the literature of source separation [CJ10], some indices have been proposed to measure the gap (based on a specific definition of gap) between original and estimated mixing matrices without inquiring to find the corresponding permutation [Com94, MM94, ACY+96]. Moreover, these indices are zero if and only if $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$, which offers a valuable guarantee.

The indices proposed in [Com94, MM94, ACY+96] are based on $\boldsymbol{S} = \boldsymbol{A}^{-1}\widehat{\boldsymbol{A}}$ (or $\boldsymbol{S} = \boldsymbol{A}^{\dagger}\widehat{\boldsymbol{A}}$ for non-square $\boldsymbol{A}$). The details of these indices are as follows.

4.3.4.1 Comon index [Com94]

Comon's index is a combination of L_1 and L_2 norms, and is calculated as:

$$\epsilon_1(\boldsymbol{S}) = \sum_{i=1}^{N}\left|\sum_{j=1}^{N}|S_{ij}|-1\right|^2 + \sum_{j=1}^{N}\left|\sum_{i=1}^{N}|S_{ij}|-1\right|^2, \\ \sum_{i=1}^{N}\left|\sum_{j=1}^{N}|S_{ij}|^2-1\right| + \sum_{j=1}^{N}\left|\sum_{i=1}^{N}|S_{ij}|^2-1\right|.$$

In [Com94], it has been proved that ϵ_1 is invariant to permutation, *i.e.* $\epsilon_1(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \epsilon_1(\boldsymbol{A}, \widehat{\boldsymbol{A}}\boldsymbol{P}_p\boldsymbol{\Lambda})$. Moreover, it has been shown that $\epsilon_1(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$ if and only if $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$, where σ is the optimal permutation. However, ϵ_1 can increase enormously, depending on the values of matrix $\boldsymbol{S}$, hence, this index is not bounded from above.

4.3.4.2 Moreau-Macchi index [MM94]

The index proposed in [MM94] measures a gap between matrix $\boldsymbol{S}$ and a permutation matrix. It is defined as:

$$\epsilon_2(\boldsymbol{S}) = \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|S_{ij}|^2}{(\max_k|S_{ik}|)^2}-1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|S_{ij}|^2}{(\max_k|S_{kj}|)^2}-1\right).$$

Dividing by the maximum value (*e.g.* $(\max_k|S_{ik}|)^2$) provides an upper bound for ϵ_2 unlike ϵ_1.

4.3.4.3 Amari index [ACY+96]

This performance index takes the form:

$$\epsilon_3(\boldsymbol{S}) = \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|S_{ij}|}{\max_k|S_{ik}|}-1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|S_{ij}|}{\max_k|S_{kj}|}-1\right).$$

The only difference between Amari and Moreau-Macchi index is the power 2, which exists in ϵ_2. Therefore, calculating ϵ_3 is less costly compared to ϵ_2. In addition, as for ϵ_2, the division by the maximum value (*e.g.* $\max_k|S_{ik}|$) provides an upper bound for ϵ_3.

An accurate investigation of indices reviewed in this section reveals that ϵ_1 is not bounded from above. Furthermore, the upper bounds on ϵ_2 and ϵ_3 have not been studied in [MM94, ACY+96], so that their upper bound cannot be easily interpreted. Even if one normalizes the Amari and Moreau-Macchi indices, the resulted upper bounds are reached when $\boldsymbol{A}$ has equivalent columns and $\widehat{\boldsymbol{A}}$ is identity matrix. Therefore, the upper bounds of the Amari and Moreau-Macchi indices do not correspond to the largest possible angular gap between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$.

In order to obtain interpretable upper bounds and to reduce computational cost, one may think of replacing $\boldsymbol{S} = \boldsymbol{A}^{\dagger}\widehat{\boldsymbol{A}}$ by $\boldsymbol{C}$, but in this case the property that $\epsilon_i = 0$ is equivalent to $\epsilon_0 = 0$, for $i \in \{1, 2, 3\}$ does not hold anymore.

4.4 Our proposed index: CorrIndex

In this section, we introduce, "*CorrIndex*", which is based on a correlation matrix. Reminding that we define $\boldsymbol{C} = |\boldsymbol{A}^H\widehat{\boldsymbol{A}}|$, where $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ and $\widehat{\boldsymbol{A}} \in \mathbb{C}^{M\times N}$ and modulus is applied entrywise. In addition, we assume that the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized by their L_2 norms.

Basically, if $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$, N entries of $\boldsymbol{C}$ are one, since $|\boldsymbol{a}_n| = |\hat{\boldsymbol{a}}_n|$ and the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized to unit L_2 norms. Remember that it is desired that a performance index is zero if and only if $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$. In order to satisfy these basic requirements in the matrix case, *i.e.* $M > 1$, CorrIndex is defined as follows:

$$\text{CorrIndex}(\boldsymbol{C}) = \frac{1}{2N}\left[\sum_{i=1}^{N} |\max_k C_{ik} - 1| + \sum_{j=1}^{N} |\max_k C_{kj} - 1|\right]. \qquad (4.8)$$

The coefficient $\dfrac{1}{2N}$ keeps CorrIndex values in the range $[0, 1]$. Moreover, modulus operator, *i.e.* $|.|$, guarantees a zero distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ if CorrIndex $= 0$ (cf. Proposition 2). In addition, according to (4.8), if the distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ is zero, then CorrIndex $= 0$. Therefore, the

two requirements mentioned above are simply satisfied by (4.8). Further, CorrIndex is invariant to permutation and scaling (cf. Proposition 1).

Remark: It can be also observed that CorrIndex is bounded: $0 \leq \text{CorrIndex} \leq 1$. According to (4.8), unlike ϵ_2 and ϵ_3, the upper bound of CorrIndex is easier to interpret when $M > 1$, since it is achieved when entries of $\boldsymbol{C}$ are minimal (*i.e.* the largest possible angular distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$). In particular, when $M \geq 2N$, $\boldsymbol{C} = \boldsymbol{0}$ when all the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are orthogonal to each other, which yields CorrIndex = 1. Next, as proved below, the zero lower bound is meaningful, since it corresponds to $\epsilon_0 = 0$.

The one-row case: On the other hand in the row vector case, *i.e.* $M = 1$, as CorrIndex is based on (4.2), we should return back to the basic minimization of finding optimal assignment (σ), which is a restatement of (4.2) as follows:

$$\underset{\sigma}{\operatorname{argmin}} \frac{1}{2} \sum_{n=1}^{N} (a_n - \hat{a}_{\sigma(n)})^2. \tag{4.9}$$

In order to satisfy (4.9) and to consider the scaling ambiguity, a new matrix $\boldsymbol{C}$ with entries $C_{ij} = (|a_i| - |\widehat{a}_j|)^2$ is used to define CorrIndex for row vectors:

$$\text{CorrIndex}(\boldsymbol{C}) = \frac{1}{2N} \left[\sum_{i=1}^{N} \min_k C_{ik} + \sum_{j=1}^{N} \min_k C_{kj} \right]. \tag{4.10}$$

Comparing (4.8) and (4.10) reveals that "max" and "1" have been replaced with "min" and "0", respectively, which helps benefit from the same properties as (4.8) in the vector case.

In the following, it is shown that CorrIndex is invariant to scaling and permutation, *i.e.* $\text{CorrIndex}(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \text{CorrIndex}(\boldsymbol{A}, \widehat{\boldsymbol{A}}\boldsymbol{P}_p\boldsymbol{\Lambda})$. Moreover, it is shown that $\text{CorrIndex}(\boldsymbol{C}) = 0$ if and only if $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$, *i.e.* $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$.

Proposition 1. *CorrIndex is invariant to permutation and scaling:*

$$\textit{CorrIndex}(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \textit{CorrIndex}(\boldsymbol{A}\boldsymbol{P}_p\boldsymbol{\Lambda}, \widehat{\boldsymbol{A}}) = \textit{CorrIndex}(\boldsymbol{A}, \widehat{\boldsymbol{A}}\boldsymbol{P}_p\boldsymbol{\Lambda}). \tag{4.11}$$

Proof. Assume that $\boldsymbol{C}_1 = |\boldsymbol{A}^H\widehat{\boldsymbol{A}}|$ and $\boldsymbol{C}_2 = |(\boldsymbol{A}\boldsymbol{P}_p\boldsymbol{\Lambda})^H\widehat{\boldsymbol{A}}|$. As modulus operator is insensitive to matrix $\boldsymbol{\Lambda}$, then $\boldsymbol{C}_2 = \boldsymbol{P}_p^H\boldsymbol{C}_1$, and since CorrIndex

is invariant to row permutation according to (4.8), the proof is complete. The same proof applies to $\boldsymbol{C}_3 = |\boldsymbol{A}^H \widehat{\boldsymbol{A}} \boldsymbol{P}_p \boldsymbol{\Lambda}|$, because of the invariance of CorrIndex to column permutation. □

Proposition 2. *Suppose that* $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ *and* $\widehat{\boldsymbol{A}} \in \mathbb{C}^{M \times N}$. *CorrIndex*$(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$ *if and only if* $\widehat{\boldsymbol{A}}$ *can be written as a permuted version of* $\boldsymbol{A}$*:*

$$\textit{CorrIndex}(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0 \iff \widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}. \tag{4.12}$$

Proof. Firstly, if $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$, then $\max_k C_{ik} = 1, \forall i$ and $\max_k C_{kj} = 1, \forall j$. Thus CorrIndex$(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$. Secondly, we prove the converse. If CorrIndex$(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$, then it implies that $\max_k C_{ik} = 1, \forall i$ and $\max_k C_{kj} = 1, \forall j$. From these two equalities, it can be inferred that there is at least one 1 in each column and row of $\boldsymbol{C}$. Let us assume $C_{ij} = |\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j| = 1$. According to the Cauchy–Schwarz inequality and the assumption of normalized columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$, we have $|\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j| \leq \|\boldsymbol{a}_i\| \|\hat{\boldsymbol{a}}_j\|$, where the equality of two sides occurs if and only if $\boldsymbol{a}_i = \hat{\boldsymbol{a}}_j$. Since such a conclusion holds for all other associated pairs of columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$, therefore, $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda}$. □

As mentioned before, it is hard to assess the relative error made on loading matrices in tensor decompositions, because of scaling and permutation ambiguities [Com14]. So as to overcome these ambiguities, we can use CorrIndex as a performance on estimating loading matrices. However, if we report CorrIndex on each loading matrix separately, it would be an optimistic index, since implicitly a different permutation would be permitted for each loading matrix. Note that according to (2.6), the corresponding columns of loading matrices construct a rank-1 tensor, hence to keep these rank-1 components unchanged, it is required to permute all loading matrices by the *same* permutation matrix. By permitting a different permutation for each loading matrix, the overall tensor reconstruction error could be less than the case in which all loading matrices have the same permutation.

In order to have a more reliable performance index, we can apply CorrIndex to a matrix $\boldsymbol{X}$, built upon loading matrices stacked one below the

other. In other words, for the tensor described in (4.5), CorrIndex$(\boldsymbol{X}, \widehat{\boldsymbol{X}})$ applies to:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \\ \boldsymbol{A}^{(3)} \end{bmatrix}, \quad \widehat{\boldsymbol{X}} = \begin{bmatrix} \widehat{\boldsymbol{A}^{(1)}} \\ \widehat{\boldsymbol{A}^{(2)}} \\ \widehat{\boldsymbol{A}^{(3)}} \end{bmatrix}.$$

4.5 Discussion and computer results

A multi-aspect comparison between CorrIndex and other reviewed methods has been carried out, and is reported in Table 4.1, where the methods of Section 4.3.1 and 4.3.2 are referred by "Greedy" and "Graph", respectively. The number of multiplications of each stage, *i.e.* computing the input matrix ($\boldsymbol{C} = |\boldsymbol{A}^H \widehat{\boldsymbol{A}}|$ or $\boldsymbol{S} = \boldsymbol{A}^\dagger \widehat{\boldsymbol{A}}$), estimating the permutation and computing the index, are reported. In addition, the last column of Table 4.1 ("Significance of upper bound") indicates if the upper bound makes sense, *i.e.* returning the maximum index value for the largest distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$. According to Table 4.1, it is inferred that CorrIndex has the lowest computational complexity compared to the others in terms of the number of multiplications besides its theoretical guarantee, its invariance to permutation and scaling ambiguity and its meaningful bounds.

In the rest of this section, we report either the relative error (for the greedy and graph-based methods), which is defined in (4.13) or the indices ϵ_i (Moreau-Macchi, Amari, ...). As greedy and graph-based methods estimate the permutation, we report the relative error between normalized $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$, by means of estimated matrix $\boldsymbol{P}_p$, as follows:

$$\text{relative error} = \frac{\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\boldsymbol{P}_p\|_F}{\|\boldsymbol{A}\|_F}, \tag{4.13}$$

where $\|.\|_F$ denotes the Frobenius norm. Reminding that $\boldsymbol{P}_p$ is aimed to be a permutation matrix, but that it might not be, and the inferred assignment of the columns by $\boldsymbol{P}_p$ does not make sense (see Section 4.3.1.1).

The index and computation time of each index in a numerical experiment is reported in Table 4.2 to evaluate the methods practically. This experiment

Table 4.1: Approximate numbers of multiplications of computing each stage of CorrIndex and other methods

Method	$\boldsymbol{C}$ or $\boldsymbol{S}$	Permutation	Index	Significance of upper bound
Greedy	N^2M	0	$2MN$	No
Graph	N^2M	$8N^3$	$2MN$	Yes
Linprog	N^2M	N^4	$2MN$	Yes
Comon [Com94]	$11N^3$	-	$2N^2$	No
Moreau-Macchi [MM94]	$11N^3$	-	$2N^2$	No
Amari [ACY$^+$96]	$11N^3$	-	$2N$	No
CorrIndex	N^2M	-	1	Yes

is executed on a laptop with a processor of 3.1 GHz Intel Core i5, 16 GB RAM, running macOS Mojave and MATLAB 2019a.

In order to show the drawbacks of greedy methods, this experiment is done on some matrices, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, whose columns are highly correlated. For this purpose, a correlation matrix, $\boldsymbol{R}^{N\times N}$, of the columns of $\boldsymbol{A}$ is designed such that its diagonal and off-diagonal entries are 1 and γ, respectively, where γ is an arbitrary mutual coherence constant among the columns of $\boldsymbol{A}$. Then, by considering the Cholesky decomposition of $\boldsymbol{R}$, *i.e.* $\boldsymbol{R} = \boldsymbol{L}^\mathsf{T}\boldsymbol{L}$, and a random orthogonal matrix $\boldsymbol{U}^{M\times N}$ ($\boldsymbol{U}$ can be obtained by the QR decomposition of a random matrix), we set $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{L}$. In this way, we have a matrix $\boldsymbol{A}$ with the columns having the correlation of γ.

$\widehat{\boldsymbol{A}}$ is generated by permuting randomly the columns of $\boldsymbol{A}$ and adding a noise matrix, $\boldsymbol{W}$, of the same size as $\boldsymbol{A}$ with i.i.d. entries of Gaussian distribution with zero mean and unit variance, and weighted by the parameter δ. The variance δ^2 of the additive noise is adjusted such that we reach a desired Signal to Noise Ratio (SNR) defined as:

$$\mathrm{SNR} = 10\log_{10}\frac{\sum_{i,j}\boldsymbol{A}(i,j)^2}{\sum_{i,j}\delta^2\boldsymbol{W}(i,j)^2}. \tag{4.14}$$

At the end, the columns of $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ are normalized.

Table 4.2: A numerical comparison on methods measuring the distance between $\boldsymbol{A}^{150\times100}$ with the mutual coherence constant $\gamma = 0.75$ and its permuted noisy version $\widehat{\boldsymbol{A}}$ with SNR $= -1.76$ dB averaged over 50 realizations. The index in five first rows of the table is the relative error defined in (4.13). On the other hand, the last four indices of the table are defined differently and are hence not comparable.

Method	Index	Computing time (ms)	Significance of upper bound
Greedy of [FIW+20]	0.37	0.9	No
Greedy of [CLDA09, CKAC14]	1.05	5.3	No
Hungarian [Mun57]	0.86	4.5	Yes
MWM [DP14]	0.86	2.9	Yes
Linprog [PC+19]	0.86	1310	Yes
Comon [Com94]	1.8e4	3.3	No
Moreau-Macchi [MM94]	897.91	3.3	No
Amari [ACY+96]	3.2e3	2.9	No
CorrIndex	0.36	0.4	Yes

In the experiment of Table 4.2, $M = 150, N = 100$, with the mutual coherence constant $\gamma = 0.75$, $\delta = 0.1$ (which is equivalent to SNR $= -1.76$ dB), and $\boldsymbol{U}$ is an orthogonal matrix obtained by concatenating the first N left-singular vectors of a random matrix whose entries are chosen randomly from a uniform distribution on $(0, 1)$. The reported values are averaged over 50 realizations.

The indices obtained by greedy methods and reported in Table 4.2 explicitly show the effect of coherence of input matrix on these types of methods. For instance, according to the performed experiment, greedy methods either report less (37%) or larger (105%) error than the exact index (86%). Note that as greedy methods try first to estimate permutation $\boldsymbol{P}_p$, and then calculate the error between matrices $\boldsymbol{A}\boldsymbol{P}_p$ and $\widehat{\boldsymbol{A}}$; hence, the indices computed by greedy methods may be compared to the exact error computed by graph-based methods. However, comparing other indices such as CorrIndex with the exact error does not make sense, since these indices are intrinsically

Table 4.3: A numerical comparison on methods measuring the distance between $\boldsymbol{A}^{150\times 100}$ with the mutual coherence constant $\gamma = 0.95$ and its permuted noiseless version $\widehat{\boldsymbol{A}}$ averaged over 50 realizations. The index in five first rows of the table is the relative error defined in (4.13). The four last indices of the table are defined differently and are hence not comparable.

Method	Index	Computing time (ms)	Significance of upper bound
Greedy of [FIW+20]	3.82e − 17	0.89	No
Greedy of [CLDA09, CKAC14]	0.10	5.2	No
Hungarian [Mun57]	0	2.7	Yes
MWM [DP14]	0	4.0	Yes
Linprog [PC+19]	0	1630	Yes
Comon [Com94]	2.14e − 13	3.3	No
Moreau-Macchi [MM94]	0	4.5	No
Amari [ACY+96]	3.35e − 16	3.8	No
CorrIndex	2.37e − 16	0.59	Yes

different from (4.13).

As can be seen in Table 4.2, CorrIndex is the fastest index. In addition, contrary to the indices of Comon, Moreau-Macchi and Amari, CorrIndex returns a value in the bounded range $[0, 1]$. One could normalize the Moreau-Macchi and Amari indices in order to obtain bounded values, but the signification of such upper bounds on these indices has never been investigated.

The experiment of Table 4.2 corresponds to an inaccurate estimation of $\boldsymbol{A}$ (*i.e.* SNR $= -1.76$), and all the performance indices (perhaps except the greedy method of [FIW+20], which yields 37%) demonstrate convincingly the fact that estimation is not accurate. In order to evaluate the indices in the opposite situation (accurate estimation), we perform another experiment with the same setting as that of Table 4.2 except that $\widehat{\boldsymbol{A}}$ is a permuted noiseless version of $\boldsymbol{A}^{150\times 100}$ with $\gamma = 0.95$. The result of this experiment is reported in Table 4.3.

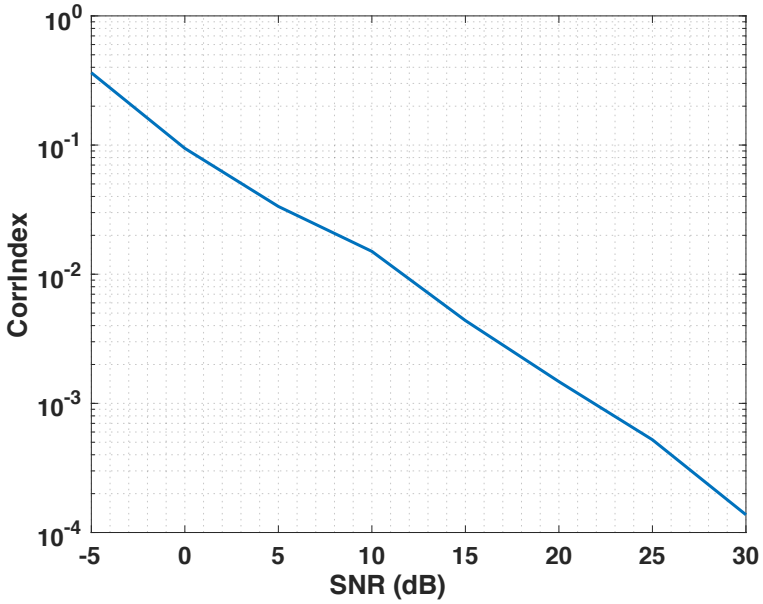

Figure 4.1: CorrIndex and noise. CorrIndex of a random matrix $\boldsymbol{A}^{6\times 4}$ and its permuted noisy version $\widehat{\boldsymbol{A}}$. This figure confirms the fact that the larger ϵ_0, the larger CorrIndex.

As it can be interpreted from the reported indices in Table 4.3, the greedy method of [CLDA09,CKAC14] does not report zero relative error between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$, which is not correct. However, in spite of highly correlated columns of $\boldsymbol{A}$, all indices (except the greedy method of [CLDA09,CKAC14]) demonstrate zero distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$, which is true. Comparing Table 4.2 and Table 4.3 reveals that the greedy methods are much more sensitive to the correlation of the columns than other indices.

CorrIndex is based on (4.2), which tries to minimize the least square error between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$. Therefore, if the distance between $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}$ increases due to the additive noise in $\widehat{\boldsymbol{A}}$, CorrIndex will return a larger value. To show this fact in practice, we performed an experiment whose result is depicted in Fig. 4.1. Generating a random matrix $\boldsymbol{A}$ of dimension 6×4, $\widehat{\boldsymbol{A}}$ is obtained by permuting its columns and by adding a noise matrix, $\boldsymbol{W}$, of the same size as $\boldsymbol{A}$ with independent and identically distributed (i.i.d.) entries of Gaussian distribution with zero mean and unit variance, and weighted by

the parameter δ. The variance δ^2 of the additive noise is adjusted such that we reach a desired SNR as described in (4.14).

Figure 4.1 confirms the fact that the larger ϵ_0, the larger CorrIndex. Therefore, in evaluating different decomposition methods, the one with the least CorrIndex would perform the best.

In Table 4.2, we show the effect of mutual coherence and noise on the results of each index. However, Fig. 4.2 investigates the effect of noise intensity, which is measured by SNR. In this experiment, as in Table 4.2, we generate a random matrix $\boldsymbol{A}$ of dimension 150×100 with the mutual coherence constant $\gamma = 0.75$ and averaged the results over 50 realizations. Then, the matrix $\widehat{\boldsymbol{A}}$ is obtained by permuting randomly the columns of $\boldsymbol{A}$ and by adding a noise matrix according to each SNR value. The goal of this experiment is to show the drawbacks of greedy methods, and to do this, we compare the result of the greedy methods of [FIW$^+$20, CLDA09, CKAC14] by one of the graph-based method (*i.e.* MWM [DP14]), which outputs the exact error. Therefore, we can simply conclude the inaccuracy of [FIW$^+$20, CLDA09, CKAC14].

Figure 4.2 shows the relative error (4.13) between $\widehat{\boldsymbol{A}}$ as an estimation of $\boldsymbol{A}$. As the error output by MWM is exact, the difference between errors output by the greedy methods of [FIW$^+$20, CLDA09, CKAC14] and the one by MWM show the inaccuracy of [FIW$^+$20, CLDA09, CKAC14]. As it is expected, the relative error by greedy method [FIW$^+$20] (resp. [CLDA09, CKAC14]) is optimistic (resp. pessimistic), since its reported error is smaller (resp. larger) than the exact error. In addition, as SNR increases, this error gets larger, which demonstrates that by decreasing the additive noise, the influence of mutual coherence becomes more effective on the result of greedy methods.

4.6 Conclusion

In this chapter, the problem of computing the distance between two matrices up to permutation and scaling ambiguities is addressed. This problem occurs

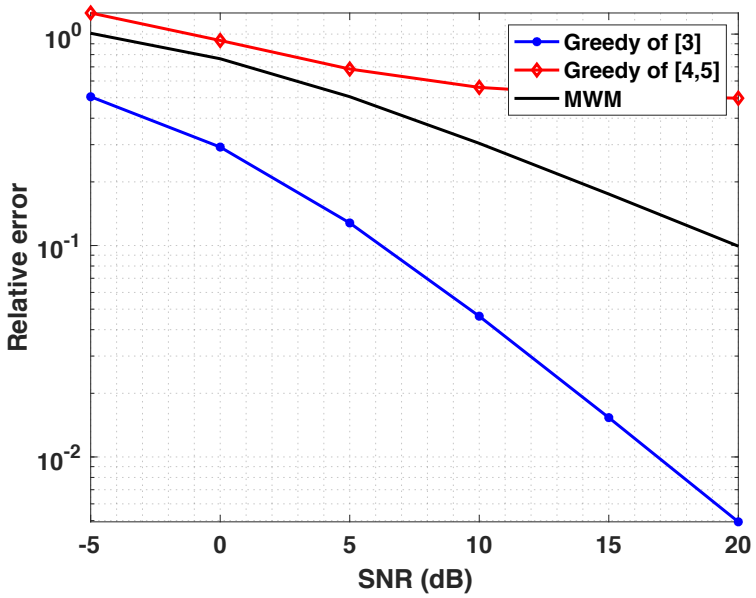

Figure 4.2: Drawbacks of greedy methods of [FIW$^+$20, CLDA09, CKAC14]. Compare the relative error between a random matrix $\boldsymbol{A}^{150\times 100}$ with the mutual coherence constant $\gamma = 0.75$ and its permuted noisy version $\widehat{\boldsymbol{A}}$ versus SNR reported by greedy methods of [FIW$^+$20, CLDA09, CKAC14] and by one of the exact indices, *i.e.* MWM averaged over 50 realizations.

for instance in tensor decompositions or in blind source separation. Existing performance indices are classified in three main categories: “greedy methods”, “graph-based methods” and “invariant indices”. These methods are reviewed, and it is inferred that greedy methods are not reliable especially in noisy situations (they are either optimistic or pessimistic). In addition, graph-based methods and invariant indices are computationally expensive. We propose a new performance index belonging to the class of invariant indices, named *CorrIndex*, whose upper and lower bounds are easy to interpret, while being computationally cheap. In the rest of this thesis, we will evaluate the performance of tensor decomposition methods by means of CorrIndex.

5 Constrained and non-constrained tensor decompositions

Contents

5.1 Introduction

In this chapter, we study constrained and non-constrained tensor decompositions, as well as introducing our own constrained tensor decomposition, Simple Forward-Backward Splitting (SFBS). As mentioned in Chapter 3, the main step in data mining by tensors is to decompose properly the tensor of third order moments. As we will show by simulation in this chapter, constrained tensor decomposition results in a more accurate and more reasonable (in the sense of being in the simplex set) estimation of probabilities of hidden and multi-view variables, which is the first step for some targeted data mining tasks such as unsupervised clustering. In addition, we review some non-constrained decompositions, which have been employed for this purpose, and will compare them experimentally and theoretically with constrained tensor decompositions.

Depending on the application, it is preferred to add some constraints to the tensor decomposition, which results normally in much more accurate and

reasonable solutions. Non-negativity, belonging to the simplex set, orthogonality and sparsity are some examples of key constraints often imposed in some applications such as medical image and signal processing [CZPA09], probability estimation in topic modeling (cf. Section 3.2) and dictionary learning [HLP14].

Generally, the algorithms of constrained tensor decomposition are inspired from constrained matrix decomposition (factorization). For instance, the algorithms mentioned in [CZPA09] for Non-negative Tensor Factorization (NTF) are extensions of Non-negative Matrix Factorization (NMF). Moreover, many of the existing algorithms are based on Alternating Optimization (AO) [SDLF$^+$17] or its special case, Alternating Least Squares (ALS) [CMDL$^+$15], in which the data fidelity term in AO is the least square error.

Over the past decade, some algorithms have been proposed for constrained tensor decomposition based on AO or ALS [HSL16,XY13b,VCTMM17], where in each step, a constrained minimization over one parameter is carried out. In each step, Alternating Direction Method of Multipliers (ADMM) [BPC$^+$11] or proximal methods [CP11] have been applied to solve the constrained minimization.

As our proposed method, SFBS, is based on the proximal concept, we limit ourself to review briefly the state-of-the-art algorithms utilizing proximal approaches. Therefore, in this chapter, after reviewing some preliminaries on the proximal concept in Section 5.2, some state-of-the-art algorithms for constrained tensor decomposition based on the proximal concept are reviewed in Section 5.3. In Section 5.5, we propose our algorithm, *i.e.* SFBS, and provide its convergence analysis. In Section 5.6, several experiments have been performed to compare our algorithm with constrained and unconstrained algorithms, in terms of performance. We also evaluate tensor decomposition methods on a real text data set, called "20 Newsgroups". In Section 5.7, we discuss briefly about advantages and drawbacks of each method based on the simulations and experiments in Section 5.6. Finally,

Section 5.8 concludes the chapter along with explaining some future perspectives.

5.2 Proximal concept and approach

In this section, we explain the proximal concept, operators and methods such as Forward-Backward Splitting, which is used in our proposed method in Section 5.5. Firstly, some required definitions and properties of functions are reviewed as follows:

* **Lower semi-continuity (lsc) [RW09, Definition 1.5]:** Suppose $\overline{\mathbb{R}} = [-\infty, +\infty]$. The function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ is called *lower semi-continuous* (lsc) on $\mathbb{R}^n$ if

$$\lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} [\inf_{\boldsymbol{x} \in \mathbb{B}(\overline{\boldsymbol{x}}, \epsilon)} f(\boldsymbol{x})] = f(\overline{\boldsymbol{x}}) \tag{5.1}$$

holds for every $\overline{\boldsymbol{x}} \in \mathbb{R}^n$, where $\mathbb{B}(\overline{\boldsymbol{x}}, \epsilon)$ is the closed ball:

$$\mathbb{B}(\overline{\boldsymbol{x}}, \epsilon) \stackrel{\mathsf{def}}{=} \{\boldsymbol{x} | d_e(\boldsymbol{x}, \overline{\boldsymbol{x}}) \leq \epsilon\}, \tag{5.2}$$

in which $d_e(\boldsymbol{x}, \overline{\boldsymbol{x}})$ is the Euclidean distance. See Fig. 5.1 for an example of a lower semi-continuous function.

* **Sub-gradient [RW09, Definition 8.3]:** Suppose $\overline{\mathbb{R}} = [-\infty, +\infty]$. For a function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ and a point $\overline{\boldsymbol{x}}$ for which $f(\overline{\boldsymbol{x}})$ is finite, a vector $\boldsymbol{v} \in \mathbb{R}^n$ is said to be the sub-gradient of f at $\overline{\boldsymbol{x}}$, *i.e.* $\boldsymbol{v} \in \partial f(\overline{\boldsymbol{x}})$, if $f(\boldsymbol{x}) \geq f(\overline{\boldsymbol{x}}) + \langle \boldsymbol{v}, \boldsymbol{x} - \overline{\boldsymbol{x}} \rangle + \mathrm{O}(|\boldsymbol{x} - \overline{\boldsymbol{x}}|)$, where $\langle ., . \rangle$ and $\mathrm{O}(.)$ denote the scalar (inner) product and limiting behavior of the remainder term[1]. Note that ∂ is the typical notation of sub-gradient, and it is replaced with ∇ for differentiable functions[2].

[1] For two arbitrary functions $p(.)$ and $q(.)$, we say $p(\boldsymbol{x}) = \mathrm{O}(q(\boldsymbol{x}))$ as $\boldsymbol{x} \to \overline{\boldsymbol{x}}$, if there exist positive values δ and M such that for all $0 < |\boldsymbol{x} - \overline{\boldsymbol{x}}| < \delta$, $|p(\boldsymbol{x})| < Mq(\boldsymbol{x})$.

[2] A differentiable function has a single sub-gradient at each point.

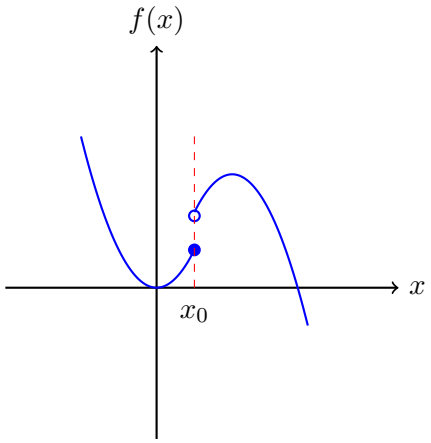

Figure 5.1: An example of a lower semi-continuous (lsc) function. The solid blue dot indicates $f(x_0)$.

* **Kurdyka-Lojasiewicz property (KL) [ABS13]:** Let us denote the domain of a function f by $\text{dom}(f)$. The function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ has the Kurdyka-Lojasiewicz property at $\boldsymbol{x}^* \in \text{dom}(f)$ if there exist $\eta \in (0, +\infty]$, a neighborhood U of $\boldsymbol{x}^*$ and a continuous concave function $\varphi : [0, \eta) \mapsto \mathbb{R}_+$ such that:

  1) $\varphi(0) = 0$,

  2) φ is differentiable on $(0, \eta)$,

  3) $\varphi'(y) \geq 0$ for all $y \in (0, \eta)$,

  4) The Kurdyka-Lojasiewicz inequality,

$$\varphi'(f(\boldsymbol{x}) - f(\boldsymbol{x}^*))\text{dist}(0, \partial f(\boldsymbol{x})) \geq 1,$$

  holds for all $\boldsymbol{x} \in U \cap \{\boldsymbol{x} | f(\boldsymbol{x}^*) < f(\boldsymbol{x}) < f(\boldsymbol{x}^*) + \eta\}$, where dist(.) denotes the distance function. In the rest of the report, this property is referred to as "KL", in short.

In the context of optimization, the KL property is important, since many problems include functions satisfying this property. As it is explained in Remark 2, many semi-algebraic (cf. Remark 2 for definition of semi-algebraic) functions have the KL property, however,

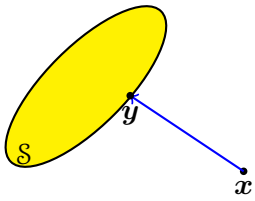

Figure 5.2: The projection of a vector $\boldsymbol{x} \in \mathbb{R}^N$ onto a closed convex set $\mathcal{S} \subset \mathbb{R}^N$

there are many other functions in real world problems, which are not semi-algebraic but satisfy the Kurdyka-Lojasiewicz inequality.

* **Graph of a function [XY13b, Section 2.2]** The graph of the function $f : \mathbb{R}^d \mapsto (-\infty, +\infty]$ is denoted by $\mathrm{Gr}(f) \in \mathbb{R}^{d+1}$ and is defined as $\mathrm{Gr}(f) \stackrel{\mathsf{def}}{=} \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in \mathrm{dom}(f)\}$.

In the rest of this section, the proximal concept and approaches are reviewed.

5.2.1 Proximity operator

The projection of a vector $\boldsymbol{x} \in \mathbb{R}^N$ onto a closed convex set $\mathcal{S} \subset \mathbb{R}^N$ is a classical problem in signal processing, which is depicted in Fig. 5.2 and can be formulated as [CZ+97, Com93, YW82]:

$$\mathrm{proj}_{\mathcal{S}}(\boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathbb{R}^N}{\mathrm{argmin}} \left\{ i_{\mathcal{S}}(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right\}, \tag{5.3}$$

where $i_{\mathcal{S}}$ is the indicator function defined by:

$$i_{\mathcal{S}}(\boldsymbol{y}) \triangleq \begin{cases} 0 & if \quad \boldsymbol{y} \in \mathcal{S} \\ \infty & if \quad \boldsymbol{y} \notin \mathcal{S} \end{cases} \tag{5.4}$$

Let $\Gamma_0(\mathbb{R}^N)$ be the class of lower semi-continuous functions $f : \mathbb{R}^N \mapsto (-\infty, +\infty]$, with $\mathrm{dom}(f) \neq \varnothing$. Then $i_{\mathcal{S}}$ belongs to $\Gamma_0(\mathbb{R}^N)$.

According to the proposition of Moreau in 1962 [Mor62], the definition of *Proximity operator* is obtained by replacing $i_{\mathcal{S}}(\boldsymbol{y})$ in (5.3) with any arbitrary function in $\Gamma_0(\mathbb{R}^N)$:

Proximity operator [CP11] For every $\boldsymbol{x} \in \mathbb{R}^N$, the unique solution of the following minimization problem:

$$\underset{\boldsymbol{y}\in\mathbb{R}^N}{\operatorname{argmin}}\, f(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2, \tag{5.5}$$

is defined as the *proximity operator* of the function $f \in \Gamma_0(\mathbb{R}^N)$, and it is denoted by $\text{prox}_f(\boldsymbol{x})$. The term $\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ is also called *proximal regularization* in the literature [HSL16, XY13b]. Thus, the proximity operator of f is $\text{prox}_f : \mathbb{R}^N \mapsto \mathbb{R}^N$, and it is characterized by:

$$\boldsymbol{p} = \text{prox}_f(\boldsymbol{x}) \Leftrightarrow (\boldsymbol{x}-\boldsymbol{p}) \in \partial f(\boldsymbol{p}), \quad \forall (\boldsymbol{x},\boldsymbol{p}) \in \mathbb{R}^N \times \mathbb{R}^N.$$

Note that $\partial f(\boldsymbol{p})$ is replaced by $\nabla f(\boldsymbol{p})$ for differentiable f.

The above definition indicates that $\text{prox}_f(\boldsymbol{x})$ is a point that minimizes f and simultaneously is as close as possible to $\boldsymbol{x}$. Therefore, $\text{prox}_f(\boldsymbol{x})$ is also called a *proximal point of* $\boldsymbol{x}$ *with respect to* f [PB14]. See Table 10.2 in [CP11] for a list of popular functions and their corresponding proximity operators. However for the sake of clarity, Table 5.1 provides proximity operators of some functions, which are discussed in this thesis.

5.2.2 Proximal methods

Although proximal methods have first appeared in the work of Martinet [Mar70] in 1970, they have only been utilized in signal processing since 2001 [CW05], and their applications in various fields are getting more and more prevalent. As mentioned before, a proximity operator can be viewed as a generalized projection [PB14]. In addition to projection, there are several interpretations, some of which we review from [PB14, CP11]:

- **The approximation of gradient step:**
  If f is twice differentiable at $\boldsymbol{x}$ and its Hessian matrix is positive definite, then it can be shown that as $\lambda \to 0$:

$$\text{prox}_{\lambda f}(\boldsymbol{x}) = \boldsymbol{x} - \lambda\nabla f(\boldsymbol{x}) + \text{O}(\lambda).$$

Therefore, for small λ, $\text{prox}_{\lambda f}(\boldsymbol{x})$ converges to a gradient step in f with step length λ. Thus, the proximity operator can be interpreted as an approximation of a gradient step for minimizing f provided that λ is small enough. This fact shows a connection between proximity operators and gradient methods, and consequently the proximal operator could be useful in optimization. It can be also expected that λ will have a similar effect as a step size in a gradient method.

- **Fixed-point of $\text{prox}_{\lambda f}$:**

  It can be proved that the fixed points of the proximity operator of f are exactly the minimizers of f [PB14, sec 2.3], *i.e.* $\text{prox}_{\lambda f}(\boldsymbol{x}^*) = \boldsymbol{x}^*$ if and only if $\boldsymbol{x}^*$ minimizes f. This also reveals a close connection between proximity operators and fixed point theory. Therefore, fixed points of appropriate operators (proximity operators) can be utilized as a solution for optimization problems. This interpretation will be used in Theorem 1 of Section 5.2.3.

- **Denoising:**

  A proximity operator can also be interpreted as a solution for denoising in signal processing applications. Consider a noisy observation $\boldsymbol{y} \in \mathbb{R}^N$ of a signal $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{w}$, where $\boldsymbol{w}$ models the additive noise. The best recovery of the original signal $\boldsymbol{x}$ is usually formulated as:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{argmin}} \left\{ f(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \right\}, \tag{5.6}$$

  where $\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2$ is the data fidelity term and $f(\boldsymbol{x})$ takes into account a prior knowledge about $\boldsymbol{x}$. According to the definition of proximity operator, (5.6) is exactly equal to $\text{prox}_f(y)$.

5.2.3 Forward-Backward Splitting

In many signal processing applications, the cost function to be minimized is the sum of two functions where one of them is usually non-differentiable or even non-convex. By following a proximal approach, these kinds of problems

can be solved by means of a particular algorithm called *Forward-Backward Splitting*. Theorem 1 [CP11] explains the relation of minimizing such a cost function with this algorithm.

Theorem 1 (Forward-Backward Splitting [CP11]). *Suppose* $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ *is a proper*[3] *lower semi-continuous function which has the KL property and is bounded from below. If* f *can be split into two parts as* $f = h+g$*, where* g *is lower semi-continuous and* $h : \mathbb{R}^N \mapsto \mathbb{R}$ *is a finite valued, differentiable function with a* β*-Lipschitz continuous gradient,* i.e., $\exists\beta$ *such that:*

$$\|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\|_2 \leq \beta\|\boldsymbol{x} - \boldsymbol{y}\|_2,$$

then it can be shown [CW05] that the minimizer of f *satisfies the following fixed point equation:*

$$\boldsymbol{x} = prox_{\gamma g}(\boldsymbol{x} - \gamma\nabla h(\boldsymbol{x})), \tag{5.7}$$

where $\gamma \in (0, +\infty)$.

Equation (5.7) suggests an iterative approach, called the *Forward-Backward Splitting* algorithm:

$$\boldsymbol{x}_{k+1} = \text{prox}_{\gamma g}(\boldsymbol{x}_k - \gamma_k\nabla h(\boldsymbol{x}_k)), \tag{5.8}$$

where the values of γ_k should be chosen from a suitable bounded interval.

Forward-Backward Splitting is a combination of two basic methods: a proximal algorithm and a gradient approach [CP11]. Actually, if $g = 0$, then (5.8) is transformed to the gradient method of the differentiable function h. On the other hand, when $h = 0$, (5.8) is the proximity operator of g, and the minimizer of f is the fixed point of this operator.

Several variations of implementing Forward-Backward Splitting exist and are reported in [CP11]. Two of them are restated (Algorithm 2 and Algorithm 3) to which we will refer in the rest of this thesis.

In Algorithm 2, several parameters are required to be set by user, such as a relaxation parameter, α_k, which cannot exceed 1. Algorithm 2 directly

[3] A function is proper, if its domain is not a null set.

Algorithm 2 Forward-Backward Splitting [CW05, CP11, Algorithm 10.5]

Input: The function $f = h + g$ as defined in Theorem 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Output: The minimizer of f

1: Fix $\epsilon \in (0, \min\{1, \frac{1}{\beta}\})$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$
4: $\quad \boldsymbol{y}_k = \boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)$
5: $\quad \alpha_k \in [\epsilon, 1]$
6: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\text{prox}_{\gamma_k g}(\boldsymbol{y}_k) - \boldsymbol{x}_k)$
7: **end for**

implements the iterative approach described in (5.8), except doing an extrapolation with α_k in line 6 between the new point obtained from applying the proximity operator and the previous estimation point ($\boldsymbol{x}_k$).

Algorithm 3 is based on Fast Iterative Shrinkage Thresholding Algorithm (FISTA) proposed in [BT09a, BT09b] and can be considered as a proximal gradient algorithm. In fact, instead of shrinkage function in FISTA, the proximity operator is used.

The main difference between Algorithm 2 and Algorithm 3 is that in Algorithm 3 the proximity operator is not employed on the previous iterate $\boldsymbol{x}_k$ but rather on a very specific linear combination of the previous two iterates, $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$ [BT09b].

Although Algorithm 3 is more user-friendly than Algorithm 2 in terms of the number of required parameters to be set, the computation complexity of Algorithm 2 is less than Algorithm 3 due to not calculating some coefficients like t_k and λ_k in lines 5 and 6 of Algorithm 3.

In many applications (including SFBS in Section 5.5), the function g is an indicator function of a particular set ($\mathcal{S}$), $i_{\mathcal{S}}$, and its proximity operator is a projection onto that set [CP11, Table 10.2]. If the desired set is non-convex, the projection onto it may not result to a unique point. It has been proved [ABS13] that in spite of the multi-valued projection, the convergence

Algorithm 3 Beck-Teboulle proximal gradient algorithm [CP11, Algorithm 10.7] based on FISTA [BT09b]

Input: The function $f = h + g$ as defined in Theorem 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Output: The minimizer of f

Set $\boldsymbol{z}_0 = \boldsymbol{x}_0$ and $t_0 = 1$
for $k = 0, 1, 2, \ldots$ **do**
    $\boldsymbol{y}_k = \boldsymbol{z}_k - \beta^{-1}\nabla h(\boldsymbol{z}_k)$
    $\boldsymbol{x}_{k+1} = \mathrm{prox}_{\beta^{-1}g}(\boldsymbol{y}_k)$
    $t_{k+1} = \dfrac{1 + \sqrt{4t_k^2 + 1}}{2}$
    $\lambda_k = 1 + \dfrac{t_k - 1}{t_{k+1}}$
    $\boldsymbol{z}_{k+1} = \boldsymbol{x}_k + \lambda_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$
end for

property of Theorem 1 is not influenced. Note that this interesting conclusion is valid only if the assumptions of Theorem 1 are satisfied, the most important being the KL property satisfied by $h + i_{\mathbb{S}}$.

5.3 Constrained algorithms based on the proximal concept

In this section, some algorithms that use proximal approaches for constrained (mostly non-negativity) CP decomposition are reviewed. Our proposed method (SFBS in Section 5.5) will be compared to all of them in Section 5.6.1.

5.3.1 Problem formulation

The problem of N order constrained CP decomposition can be expressed as the following minimization:

$$\min_{\boldsymbol{\lambda},\boldsymbol{A}^{(1)},\ldots,\boldsymbol{A}^{(N)}} \frac{1}{2}\|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2$$
$$\text{s.t. } \boldsymbol{\mathcal{C}}_{\boldsymbol{\lambda}}, \boldsymbol{\mathcal{C}}_{\boldsymbol{A}^{(n)}},\ 1 \leq n \leq N$$

where $\boldsymbol{\mathcal{T}}$ is a N order tensor of dimension $n_1 \times n_2 \times \ldots \times n_N$ and rank R with the coefficient vector $\boldsymbol{\lambda}_{R\times 1}$ and N loading matrices $\boldsymbol{A}^{(1)}_{n1\times R}, \boldsymbol{A}^{(2)}_{n2\times R}, \ldots, \boldsymbol{A}^{(N)}_{nN\times R}$. In addition, $\boldsymbol{\mathcal{C}}_{\boldsymbol{\lambda}}, \boldsymbol{\mathcal{C}}_{\boldsymbol{A}^{(n)}}$ represent the constraint over the coefficient vector $\boldsymbol{\lambda}$ and loading matrix $\boldsymbol{A}^{(n)}$, respectively.

Since minimizing the cost function of the CP decomposition over all loading matrices is a non-convex problem, a common strategy to transform it to a sequence of convex problems (if constraints are already convex) is Alternating Optimization (AO) framework [SDLF$^+$17] or Block Coordinate Descent (BCD) [Tse01]. In AO, by fixing all the loading matrices (by initialization or using their previous estimation) except one of them, one tries to minimize the cost function over just one loading matrix.

For example, ALS steps for a third order constrained tensor decomposition of dimensions $n_1 \times n_2 \times n_3$ and rank R, $\boldsymbol{\mathcal{T}} = [\![\mathbb{1}_R; \boldsymbol{A}^{(1)}_{n_1\times R}, \boldsymbol{A}^{(2)}_{n_2\times R}, \boldsymbol{A}^{(3)}_{n_3\times R}]\!]$, in an unfolding form, are as follows:

$$\begin{aligned}
\boldsymbol{A}^{(1)}_{k+1} &= \underset{\boldsymbol{A}_{n_1\times R}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(1)}_{n_1\times n_2 n_3} - \boldsymbol{A}(\boldsymbol{A}^{(3)}_k \odot \boldsymbol{A}^{(2)}_k)^T\|_F^2 \text{ s.t. } \mathcal{C}_{\boldsymbol{A}}(\boldsymbol{A}) \\
\boldsymbol{A}^{(2)}_{k+1} &= \underset{\boldsymbol{B}_{n_2\times R}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(2)}_{n_2\times n_1 n_3} - \boldsymbol{B}(\boldsymbol{A}^{(3)}_k \odot \boldsymbol{A}^{(1)}_{k+1})^T\|_F^2 \text{ s.t. } \mathcal{C}_{\boldsymbol{B}}(\boldsymbol{B}) \\
\boldsymbol{A}^{(3)}_{k+1} &= \underset{\boldsymbol{C}_{n_3\times R}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(3)}_{n_3\times n_1 n_2} - \boldsymbol{C}(\boldsymbol{A}^{(2)}_{k+1} \odot \boldsymbol{A}^{(1)}_{k+1})^T\|_F^2 \text{ s.t. } \mathcal{C}_{\boldsymbol{C}}(\boldsymbol{C})
\end{aligned} \tag{5.9}$$

where $\mathcal{C}.(\cdot)$ represents the desired constraint over loading matrices, and it can be added to its corresponding fidelity term in the form of *generalized* regularization, *e.g.* an indicator function as defined in (5.4).

Non-negativity is a relevant constraint that has been employed in the literature, but in this thesis, we discuss also other useful constraints such as

simplex set and sparsity.

5.3.2 Alternating Optimization-Alternating Direction Method of Multipliers (AO-ADMM) [HSL16]

As mentioned in [PB14, CP11], Alternating Direction Method of Multipliers (ADMM) [BPC$^+$11] can be considered as a special case of the proximal method. In [HSL16], a constrained CP decomposition by means of ADMM is discussed.

AO-ADMM attempts to minimize each step of (5.9) by ADMM. The details of AO-ADMM for constrained CP decomposition can be found in [HSL16]. The proposed algorithm in [HSL16] is capable of applying several constraints on loading matrices, such as non-negativity, sparsity, smoothness, cardinality, etc.

The convergence of AO is briefly reviewed in [HSL16] based on the works of [Tse01, RHL13]. Although it is mentioned that adding a proximal regularization (cf. (5.5)) provides a *unique* solution which serves the convergence of BCD, this regularization is omitted in the formulation of AO-ADMM for the sake of convenience. The convergence of each step which is an ADMM algorithm can be shown for convex functions [HSL16, BPC$^+$11] and also recently for some non-convex functions such as $\ell_q, \quad 0 < q < 1$ [WYZ19], but not yet for some others such as ℓ_0. Note that ℓ_q norm of the vector $\boldsymbol{x}$ of dimension n for $q > 0$ is defined as $(\sum_{i=1}^{n} |x_i|^q)^{\frac{1}{q}}$; however $\ell_0(\boldsymbol{x})$ pseudo-norm is the number of the non-zero elements of $\boldsymbol{x}$.

5.3.3 Alternating Proximal Gradient (APG) [XY13b]

In [XY13b], the convergence of a general BCD algorithm is studied, where for updating each block of unknown variables, three kinds of updates are investigated, namely *original*, *proximal* and *prox-linear*. In two of them, proximal regularization is considered. This general BCD algorithm can be applied on a vast variety of constrained optimization, including constrained

tensor decomposition.

Based on the comparison done in [XY13b], prox-linear updating (i) yields better objective values than original and proximal updating, (ii) is easier to compute and (iii) often allows closed form solutions.

APG does not explicitly follow Forward-Backward Splitting procedure, nevertheless its algorithm for non-negativity constraint ends in projection to non-negative orthant along with updating a particular coefficient as in line 5 of Algorithm 3.

The convergence analysis in [XY13b] reveals that BCD with original updating requires strong convexity to provide a particular convergence guarantee[4], while by proximal and prox-linear updating, the global convergence to a critical point is achieved under mild conditions such as satisfying KL inequality or Lipschitz continuity.

5.3.4 Fast Non-negative Tensor Factorization-APG (FastNTF-APG) [ZZZ+16]

FastNTF-APG [ZZZ+16] is a modified version of APG [XY13b] dedicated to non-negative tensor decomposition. In [ZZZ+16], it is mentioned that, contrary to classical algorithms of Non-negative Tensor Factorization (NTF), which suffer from slow convergence especially in practical applications, FastNTF-APG speeds up NTF and overcomes this bottleneck by combining APG with the low rank approximation.

As APG, FastNTF-APG utilizes prox-linear updating, and its idea to speed up convergence can be explained by following each step of (5.9). Without loss of generality, let us explain the idea motivating the first step. In order to reduce the computational complexity, matrix $\boldsymbol{\mathcal{T}}^{(1)}$ in $\boldsymbol{\mathcal{T}}^{(1)}(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T$ is replaced with its low rank approximation. This is also efficient for filtering the noise out of $\boldsymbol{\mathcal{T}}^{(1)}$.

As in APG, it is also needed in FastNTF-APG to compute the Lipschitz

[4]Subsequence converging to a Nash point [XY13b, Equation 2.3]

constant of the gradient of $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(1)} - \boldsymbol{A}(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T\|_F^2$ w.r.t. $\boldsymbol{A}$ (considering again the update of the first loading matrix, to fix the ideas). It is mentioned in [ZZZ+16] that the Lipschitz constant is $\|(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})\|_F$, but as we prove in Appendix A, the Lipschitz constant is in fact the *spectral norm*[5] of $\{(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})\}$.

Let us remark a point about the importance of the Lipschitz constant of the gradient of the fidelity terms in (5.9). To employ the Forward-Backward Splitting algorithm for minimizing the objective functions in (5.9), the fidelity term (*e.g.* $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(1)} - \boldsymbol{A}(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T\|_F^2$) roles as the function "h" in Theorem 1. In Theorem 1, "β" is the Lipschitz constant of the gradient of "h". On the other hand, "γ" is the coefficient of ∇h in the fixed point equation (5.7) in the Forward-Backward Splitting algorithm (cf. Theorem 1). As we shall see in Section 5.5.4 (cf. Theorem 2), by choosing $\gamma \in [0, \frac{1}{\beta}]$, the convergence of Theorem 1 is guaranteed. Therefore, computing "β" (or the Lipschitz constant of the gradient of the fidelity term) has a crucial role in the convergence of algorithms, which are based on the Forward-Backward Splitting algorithm.

5.3.5 Block Coordinate Variable Metric Forward-Backward (BC-VMFB) [CPR16, VCTMM17, VCTM+17]

BC-VMFB consists of two main steps: a gradient step related to the data fidelity, which is assumed to be differentiable and has a β-Lipschitz gradient, and a proximal step linked to the regularization term, for which a new proximity operator should be calculated. This proximity operator of the function φ is associated with a symmetric positive definite matrix $\boldsymbol{Z}$ by the following definition [VCTMM17]:

$$\text{prox}_{\boldsymbol{Z},\varphi}(\boldsymbol{v}) = \underset{\boldsymbol{u}}{\text{argmin}} \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_{\boldsymbol{Z}}^2 + \varphi(\boldsymbol{v})$$

where $\|\boldsymbol{x}\|_{\boldsymbol{Z}}^2 = \langle \boldsymbol{x}, \boldsymbol{Z}\boldsymbol{x} \rangle$, and $\langle \cdot, \cdot \rangle$ is the inner product. In the above definition of the proximity operator, $\boldsymbol{Z}$ is called *preconditioning* matrix [VCTMM17].

[5]The spectral norm of a matrix is defined as its maximum singular value [Ber05].

The definition of the proximity operator given in (5.2) may be obtained from the above definition, if $\boldsymbol{Z}$ is the identity matrix. It is observed empirically in [CMLZ18] that utilizing preconditioning matrix speeds up the convergence of Proximal Alternating Linearized Minimization (PALM) [BST14].

Let us bring two points about non-negative tensor decomposition by means of BC-VMFB. Firstly, the convergence of BC-VMFB is studied in [CPR16], and it is shown that if the cost function satisfies the KL property, then its convergence to a critical point is guaranteed. Secondly, the algorithm proposed in [VCTMM17] is designed to handle vectors, so each loading matrix has to be vectorized before utilization.

5.4 Non-constrained algorithms for symmetric tensor decompositions

As mentioned in Section 3.3.1.1, in order to estimate the probabilities of hidden variables and the conditional probabilities of multi-view variables (which are denoted by $\boldsymbol{\varphi}$ and $\boldsymbol{A}$, respectively), it is required to decompose the third order moments tensor ($\boldsymbol{\mathcal{T}}$). The focus of this section is on the non-constrained decomposition for this purpose. Remind that the third order moments ($\boldsymbol{\mathcal{T}}$) and the second order moments ($\boldsymbol{P}$) are related to the desired probabilities according to (3.6) and (3.5), which are worth to be mentioned again:

$$\boldsymbol{P} = \sum_{k=1}^{K} \varphi_k \, \boldsymbol{a}_k \otimes \boldsymbol{a}_k$$

$$\boldsymbol{\mathcal{T}} = \sum_{k=1}^{K} \varphi_k \, \boldsymbol{a}_k \otimes \boldsymbol{a}_k \otimes \boldsymbol{a}_k.$$

5.4.1 Robust tensor power method [AGH+14]

We describe in this section the approach of [AGH+14], called "Robust tensor power method", whose drawbacks we shall investigate experimentally in Sec-

tion 5.6.2. This method of decomposition is dedicated to symmetric tensors and to the problem described in Section 3.3, where the empirical estimation of the second order moments is also available in addition to the tensor of third order moments. In [AGH+14], the authors propose to use the two moments ($\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$) defined in (3.3) and (3.4) with exactly the same encoding explained in Section 3.3.1.1. Therefore, these moments satisfy (3.5) and (3.6).

Matrix $\boldsymbol{P}$ is theoretically positive semi-definite, since $\boldsymbol{P}$ is a covariance matrix (note that φ_k are non-negative numbers in (3.5)). Hence in a similar way as had been done for Blind Source Separation [CJ10, Com94], there exists a "whitening" matrix $\boldsymbol{W}$ such that $\boldsymbol{W}^\mathsf{T}\boldsymbol{P}\boldsymbol{W} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix; $\boldsymbol{W}$ can theoretically be easily obtained from the EigenValue Decomposition (EVD) $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\mathsf{T}$, $\boldsymbol{U}^\mathsf{T}\boldsymbol{U} = \boldsymbol{I}$, by setting $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{D}^{-1/2}$. Therefore,

$$\sum_{k=1}^{K} \tilde{\boldsymbol{a}}_k \tilde{\boldsymbol{a}}_k^\mathsf{T} = \boldsymbol{I},$$

in which $\tilde{\boldsymbol{a}}_k \overset{\mathsf{def}}{=} \sqrt{\varphi_k}\, \boldsymbol{W}^\mathsf{T}\boldsymbol{a}_k$. In other words, the matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{K\times K}$ containing $\tilde{\boldsymbol{a}}_k$ as its columns is orthogonal, *i.e.* $\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{A}}^\mathsf{T} = \boldsymbol{I}$.

As mentioned in Section 3.4, in practice, the matrix $\boldsymbol{P}$ can be estimated via a particular encoding of multi-view variables $\boldsymbol{x}_\ell$ and a moment estimator for approximating (3.3), and consequently, $\boldsymbol{P}$ may have negative eigenvalues (of course it is not the case with the encoding explained in Section 3.3.1.1 and estimators described in Section 3.4) because of estimation errors. This issue was not investigated in [AGH+14], because they used left singular vectors of $\boldsymbol{P}$ instead of eigenvectors to construct $\boldsymbol{W}$ (cf. Algorithm 4). Nevertheless, the issue still remains, since left and right singular vectors are not imposed to be the same. Next, this whitening matrix is applied to the tensor $\boldsymbol{\mathcal{T}}$ to yield:

$$\widetilde{\boldsymbol{\mathcal{T}}} \overset{\mathsf{def}}{=} \boldsymbol{\mathcal{T}} \underset{1}{\bullet} \boldsymbol{W} \underset{2}{\bullet} \boldsymbol{W} \underset{3}{\bullet} \boldsymbol{W},$$

which have the following element-wise definition according to (2.2) [Com14]:

$$\widetilde{\boldsymbol{\mathcal{T}}}(r,t,s) = \sum_{r',t',s'} \boldsymbol{\mathcal{T}}(r',t',s')\boldsymbol{W}(r,r')\boldsymbol{W}(t,t')\boldsymbol{W}(s,s').$$

This new tensor satisfies

$$\widetilde{\boldsymbol{\mathcal{T}}} = \sum_{k=1}^{K} \varphi_k^{-1/2}\ \tilde{\boldsymbol{a}}_k \otimes \tilde{\boldsymbol{a}}_k \otimes \tilde{\boldsymbol{a}}_k.$$

The conclusion is that $\widetilde{\boldsymbol{\mathcal{T}}}$ ideally admits an *orthogonal* CP decomposition; see *e.g.* [Com14, Com94] for an introduction. Many algorithms have been devised for this kind of decomposition, including the pair-sweeping CoM algorithm, or Joint Approximate Diagonalization (JAD) algorithms [CJ10]. But authors in [AGH$^+$14] utilized the tensor power iteration [DLC$^+$95] to extract the dominant "eigenvector[6]", $\hat{\boldsymbol{a}}$, and the dominant "eigenvalue", $\widehat{\varphi}$, of $\widetilde{\boldsymbol{\mathcal{T}}}$ and then proceeded by deflation, *i.e.* $\widetilde{\boldsymbol{\mathcal{T}}} \leftarrow \widetilde{\boldsymbol{\mathcal{T}}} - \widehat{\varphi}\ \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}}$, to get the remaining ones.

Unfortunately, except for the tensor power iteration [DLC$^+$95], the pseudo code in [AGH$^+$14] is not complete in terms of describing the entire algorithm. Our description in Algorithm 4 hopefully fills this lack. We used this algorithm in our subsequent computer experiments in Section 5.6.2.

We shall show experimentally in Section 5.6.2 that in spite of its name, Robust tensor power method is not robust to the additive noise. Actually, robustness in [AGH$^+$14] is considered as robustness to the initial point of eigenvector (cf. line 4 of Algorithm 4). It is proved that any initial vector, close enough to the solution, will converge to an eigenvector of $\widetilde{\boldsymbol{\mathcal{T}}}$

Shifted Symmetric Higher-Order Power Method (SS-HOPM) [KM11] Robust tensor power method [AGH$^+$14] is based on Symmetric Higher-Order Power Method (S-HOPM), which is a rank-1 approximation and tries to find

[6]Recall that there exist several definitions of tensor eigenvectors [Lim05, QL17]. The definition used in [AGH$^+$14] – and hence here – is $\boldsymbol{\mathcal{T}} \bullet \boldsymbol{v} \bullet \boldsymbol{v} = \lambda \boldsymbol{v}$, which has the undesirable property that λ depends on the norm of $\boldsymbol{v}$. In fact, if $(\lambda, \boldsymbol{v})$ is an eigenpair, then so is $(\alpha\lambda, \alpha\boldsymbol{v})$ for any nonzero α. This is analyzed in *e.g.* [QL17].

Algorithm 4 Robust tensor power method [AGH+14]

Input: The empirical estimation of moments, *i.e.* $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$

Output: The estimation of desired probabilities, *i.e.* $\boldsymbol{A}$ and $\boldsymbol{\varphi}$

Whitening: $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$; $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}^{-1/2}$; $\widetilde{\boldsymbol{\mathcal{T}}} \overset{\mathsf{def}}{=} \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{W} \bullet_2 \boldsymbol{W} \bullet_3 \boldsymbol{W}$

for $k = 1, \ldots, K$ **do**

  for $m = 1, \ldots, M_1$ **do**

    1) draw an initial value for vector of unit 2-norm for $\hat{\boldsymbol{a}}_k$.

    2) compute M_2 power iteration updates in norm 2, *i.e.*:

    $\boldsymbol{t} = \widetilde{\boldsymbol{\mathcal{T}}} \bullet_2 \hat{\boldsymbol{a}}_k \bullet_3 \hat{\boldsymbol{a}}_k$; $\widehat{\varphi_k} \leftarrow \|\boldsymbol{t}\|_2$; $\hat{\boldsymbol{a}}_k \leftarrow \dfrac{\boldsymbol{t}}{\widehat{\varphi}_k}$.

  end for

  Pick the trial (m) having the largest $\widehat{\varphi}_k$.

  Refine $\hat{\boldsymbol{a}}_k, \widehat{\varphi}_k$ by M_2 extraneous iterations.

  Deflation:$\widetilde{\boldsymbol{\mathcal{T}}} \leftarrow \widetilde{\boldsymbol{\mathcal{T}}} - \widehat{\varphi}\, \hat{\boldsymbol{a}}_k \otimes \hat{\boldsymbol{a}}_k \otimes \hat{\boldsymbol{a}}_k$

end for

De-whitening: $\boldsymbol{B} = (\boldsymbol{W}^T)^{\dagger}$; $\boldsymbol{a}_k = \widehat{\varphi}_k \boldsymbol{B} \hat{\boldsymbol{a}}_k$; $\varphi_k = \dfrac{1}{\widehat{\varphi}_k^2}$ for $k = 1, \ldots, K$

Back to norm 1: $\alpha = \|\boldsymbol{a}_k\|_1$; $\boldsymbol{a}_k \leftarrow \dfrac{\boldsymbol{a}_k}{\alpha}$; $\varphi_k \leftarrow \dfrac{\varphi_k}{\alpha}$ for $k = 1, \ldots, K$ (our suggestion)

the largest eigenvalue of a symmetric tensor [DLC+95]. Robust tensor power method generalizes S-HOPM to a rank-K approximation by performing deflation after estimating each of K eigenvalues. S-HOPM does not have the guarantee of convergence, unless for an even-order tensor whose corresponding cost function is convex. Shifted Symmetric Higher-Order Power Method (SS-HOPM) improves the convergence of S-HOPM by adding an additive adjustable term, called *shift*, to the cost function. Although SS-HOPM like S-HOPM requires also the order of tensor to be even for its convergence analysis, it relaxes the convexity assumption on the cost function.

5.4.2 Singular Value based Tensor Decomposition (SVTD) [RCG18]

In spite of other methods described heretofore, Singular Value based Tensor Decomposition (SVTD) is based on matrix decomposition rather than tensor decomposition. In addition, unlike Power method and its variants which utilize the eigenvalues and the eigenvectors of the whitened third order moments, SVTD employs its singular values and singular vectors. Therefore, as Robust tensor power method, SVTD also requires the second order moments for whitening the third order moments. Additionally, the first order moments (or simply the average of multi-view variables) is also needed in SVTD to estimate $\boldsymbol{\varphi}$.

SVTD can be viewed as the joint diagonalization of the second order moments ($\boldsymbol{P}$) and a particular slice of the whitened third order moments ($\boldsymbol{\mathcal{T}}$); see [RCG18, Algorithm 1] for much more details. The authors of [RCG18] mentioned that SVTD has some advantages over Robust tensor power method in terms of executing time and required memory space. However, the experiments in [RCG18] reveal that its performance is the same as that of Robust tensor power method.

5.5 Proposed Tensor Decomposition Scheme: Simple Forward-Backward Splitting (SFBS)

In this section, we describe SFBS, the method we propose for constrained (*e.g.* non-negativity) CP decomposition. First, in Section 5.5.1, the cost function, its solution by means of Forward-Backward Splitting and SFBS algorithm, which is based on Algorithm 2 are introduced. Then, some prevalent constraints such as non-negativity and the simplex space along with required modifications in SFBS algorithm for these particular constraints

are discussed in Section 5.5.2. In Section 5.5.4, the convergence theorem for SFBS is stated, whereas some required details of calculations can be found in Appendix A.

5.5.1 Formulation and algorithm

In section 2.3.1, CP decomposition and some constraints are described. Let us now formulate *constrained* CP decomposition and some prevalent constraints in this section.

Consider the N-th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ of rank R. Assume that $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, where $\boldsymbol{\lambda} \in \mathbb{R}_+^R$ and $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$. A general problem of the constrained CP decomposition of $\boldsymbol{\mathcal{T}}$ can be formulated as follows:

$$\min_{\boldsymbol{\lambda}, \boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2 \tag{5.10}$$
$$\text{s.t. } \mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)}),\ 1 \leq n \leq N,$$

where $\mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)})$ are, respectively the constraints on the vector $\boldsymbol{\lambda}$ (including the above mentioned constraint, *i.e.* $\boldsymbol{\lambda} \in \mathbb{R}_+^R$, such as belonging to the simplex set) and the matrix $\boldsymbol{A}^{(n)}$.

As mentioned in Section 5.3.2, a common strategy is to solve (5.10) via ALS. Moreover, a constrained optimization can be transformed into an unconstrained one by adding the indicator function of the constraint set to the cost function. To be more precise, at the n^{th} step of ALS for solving (5.10), we have:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \big\| \boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(n)}, \ldots, \boldsymbol{A}^{(N)}]\!] \big\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}), \tag{5.11}$$

where $i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)})$ is defined as follows:

$$i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}) = \begin{cases} 0 & if \quad \boldsymbol{A}^{(n)} \in \mathcal{C}_{\boldsymbol{A}^{(n)}} \\ \infty & if \quad \boldsymbol{A}^{(n)} \notin \mathcal{C}_{\boldsymbol{A}^{(n)}} \end{cases}.$$

The vector $\boldsymbol{\lambda}$ is omitted in (5.11), since it can be calculated by normalizing loading matrices $(\boldsymbol{A}^{(n)})$. Otherwise, the vector $\boldsymbol{\lambda}$ as one of the unknown variables can be optimized in one of the steps of ALS.

Define $\boldsymbol{W} \triangleq (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$. Then by the mode-$n$ unfolding of (5.11), we have:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}). \tag{5.12}$$

Observe that $i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)})$ is a lower semi-continuous function (cf. the definition of lsc in page 70) for many prevalent constraints such as non-negativity and the simplex set (see Section 5.5.2 for the definition of these constraints) and $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$ is finite valued[7], differentiable and β-Lipschitz continuous gradient where $\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$ denotes the spectral norm of matrix $\boldsymbol{W}\boldsymbol{W}^T$ (see Appendix A for calculations). As mentioned before, the spectral norm of a matrix is equal to its maximum singular value. Since $\boldsymbol{W}\boldsymbol{W}^T$ is a symmetric matrix, its singular values are the squared of those of $\boldsymbol{W}$. Moreover, the cost function in (5.12) is proper, lower semi-continuous with the KL property [ABS13] (see Section 5.5.4 for more explanation). Consequently, all the required assumptions of Theorem 1 are satisfied for (5.12), and according to this theorem, the minimizer of (5.12) is the convergence point of the following fixed point equation:

$$\boldsymbol{A}^{(n)} = \text{prox}_{\gamma i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}}\{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T})\}.$$

Since the proximity operator of $\gamma i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}$ is the projection onto $\mathcal{C}_{\boldsymbol{A}^{(n)}}$, we have:

$$\boldsymbol{A}^{(n)} = \text{proj}_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}\{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T})\}. \tag{5.13}$$

SFBS is described in Algorithm 5. Although Algorithm 5 is based upon Algorithm 2, we experimentally find the proper values of some parameters of Algorithm 2 such as γ_k and α_k to obtain the best results. As it is expressed in Algorithm 5, we ignore ϵ and fix the value of $\gamma_k = \frac{e}{\beta}$ in all iterations over k (we experimentally observed that $e = 1.4$ or $e = 1.9$ are almost always proper values). In addition, we remove the effect of α_g by setting it to 1 in the linear updating of the estimated variable (Line 8 in Algorithm 5).

[7] Being finite-value is not boundedness, but it means that a function takes values in the real line, *i.e.* $(-\infty, +\infty)$.

Algorithm 5 The algorithm of SFBS

Input: $\boldsymbol{\mathcal{T}}$, $\mathcal{C}_{\boldsymbol{A}^{(n)}}$, initial $\boldsymbol{A}_0^{(n)}$, $n \in [1, \ldots, N]$, e

Output: Estimated $\boldsymbol{A}^{(n)}$, $n \in [1, \ldots, N]$

repeat
  for $n = 1, 2, \ldots, N$ **do**
    $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$
    $\beta = \{\max(\text{singular value}(\boldsymbol{W}))\}^2$
    set $\gamma = \frac{e}{\beta}$ and choose α_g.
    for $g = 0, 1, 2, \ldots$ **do**
      $\boldsymbol{Y} = \boldsymbol{A}_g^{(n)} - \gamma(\boldsymbol{A}_g^{(n)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{\mathcal{T}}^{(n)}\boldsymbol{W}^T)$
      $\boldsymbol{A}_{g+1}^{(n)} = \boldsymbol{A}_g^{(n)} + \alpha_g(\text{proj}_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{Y}) - \boldsymbol{A}_g^{(n)})$
    end for
  end for
until some termination criterion

As SFBS, compared to the previous algorithms, requires less parameters to be set (especially compared to APG, which needs a calculation of a coefficient in each iteration), and also it is easy enough to understand and implement, we call it *Simple* Forward-Backward Splitting (SFBS). Although detailed descriptions about the advantages of SFBS compared to other methods (discussed in this thesis) are provided in Section 5.7, we list them below in brief to complete the current section:

- Unlike other constrained methods, it is explained that how SFBS can handle a variety of constraints such as the simplex set on all loading factors along with a coefficient vector, $\boldsymbol{\lambda}$,

- Unlike Robust tensor power method, SFBS is capable of handling over-complete cases (when the number of hidden variables, K, is larger than the number of multi-view ones, D) in estimating probabilities (cf. Section 3.3.1),

- Compared to APG, as one of the most efficient methods based on

proximal concept, SFBS performs better in noisy scenario in terms of relative reconstruction error as well as estimating loading factors,

- Unlike AO-ADMM, a complete convergence analysis is provided in Section 5.5.4.

5.5.2 Some constraints

As mentioned before, the SFBS is not limited to any particular constraint. Any constraint whose indicator function satisfies the assumptions of Theorem 1 can be considered. In this section, the required modifications of Algorithm 5 are described, for some widespread constraints, all of which are lsc (cf. the definition of lsc in page 70).

5.5.2.1 Non-negativity

Non-negativity is one of the most common constraints in the literature. In many applications such as image processing [CZPA09] or chemometrics [LC09], the non-negativity of loading matrices is essential and helpful in the performance of tensor decomposition. The non-negativity constraints of (5.10) are expressed as: $\boldsymbol{A}^{(n)} \in \mathbb{R}_{+}^{I_n \times R}$ and $\boldsymbol{\lambda} \in \mathbb{R}_{+}^{R}$.

The projection to the non-negative orthant is done with the max operator, thereby the line 8 of Algorithm 5 would be $\boldsymbol{A}_{g+1}^{(n)} = \boldsymbol{A}_{g}^{(n)} + \alpha_g(\max(\boldsymbol{Y}, 0) - \boldsymbol{A}_{g}^{(n)})$. In other words, the proximity operator of the non-negativity constraint retains the non-negative elements of the array and replaces its negative values with zero.

As mentioned in Section 3.3.1.1, in order to estimate the probabilities of hidden variables and the conditional probabilities of multi-view variables (which are denoted by $\boldsymbol{\varphi}$ and $\boldsymbol{A}$, respectively), it is required to decompose the third order moments tensor ($\boldsymbol{\mathcal{T}}$). One possibility is to apply constrained tensor decompositions such as SFBS. Although it is better to consider $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ belonging to a simplex set, decomposing the third order moments tensor under the non-negativity constraint of $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ would be acceptable enough.

Hence, one can use SFBS with the non-negativity constraint to estimate the desired probabilities in the problem described in Section 3.3.

5.5.2.2 Simplex set

In the applications involving the probability estimation or the distribution approximation, the simplex set (or the probability simplex) constraint unavoidably appears (cf. Section 3.3.1.1). A vector $\boldsymbol{x} \in \mathbb{R}^P$ belongs to the simplex set $\boldsymbol{\mathcal{S}}^P$, if $\{\boldsymbol{x}^T\mathbf{1} = 1,\ x_i \geq 0,\ i \in [1, \ldots, P]\}$. This constraint can be written as $\{\boldsymbol{A}^{(n)}(:, j) \in \boldsymbol{\mathcal{S}}^{I_n},\ j \in [1, \ldots, R],\ n \in [1, \ldots, N]\}$ and $\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}^R$.

An algorithm for projecting a vector onto the simplex set is proposed in [Con16]. Therefore, the line 8 of Algorithm 5 would include a projection algorithm to the simplex, whose input is $\boldsymbol{Y}$.

An implementation trick. As simplex set is a crucial constraint in text mining application (estimating probabilities), here we propose a practical trick to handle this constraint over all parameters (the coefficient vector and all loading matrices).

An efficient way to apply the simplex constraint to all the columns of loading matrices $\boldsymbol{A}^{(n)}, n \in [1, \ldots, N]$ and to the coefficient vector $\boldsymbol{\lambda}$ is to first combine $\boldsymbol{\lambda}$ with one of the loading matrices, let us say $\boldsymbol{A}^{(N)}$. By combination, we mean $\boldsymbol{A}^{(N)}_{\boldsymbol{\lambda}} = \boldsymbol{A}^{(N)}\text{Diag}(\boldsymbol{\lambda})$, where $\text{Diag}(\boldsymbol{\lambda})$ is a diagonal matrix containing $\boldsymbol{\lambda}$ on its diagonal. Then, the simplex constraint may be applied to every column of every matrix $\boldsymbol{A}^{(n)}, n \in [1, \ldots, N-1]$, and only to the vectorization of matrix $\boldsymbol{A}^{(N)}_{\boldsymbol{\lambda}}$. Each entry λ_r of $\boldsymbol{\lambda}$ is eventually obtained by normalizing the rth column of matrix $\boldsymbol{A}^{(N)}_{\boldsymbol{\lambda}}$ with respect to ℓ_1 norm, and the resulting normalized matrix yields an estimation of $\boldsymbol{A}^{(N)}$. It can then be proved easily that the estimated $\boldsymbol{\lambda}$ and every column of loading matrix $\boldsymbol{A}^{(N)}$ indeed lie in the simplex set.

As mentioned before, constrained tensor decompositions such as SFBS is a reasonable choice to estimate the desired probabilities in the problem described in Section 3.3. Since we seek the probabilities, it is expected that $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ belong to a simplex set. Following the instructions of ap-

plying the simplex set constraint described above, one can utilize SFBS under the simplex set constraint to decompose the third order moments tensor. If we denote the obtained coefficient vector and the loading factors by $\widehat{\boldsymbol{\lambda}}, \widehat{\boldsymbol{A}^{(1)}}, \widehat{\boldsymbol{A}^{(2)}}$ and $\widehat{\boldsymbol{A}^{(3)}}$, the estimated $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ can be considered simply as $\widehat{\boldsymbol{\lambda}}$ and $\dfrac{\widehat{\boldsymbol{A}^{(1)}} + \widehat{\boldsymbol{A}^{(2)}} + \widehat{\boldsymbol{A}^{(3)}}}{3}$, respectively.

5.5.2.3 Sparsity with ℓ_0 pseudo-norm (cardinality)

In some applications such as two-dimensional dictionary learning [HLP14] and co-clustering [PS11], a constraint on the number of the non-zero elements of loading matrices (or cardinality) is needed. This constraint is measured with ℓ_0 pseudo-norm and known as *sparsity* in compressive sensing [Don06].

In the penalized form, an ℓ_0 pseudo-norm term is added to the cost function [Tib96]. Although ℓ_0 is non-convex, the resulted cost function can be solved by means of Theorem 1. As mentioned before, contrary to AO-ADMM, the convergence of algorithms based on Theorem 1, including Algorithm 5, is guaranteed for the ℓ_0 pseudo-norm constraint.

In order to use Algorithm 5 with a cardinality constraint, the proximity operator of ℓ_0 should be calculated. It is mentioned in [ABS13, PB14] that the proximity operator of ℓ_0 is a function called *hard-thresholding*, defined as follows:

$$f_\eta^h(y) \triangleq \begin{cases} y & |y| > \eta \\ 0 & |y| \leq \eta \end{cases}. \tag{5.14}$$

To be more precise, according to (5.5) the proximity operator of ℓ_0 is defined as:

$$\operatorname{prox}_{\gamma\|\cdot\|_0}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \gamma\|\boldsymbol{y}\|_0, \tag{5.15}$$

which is equal to $[f^h_{\sqrt{2\gamma}}(x_i)]_{i=1}^N$ (a vector of size N whose elements are $f^h_{\sqrt{2\gamma}}(x_i)$ for $i \in [1, \ldots, N]$). With this in mind, the line 8 of Algorithm 5 includes applying hard-thresholding on each element of $\boldsymbol{Y}$ ($[f^h_{\sqrt{2\gamma}}(Y_{(i,j)})]_{(i=1,j=1)}^{(I_n,R)}$).

Table 5.1: Proximity operator of functions used in this thesis

Function	**Proximity operator**
Indicator function of $\mathcal{S}$ ($i_{\mathcal{S}}$ (5.4))	Projection onto $\mathcal{S}$ ($\text{proj}_{\mathcal{S}}$ (5.3))
ℓ_0 pseudo-norm (cardinality)	hard-thresholding (5.14)
ℓ_1 norm	soft-thresholding (5.16)

5.5.2.4 Sparsity with ℓ_1 norm

A common and convex approximation of the ℓ_0 pseudo-norm is ℓ_1 norm. The resulting cost function obtained by replacing ℓ_0 in (5.15) with ℓ_1 norm is called *LASSO regression* [Tib96, Don06].

In order to minimize LASSO with Algorithm 5, the proximity operator of ℓ_1 should be computed. In [PB14], it is stated that according to (5.5), the proximity operator of $\ell_1(\boldsymbol{x})$ is $f^s_{\sqrt{2\gamma}}$, which should be applied element-wise on $\boldsymbol{x}$. f^s_η is called *soft-thresholding*, and is defined as:

$$f^s_\eta(y) \triangleq \begin{cases} y-\eta & y>\eta \\ 0 & |y|\leq\eta \\ y+\eta & y<-\eta \end{cases} . \tag{5.16}$$

As for ℓ_0, $f^s_{\sqrt{2\gamma}}$ should be applied on each element of $\boldsymbol{Y}$ in the line 8 of Algorithm 5, instead of a projection.

5.5.3 Constrained CP decomposition based on Proximal Forward-Backward Splitting without AO [NMSC21]

Although all the algorithms discussed so far, including SFBS, are based on AO, some other constrained decompositions such as our work in [NMSC21] update all the loading factors together. To be more precise, in AO framework, one tries to update one of the loading factors in each step, while the other loading factors are constant (either by initialization or by updating from previous iterations), the proposed algorithm in [NMSC21] tries to estimate all the loading factors, simultaneously. To do this, firstly, all loading

factors are concatenated into a one variable matrix, like $\boldsymbol{X}$ defined in (5.22). Then, Forward-Backward Splitting is applied to minimize a constrained objective over the vectorization of $\boldsymbol{X}$. For instance, in our work [NMSC21], the considered constraint for CP decomposition is the coherence of the columns of loading factors.

5.5.4 Convergence guarantee

In this section, we analyse the convergence of the SFBS algorithm. SFBS, like other methods reviewed in Section 5.3, utilizes BCD or AO (to be exact, ALS). Therefore, the convergence analysis is firstly studied for AO in Section 5.5.4.1, and then, the convergence guarantee of each step (or block) of AO (or BCD) is investigated in Section 5.5.4.2.

5.5.4.1 The convergence of AO with Proximal Minimization

The overall convergence of AO when the minimization in each step is carried out by means of the proximal concept is proved in [XY13b,BST14]. Since the convergence analysis provided in [BST14] is suitable for the algorithm based on Proximal Forward-Backward, we refer the overall convergence of SFBS (which is also based on Proximal Forward-Backward) to the analysis performed in [BST14], namely, Proximal Alternating Linearized Minimization (PALM).

Let us rewrite (5.10) in an unconstrained manner by means of constraints regularization (*i.e.* indicator functions or ℓ_p norms), and mention some new notations, which help us to discuss the required assumptions of PALM more efficiently:

$$\Psi(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) = f_d(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) + \sum_{n=1}^{N} r_n(\boldsymbol{A}^{(n)}) + r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \tag{5.17}$$

where $f_d(.)$ is the fidelity term as follows:

$$f_d(\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}) = \frac{1}{2} \| \boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!] \|_F^2,$$

and $r_n(.)$ or $r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda})$ could be indicator functions or ℓ_p norm, depending on the constraints on $\boldsymbol{A}^{(n)}$ and $\boldsymbol{\lambda}$.

In the sequel, the required assumptions of PALM are reviewed, and we shall explain how SFBS, which follows PALM procedure, satisfies these assumptions by considering (5.17). Let us first bring up some required points through our explanation over assumptions.

Based on the definition of KL on page 71, some classes of functions that satisfy the KL property are summarized in [XY13b, Section 2.2], among which we are interested in a particular class of functions, namely *semi-algebraic*. The exact definition of semi-algebraic functions and some of their properties can be found in Remark 2, which will be practical in the explanation over the assumptions of PALM.

Remark 2. *A set $\mathcal{D} \subset \mathbb{R}^d$ is called semi-algebraic if it can be represented as follows [XY13b, Section 2.2]*

$$\mathcal{D} = \bigcup_{i=1}^{s} \bigcap_{j=1}^{t} \{\boldsymbol{x} \in \mathbb{R}^d | p_{ij}(\boldsymbol{x}) = 0, q_{ij}(\boldsymbol{x}) > 0\}, \tag{5.18}$$

where p_{ij}, q_{ij} are polynomial functions. A function is semi-algebraic if its graph (see page 72 for definition) is a semi-algebraic set.

Some of the elementary properties of semi-algebraic functions are noted below:

1- *Semi-algebraic functions satisfy the KL property.*

2- *The sum of semi-algebraic functions is semi-algebraic.*

3- *The sum of a polynomial and a semi-algebraic function is semi-algebraic.*

4- *The indicator functions of a polyhedral set (such as a non-negative set and the probability simplex) is semi-algebraic.*

5- *$\ell_1, \ell_2, \ell_\infty$ norms are semi-algebraic.*

6- *The sum of ℓ_0 pseudo-norm and a polynomial is semi-algebraic [ABS13, Example 5.4].*

The assumptions in [BST14] are mentioned as several parts in three categories, which are listed below:

Assumption 1 (i): $r_n(.)$ and $r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda})$ are proper and lsc: since the domains of $r_n(.)$ and $r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda})$ are not null, these functions are proper, in addition, as mentioned before, indicator functions and ℓ_p norm are lsc according to the definition of lsc on page 70.

(ii): f_d is a C^1 function (the class of functions with first order differentiability): since f_d is quadratic, therefore it is a C^2 function, and hence it is a C^1 function.

Assumption 2 (i): $\inf \Psi(.) > -\infty, \inf r_n(.) > -\infty, \inf r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}) > -\infty$: these assumptions are satisfied, since $f_d \geq 0$, $r_n(.) \geq 0, r_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}) \geq 0$.

(ii): f_d is globally Lipschitz (see below for a detailed definition): since f_d is quadratic, it is globally Lipschitz.

Globally Lipschitz [BST14, Assumption 2 (ii)]. *For any fixed y, the function $x \to G(x, y)$ is $C^{1,1}_{L_1(y)}$, namely the partial gradient $\nabla_x G(x, y)$ is globally Lipschitz with moduli $L_1(y)$, that is*

$$\|\nabla_x G(x_1, y) - \nabla_x G(x_2, y)\| \leq L_1(y) \|x_1 - x_2\|, \ \forall x_1, x_2 \in \mathbb{R}^n$$

Likewise, for any fixed x, the function $y \to G(x, y)$ is $C^{1,1}_{L_2(x)}$.

(iii): The following inequalities are trivially fulfilled, since Assumption 2 (ii) is satisfied (cf. [BST14, Remark 3 (iii)]).

Inequalities 3.5 and 3.6 from [BST14]. For $i = 1, 2$, there exists $\lambda_i^-, \lambda_i^+ > 0$, such that

$$\begin{aligned} &\inf\{L_1(y^j) : j \in \mathbb{N}\} \geq \lambda_1^- \quad \text{and} \quad \inf\{L_2(x^j) : j \in \mathbb{N}\} \geq \lambda_2^- \\ &\sup\{L_1(y^j) : j \in \mathbb{N}\} \geq \lambda_1^+ \quad \text{and} \quad \sup\{L_2(x^j) : j \in \mathbb{N}\} \geq \lambda_2^+, \end{aligned}$$

where j is the index denoting a PALM iteration.

(iv): The following inequality is satisfied for f_d, whenever f_d is a C^2 function (cf. [BST14, Remark 3 (iv)]): since f_d is quadratic, it is a C^2 function.

Inequality 3.7 from [BST14]. ∇G is Lipschitz continuous on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$. In other words, for each bounded subset $B_1 \times B_2$ of $\mathbb{R}^n \times \mathbb{R}^m$, there exists $Z > 0$ such that for all $(x_i, y_i) \in B_1 \times B_2, i = 1, 2$:

$$\|(\nabla_x G(x_1, y_1) - \nabla_x G(x_2, y_2), \nabla_y G(x_1, y_1) - \nabla_y G(x_2, y_2))\| \leq Z\|(x_1 - x_2, y_1 - y_2)\|.$$

Assumption 3 $\Psi(.)$ satisfies the KL property: according to Remark 2, for most of prevalent constraints such as non-negativity, the simplex set and ℓ_p norm ($p = 0, 1, 2, \infty$), $\Psi(.)$ is semi-algebraic, hence satisfies the KL. In the rest of this thesis, we call this assumption "*KL assumption*".

5.5.4.2 The convergence of each step of AO

The convergence of each step of AO in SFBS, which is Forward-Backward Splitting for each block of variable (cf. Theorem 1), is next discussed. It is usually expected to have the *global convergence* when the cost function to be minimized is convex [ABS13], which means that the algorithm generates a converging sequence to the solution regardless of the starting point. However, if the objective function is non-convex, the monotonicity of the sequences generated by descent methods will be broken and oscillatory behaviors may appear [ABS13]. In order to achieve the convergence in such cases, it is necessary to limit ourselves to functions with some particular properties, such as KL [ABS13].

The convergence of Forward-Backward Splitting in Theorem 1 is proved in [ABS13], and the provided convergence analysis is not only applicable for continuous and convex cost functions, but is also usable for non-smooth and non-convex functions (or a non-convex set of constraints). It has been proved

that the sequence $\boldsymbol{x_k}$ generated by Theorem 1 converges to a critical point of $f = h + g$, if the mentioned sequence is bounded [ABS13]. Therefore, the convergence guarantee of each step of SFBS results from the following theorem, quoted from [ABS13]:

Theorem 2 (The convergence of Forward-Backward Splitting [ABS13])**.** *Suppose $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ is a proper lower semi-continuous function, which has the KL property and is bounded from below. Assume that f can be spilt into two parts as $f = h + g$, where g is lower semi-continuous and $h : \mathbb{R}^N \mapsto \mathbb{R}$ is a finite valued, differentiable function with a β-Lipschitz continuous gradient.*

If the sequence generated by Algorithm 5 is bounded, then this sequence will converge to a critical point of f. In addition, by choosing $\gamma \in [0, \frac{1}{\beta}]$ (γ is the coefficient of ∇h in the fixed point equation (5.7)*, and β is the Lipschitz constant of ∇h), the values of the cost function are not increasing.*

Let us explain that the assumptions in Theorem 2 are completely satisfied for each step of SFBS. According to Theorem 2, the objective function in each step, *i.e.* (5.12), should be a proper lower semi-continuous (lsc) function, which has the KL property and is bounded from below. As mentioned before, it is proper as its domain is not null. This function is lsc, since the fidelity term, $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$, is continuous and the regularization term, equivalent to g in Theorem 2, is lsc (cf. Assumption 1 (i)). Finally, the objective function in (5.12) is bounded from below (since its values are always non-negative), and satisfies KL according to the Remark 2. Assumptions on h and g are the same as that of Theorem 1, and the same explanations provided for Theorem 1 hold also here.

The only assumption to be discussed is that of generating a *bounded* sequence by SFBS, which is a critical assumption in Theorem 2. [ABRS10, Remark 5] discusses about the assumptions, which guarantee the boundedness of generated sequence. For instance, the *coercivity* [RW09, Definition 3.25] of objective function is simply sufficient to obtain bounded sequence from

SFBS. A function is coercive, if it is bounded below on bounded sets and

$$\liminf_{|x|\to\infty} \frac{f(x)}{x} = \infty.$$

It can be observed that the objective function in (5.12) is coercive.

We now review the main theoretical and practical advantages of SFBS described in Algorithm 5. Firstly, it can be applied even on non-convex and non-smooth constraints such as cardinality [ABS13], and this does not affect its convergence. Secondly, it does not require any critical setting of the parameters. The only parameters to be set are γ and α, where the suggested values in Algorithm 5 are almost always suitable. Thirdly, compared to state-of-the-art methods such as APG and BC-VMFB, SFBS is easy enough to understand and implement. Fourthly, contrary to BC-VMFB, SFBS works with variables in matrix form, so there is no need to vectorize loading matrices. This brings an advantage in working with large dimension tensors. Moreover, as it is developed in the next section, SFBS can be adapted to many different constraints.

5.6 Simulation

The computer experiments described in this section can be classified in three main groups:

- **Synthetic data:** the decomposition of data tensor synthetically generated under the non-negative and the simplex constraint;
- **Synthetic data with hidden relation**[8]**:** the decomposition of the third order moments tensor of synthetic data generated according to the generative processes described in Section 3.4.1 under the non-negative and the simplex constraint;
- **Real data:** the decomposition of the third order moments tensor of a well-known text data set, namely 20 Newsgroups, which consists of

[8]By hidden relation, we mean the relation between hidden/latent variables (topics) and multi-view variables (words), which is depicted in Fig. 3.1.

approximately 20000 posts on 20 topics [Nig00], under the simplex constraint.

We compare SFBS with those described in Section 5.3, namely AO-ADMM [HSL16], APG [XY13b], FastNTF-APG [ZZZ+16] and BC-VMFB [VCTMM17][9]. The software of CP decomposition via least squares, namely *Nway*, from a well-known tensor Tool-box [BK+21] is also added in our comparisons. Algorithms are either tested on artificially generated tensors, which are corrupted with additive noise, or on some artificially estimated third order moments, which contain intrinsic noise due to the lack of accuracy in estimating moments. In addition, some experiments on a real text data set (20 Newsgroups) are also added to complete this chapter.

In order to compare the performances of algorithms fairly, we require all the algorithms to iterate until either they reach a maximum predefined number of iterations (denoted by *iterations max-number*) or the variation of *relative objective value* between two successive iterations is less than a desired small value, namely ϵ_1. If we denote the objective value in iteration k by $\Psi(k)$, which is the difference between the input tensor and the estimated tensor, the criterion to stop iterations is as follows:

$$\Delta\Psi(k) = \frac{\Psi(k) - \Psi(k-1)}{\Psi(k)} \leq \epsilon_1. \tag{5.19}$$

In addition, the reported result of each experiment is averaged over several realizations of the tensors, and for each realization, all the methods are initialized by an identical set of initializations to choose the best initialization point. Therefore, it is possible that not the same initialization point will be the best for all the methods, but nonetheless, the comparison is fair, since the set of initializations is the same for all the considered methods.

[9]We would like to thank the corresponding authors of FastNTF-APG [ZZZ+16] and BC-VMFB [VCTMM17], Guoxu Zhou and Caroline Chaux, respectively, who sent us the MATLAB codes of their methods. The MATLAB codes of AO-ADMM [HSL16] and APG [XY13b] are made available by the authors at [Hua15] and [XY13a], respectively. Therefore, the original codes of authors have been used to obtain the results reported in all figures of this section.

In the sequel, we mention the number of the realizations of tensors and the number of initialization points by *average number* and *initialization number*, respectively.

All computer experiments reported in this section have been executed either on a laptop with a processor of 3.1 GHz Intel Core i5, 16 GB RAM or on a PC with a processor of 3.2 GHz Intel Core i5, 8 GB RAM, both running macOS Mojav and MATLAB 2019a.

5.6.1 Synthetic data

According to the considered constraints (non-negativity or the simplex set), we generate randomly (uniform distribution in the interval $[0, 1]$) loading matrices and a coefficient vector. Then, the noiseless tensor $\boldsymbol{\mathcal{T}}_o$ is computed via (2.6).

In order to work in a noisy context, a noise tensor, $\boldsymbol{\mathcal{T}}_n$, with i.i.d. entries of Gaussian distribution with zero mean and unit variance, of the same size as $\boldsymbol{\mathcal{T}}_o$, is weighted by the parameter σ and added to $\boldsymbol{\mathcal{T}}_o$. As $\boldsymbol{\mathcal{T}}_n$ has unit variance, then the variance of $\sigma\boldsymbol{\mathcal{T}}_n$ is σ^2. σ is adjusted such that we reach a desired Signal to Noise Ratio (SNR) according to the following relation:

$$\text{SNR} = 10\log_{10}\frac{\frac{1}{M}\sum_{i,j,k}\boldsymbol{\mathcal{T}}_o(i,j,k)^2}{\frac{1}{M}\sum_{i,j,k}\sigma^2\boldsymbol{\mathcal{T}}_n(i,j,k)^2}, \tag{5.20}$$

where M is the total number of elements in tensors $\boldsymbol{\mathcal{T}}_o$ or $\boldsymbol{\mathcal{T}}_n$.

Denote by $\boldsymbol{\mathcal{T}}$ the desired tensor to be decomposed, $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_o + \sigma\boldsymbol{\mathcal{T}}_n$. After decomposing $\boldsymbol{\mathcal{T}}$, the estimation of $\boldsymbol{\mathcal{T}}_o$ can be calculated through (2.6), as a rank-R approximation, which we call $\hat{\boldsymbol{\mathcal{T}}}$. The relative reconstruction error is computed as follows:

$$\epsilon(\hat{\boldsymbol{\mathcal{T}}}) = \frac{\|\hat{\boldsymbol{\mathcal{T}}} - \boldsymbol{\mathcal{T}}_o\|_F^2}{\|\boldsymbol{\mathcal{T}}_o\|_F^2}, \tag{5.21}$$

which will be reported as it is (not in the form of percentage[10]) for all the experiments of synthetic scenarios. In addition to the reconstruction error,

[10]For example, if we report $\epsilon(\hat{\boldsymbol{\mathcal{T}}}) = 1.5$, it should not be interpreted as 1.5%, but as 150%.

we will report the error of estimating loading factors but according to the points remarked in Remark 3.

Remark 3. *As mentioned in Chapter 4, it is hard to assess the relative error made on loading matrices, because of the scaling and the permutation ambiguities of tensor decomposition [Com14]. So as to overcome these ambiguities, we report CorrIndex measure described in Chapter 4 as the performance on estimating loading factors. However, if we report CorrIndex on each loading factor separately, it would be an optimistic measure, since implicitly a specific (not common) permutation is permitted for each loading matrix.*

In order to have a more reliable performance index, we report CorrIndex based on the matrix $\boldsymbol{X}$ which consists of all loading factors together. In other words, we report CorrIndex($\boldsymbol{X}, \widehat{\boldsymbol{X}}$), where $\boldsymbol{X}, \widehat{\boldsymbol{X}}$ are defined as follows

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \\ \vdots \\ \boldsymbol{A}^{(N)} \end{bmatrix}, \quad \widehat{\boldsymbol{X}} = \begin{bmatrix} \widehat{\boldsymbol{A}^{(1)}} \\ \widehat{\boldsymbol{A}^{(2)}} \\ \vdots \\ \widehat{\boldsymbol{A}^{(N)}} \end{bmatrix}. \tag{5.22}$$

*Nevertheless, it is worth to report also exact errors based on the methods described in 4.3.2 (*e.g. *Hungarian), and compare them with the results interpreted from CorrIndex.*

We should note two technical points about our practical implementations. First, in order to compute Lipschitz constant (line 4 of Algorithm 5), one can employ $[\texttt{norm(W)}]^2$ command in MATLAB, where $\boldsymbol{W}$ is the matrix composed of constant loading matrices and is defined in line 3 of Algorithm 5. Second, as all the investigated algorithms in this chapter require the rank R, as their input, one could employ Corcondia [BK03] to obtain an estimation of R. However, in synthetic scenarios, we are aware of real rank. In working with real text data set, some points on choosing the rank are remarked in Section 5.6.3.

5.6.1.1 Non-negativity constraint

Small size tensors. In all simulations of this section, $\epsilon_1 = 10^{-20}$, $e = 1.9$ and we repeat five times the loop of line 6 for each mode in Algorithm 5 (SFBS algorithm).

Figure 5.3 shows the relative reconstruction error of the noiseless tensor of size $10 \times 10 \times 10$, of rank $R = 6$ with initialization number, average number and iterations max-number, equal to 10, 10 and 5000, respectively (cf. see page 101 for the definition of these parameters). As it can be seen at the first glance, BC-VMFB, in spite of our effort to adjust its parameters properly, results in a relative error around 1, which means 100% (remind that we always report relative error according to (5.21) which is not in the form of percentage). Although we consulted the corresponding author of BC-VMFB about the parameters of their algorithm, we found it hard to adjust several parameters of BC-VMFB. In fact, not only these parameters depend on data, but also they have a critical effect on the final result of the method. Therefore, it seems unnecessary to include BC-VMFB in the rest of comparisons in this experiment. In addition, since FastNTF-APG tries to decompose a low-rank approximation of the desired tensor, it can be expected that the performance of FastNTF-APG is worse than APG. As a result, we remove FastNTF-APG in some comparisons of this chapter. Although most of the methods achieve reasonable performances, APG and SFBS converge rather faster.

In Fig. 5.4, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian algorithms (which is an exact error) is reported for the same experiment of Fig. 5.3. As it can be seen in Fig. 5.4, the same interpretation as that of Fig. 5.3 can be concluded from CorrIndex and Hungarian. Moreover, the gap between loading factors[11] and their estimations (Fig. 5.5 - Fig. 5.7) reveals the same conclusion, that is, APG and SFBS converge slightly faster than others, whereas all the methods (except FastNTF-APG and BC-VMFB) reach a reasonable performance.

[11] In plots, we show the loading factors of a third order tensor by $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

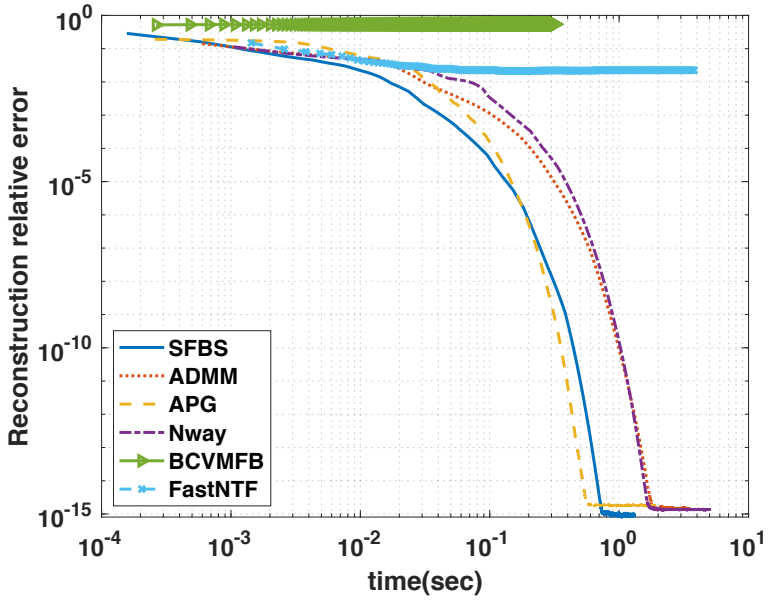

Figure 5.3: The reconstruction relative error in decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 6$ under the non-negativity constraint over all loading factors, in noiseless case with the following setting: $\epsilon_1 = 10^{-20}$, average number= 10, initialization number= 10, iterations max-number= 5000, $e = 1.9$.

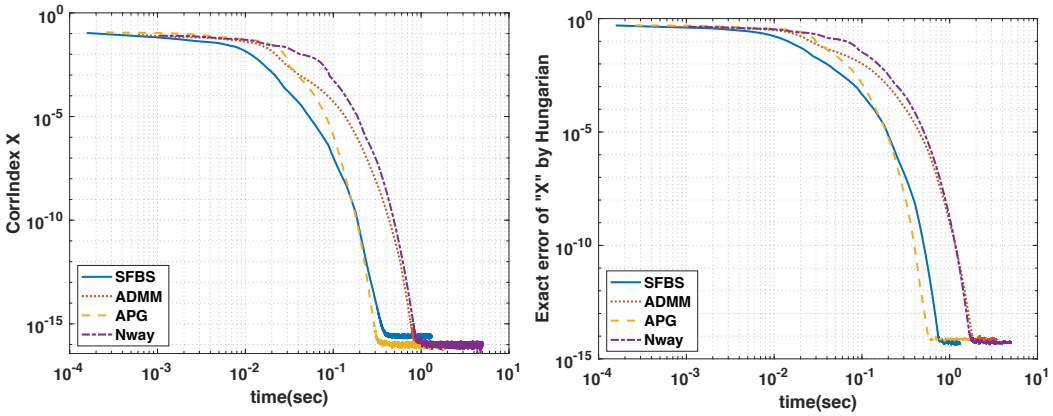

Figure 5.4: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.3.

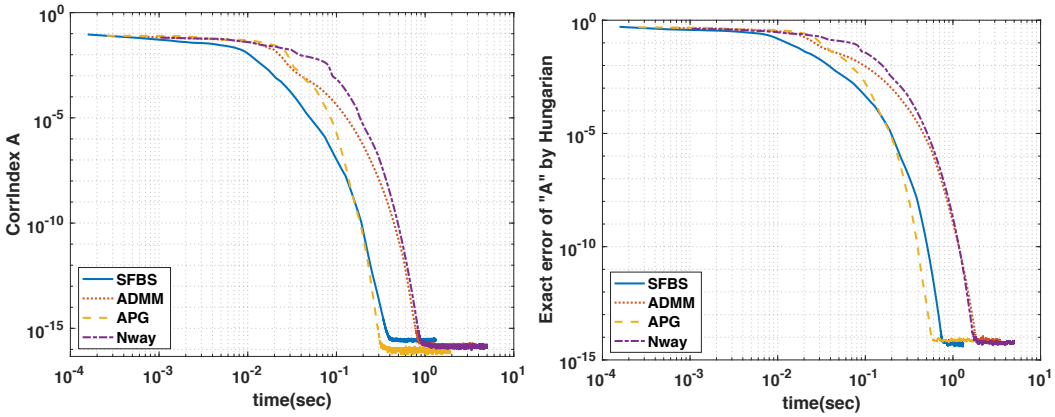

Figure 5.5: Compare the estimation of matrix $\boldsymbol{A}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.3.

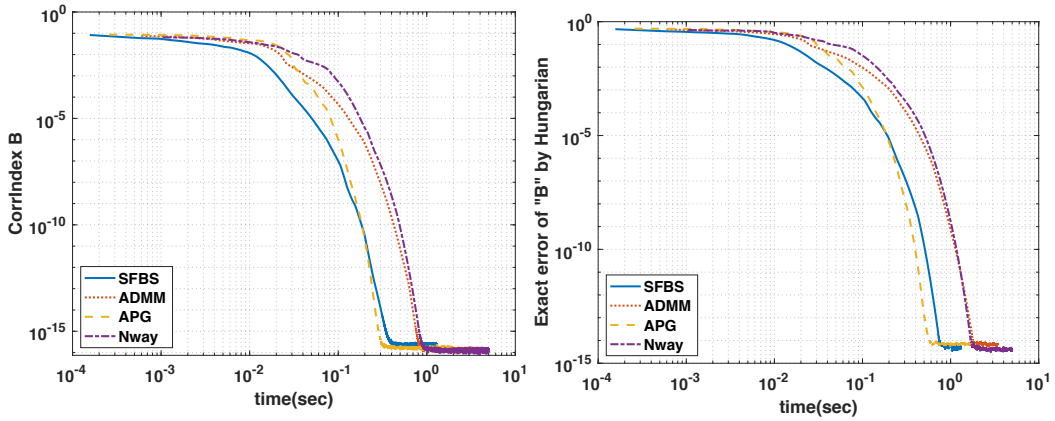

Figure 5.6: Compare the estimation of matrix $\boldsymbol{B}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.3.

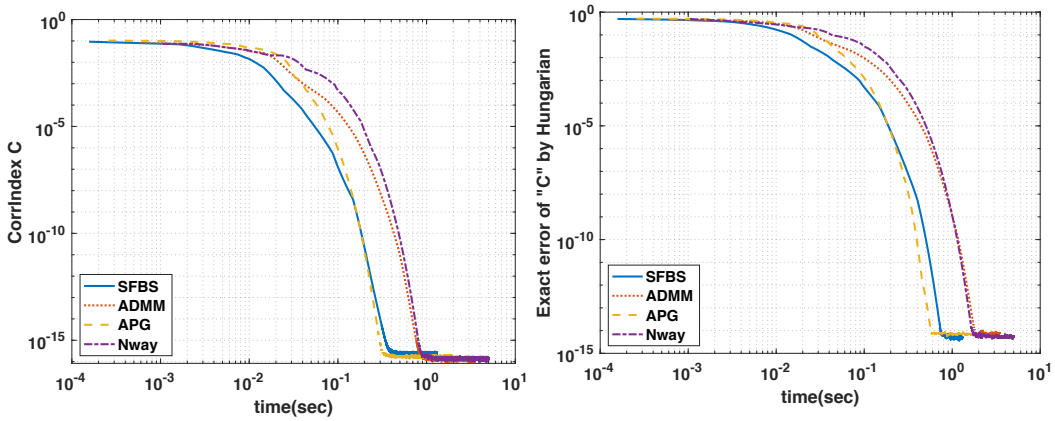

Figure 5.7: Compare the estimation of matrix $\boldsymbol{C}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.3.

Since the relative error of loading factors (like what are represented in Fig. 5.5 - Fig. 5.7) results almost the same result as the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian, in the rest of the experiments of this section, we shall just report the relative reconstruction error and the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian (like Fig. 5.4).

Figure 5.8 indicates the relative reconstruction error of a noisy tensor of size $10 \times 10 \times 10$, of rank $R = 6$ with SNR $= 10$, initialization number $= 20$, average number $= 200$ and iterations max-number $= 1000$. Figure 5.8 (Left) compares all the methods together, whereas Figure 5.8 (Right) excludes BC-VMFB in order to show the results of other methods more precisely.

As in Fig. 5.3, BC-VMFB does not perform properly in the experiment of Fig. 5.8. In addition, although Nway carries out well in a noiseless case, its performance is not acceptable in the noisy situation (cf. Fig. 5.8). As mentioned before, FastNTF-APG replaces the noisy tensor with its low rank approximation, which helps to filter the noise out. As it can be seen in Fig. 5.8, the result of FastNTF-APG is better than APG in a noisy scenario, whereas APG is one the best methods for the decomposition of a noiseless tensor (cf. Fig. 5.3). Moreover, SFBS and AO-ADMM outperform other methods in the experiment of Fig. 5.8. Therefore, the only algorithm that performs properly and better than others in both noiseless and noisy situations is SFBS.

In Fig. 5.9, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian algorithms (which is the exact error) is reported for the same experiment of Fig. 5.8. Although according to the relative reconstruction error in Fig. 5.8, both AO-ADMM and SFBS outperform others, in estimating loading factors (cf. Fig. 5.9), SFBS performs better than AO-ADMM. The performance on estimating loading factors is critical in many applications such as data mining and text mining explained in Chapter 3. In addition, as Nway performs inadequately in the noisy case, this reveals the importance of considering constraints in the decomposition of tensors (remind that Nway executes simply non-constrained CP decomposition).

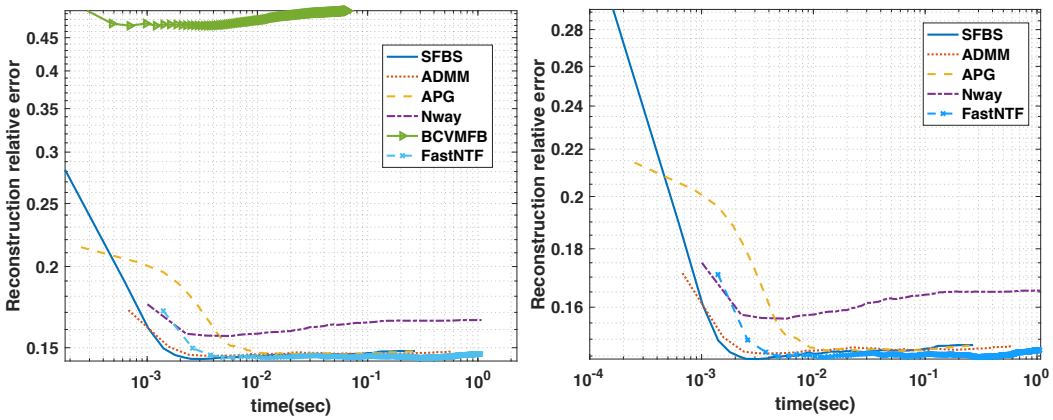

Figure 5.8: The reconstruction relative error including BC-VMFB (Left) and excluding BC-VMFB (Right) in decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 6$ under the non-negativity constraint over all loading factors, in a noisy case with SNR $= 10$ and with the following setting: $\epsilon_1 = 10^{-20}$, average number $= 200$, initialization number $= 20$, iterations max-number $= 1000$, $e = 1.9$.

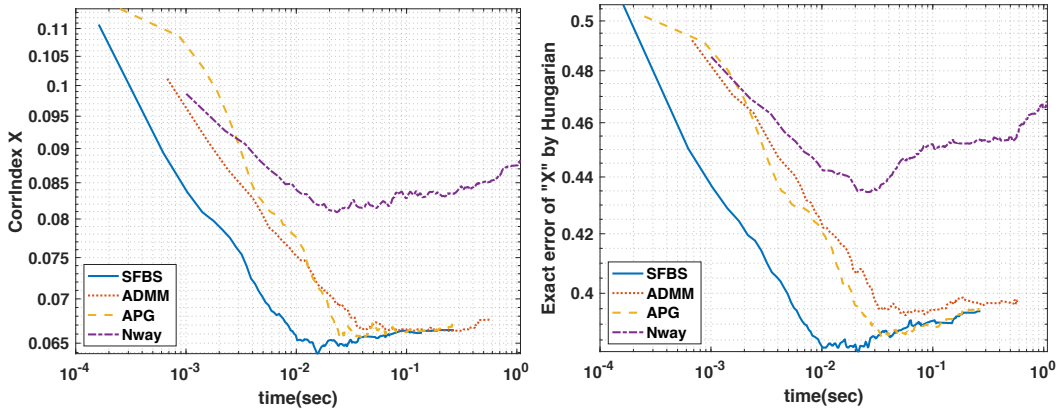

Figure 5.9: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.8.

Large size tensors. Figures 5.10 and 5.11 correspond to the decomposition of a noiseless tensor of size $100 \times 100 \times 100$, of rank $R = 3$ with initialization number , average number and iterations max-number, equal to 10, 10 and 5000, respectively (cf. see page 101 for the definition of these parameters). Comparing the results in Fig. 5.10 and Fig. 5.11, it can be inferred that all methods except BC-VMFB and FastNTF-APG allow to reach an acceptable performance in terms of relative reconstruction error *and* estimation of loading factors. In addition, SFBS performs slightly better than others. Therefore, the same interpretations can be concluded for large tensors (of size $100 \times 100 \times 100$) as those for small ones (of size $10 \times 10 \times 10$) in Fig. 5.3 and Fig. 5.4.

Figure 5.12 indicates the relative reconstruction error of a noisy tensor of size $100 \times 100 \times 100$, of rank $R = 3$ with SNR, initialization number, average number and iterations max-number, equal to 10, 20, 200 and 1000, respectively. As the decomposition of large noisy tensors is a difficult problem to handle, FastNTF-APG is not anymore better than APG like the experiment in Fig. 5.8. In addition, unlike Fig. 5.8, Nway reaches the same level of relative error as others after convergence. Nevertheless, in this experiment, AO-ADMM and SFBS outperform others and also SFBS is slightly better.

In Fig. 5.13, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian algorithms (which is the exact error) is reported for the same experiment of Fig. 5.12. As it can be seen in Fig. 5.13, SFBS performs slightly better in estimating loading factors compared to the others.

5.6.1.2 Simplex constraint

In this section, we investigate practically the performance of tensor decomposition algorithms under the simplex set constraint over all the columns of all loading factors and over $\boldsymbol{\lambda}$. In the simulations of this section, we set $\epsilon_1 = 10^{-20}$ and repeat five times the loop of line 6 for each mode in Algorithm 5 (SFBS algorithm).

In this section, we compare SFBS with AO-ADMM, since, firstly, the

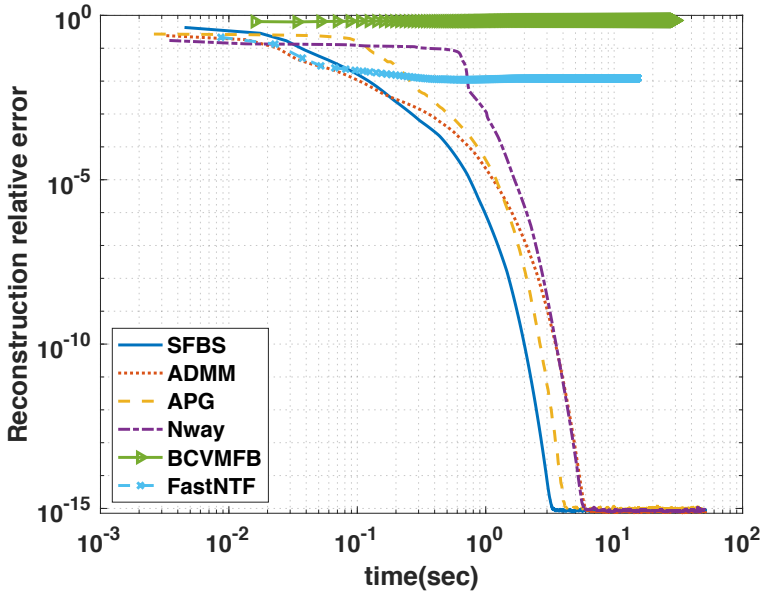

Figure 5.10: The reconstruction relative error in decomposing a tensor of dimension $100 \times 100 \times 100$, of rank $R = 3$ under the non-negativity constraint over all loading factors, in a noiseless case with the following setting: $\epsilon_1 = 10^{-20}$, average number = 10, initialization number = 10, iterations max-number = 5000, $e = 1.9$.

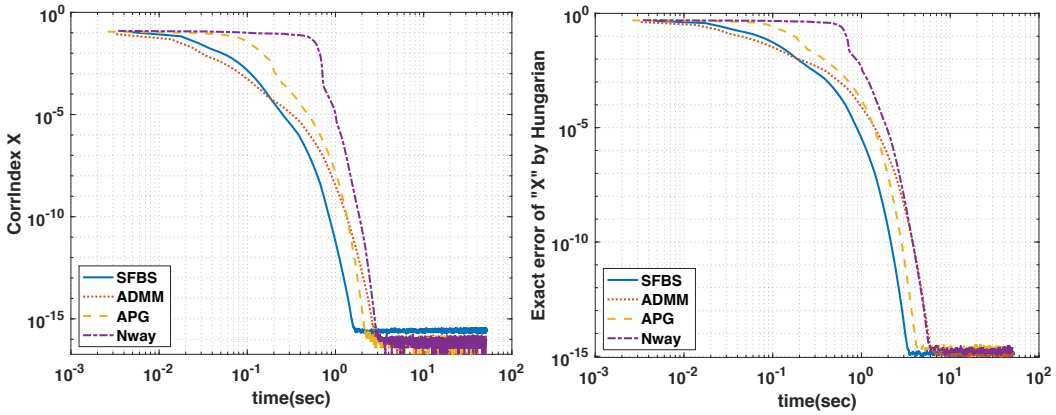

Figure 5.11: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.10.

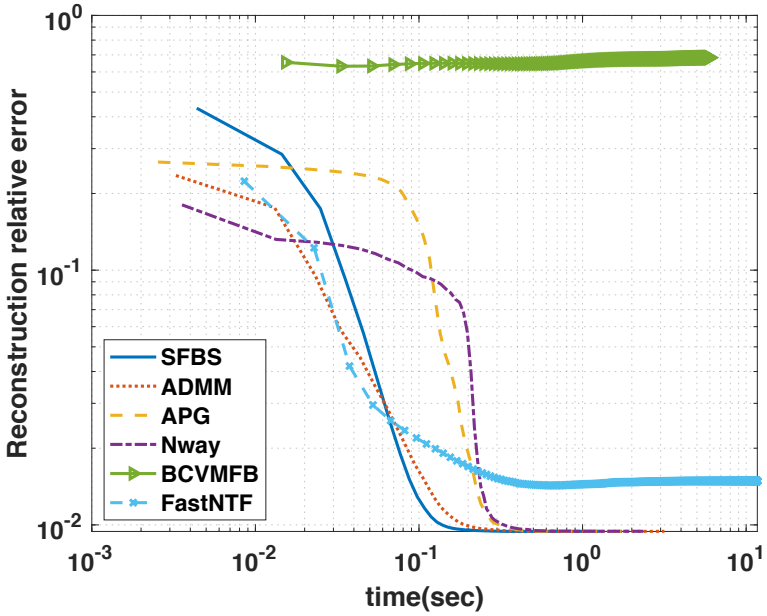

Figure 5.12: The reconstruction relative error including BC-VMFB (Left) and excluding BC-VMFB (Right) in decomposing a tensor of dimension $100 \times 100 \times 100$, of rank $R = 3$ under the non-negativity constraint over all loading factors, in a noisy case with SNR $= 10$ and with the following setting: $\epsilon_1 = 10^{-20}$, average number$= 200$, initialization number$= 20$, iterations max-number$= 1000$, $e = 1.9$.

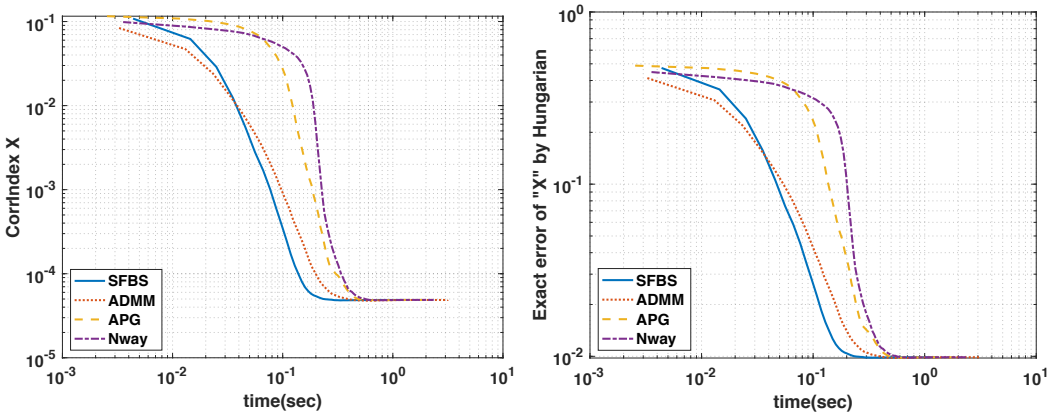

Figure 5.13: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.12.

simplex constraint is directly considered and investigated by the authors in [HSL16] as well as in their implementation. Secondly, according to the results under the non-negativity constraint in Section 5.6.1.1, AO-ADMM is the only algorithm, which has the closest performance to that of SFBS in both noiseless and noisy cases.

The methodology for applying the simplex set constraint on all the columns of all loading factors in addition to the vector of coefficients ($\boldsymbol{\lambda}$) is explained in Section 5.5.2.2, and we have utilized this methodology in our simulations of the current section.

Noiseless cases. Figure 5.14 shows the relative reconstruction error of the noiseless tensor of size $10 \times 10 \times 10$, of rank $R = 3$ with initialization number, average number, e and iterations max-number, equal to 10, 10, 1.5 and 20000, respectively, under the simplex set constraint over all the columns of all loading factors in addition to $\boldsymbol{\lambda}$[12]. By comparing Fig. 5.14 and Fig. 5.3, it can be inferred that the simplex set is a more difficult constraint than non-negativity to be achieved, since the maximum required iterations for the simplex set constraint is 20000, and it is more than what is set for the non-negativity constraint (*i.e.* 5000) for the convergence of the algorithms. As it can be seen in Fig. 5.14, both SFBS and AO-ADMM achieve the same level of relative error, however SFBS converges slightly faster.

In Fig. 5.15, the gap between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian algorithm (which is the exact error) is reported for the same experiment of Fig. 5.14. The same conclusion as before can be drawn from estimating loading factors, that is, both SFBS and AO-ADMM perform well, however, SFBS converges a bit faster.

In order to show that Fig. 5.15 has sufficient information even under the simplex set constraint (like the non-negativity constraint), we bring the result of the estimation of first mode loading factor ($\boldsymbol{A}$) in Fig. 5.16, which has the same information as Fig. 5.15. As matrix $\boldsymbol{X}$ (cf. Fig. 5.15) does not

[12] See page 101 and Algorithm 5 for the definition of these parameters.

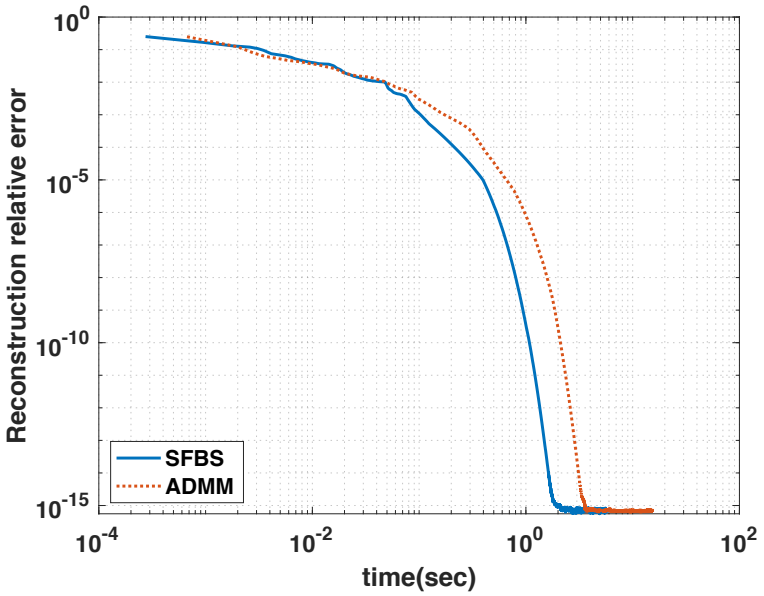

Figure 5.14: The reconstruction relative error in decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 3$ under Simplex set constraint over all the columns of all loading factors and over $\boldsymbol{\lambda}$, in a noiseless case with the following setting: $\epsilon_1 = 10^{-20}$, average number= 10, initialization number= 10, iterations max-number= 20000, $e = 1.5$.

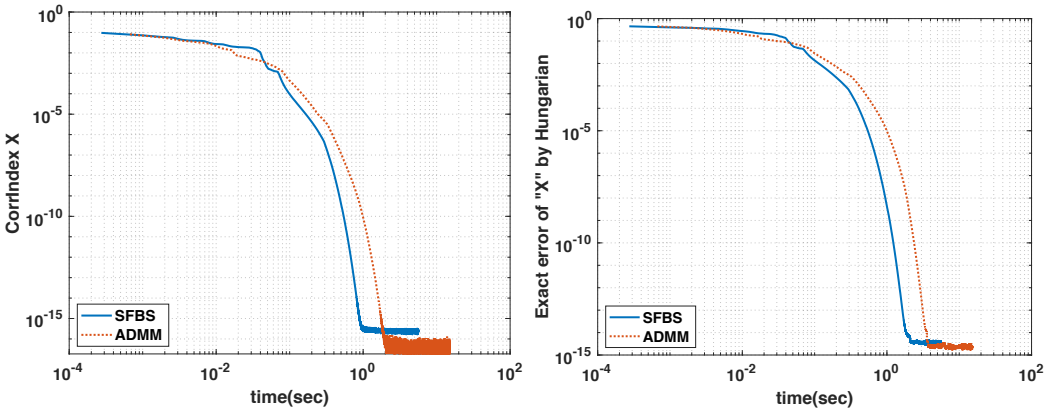

Figure 5.15: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.14.

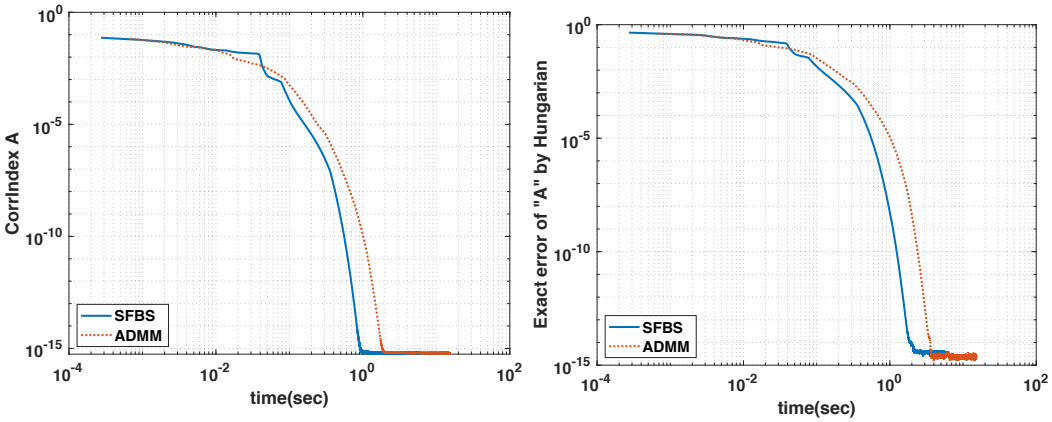

Figure 5.16: Compare the estimation of matrix $\boldsymbol{A}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.14.

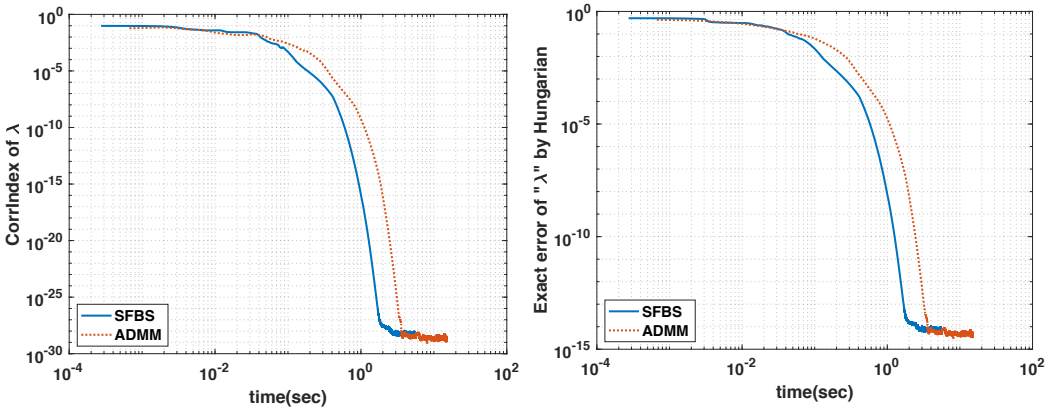

Figure 5.17: Compare the estimation of vector $\boldsymbol{\lambda}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.14.

include the information of the estimation of $\boldsymbol{\lambda}$, we compare the performance of algorithms in the estimation of $\boldsymbol{\lambda}$ in Fig. 5.17, which reveals the same conclusion as the other plots of this experiment.

Noisy cases. Figure 5.18 show the relative reconstruction error of a tensor of size $10 \times 10 \times 10$, of rank $R = 3$ with SNR, initialization number, average number, e and iterations max-number, equal to 10, 20, 200, 1.9 and 1000, respectively, under the simplex set constraint over all the columns of all loading factors in addition to $\boldsymbol{\lambda}$. Figure 5.19 represents the gap between $\boldsymbol{X}$

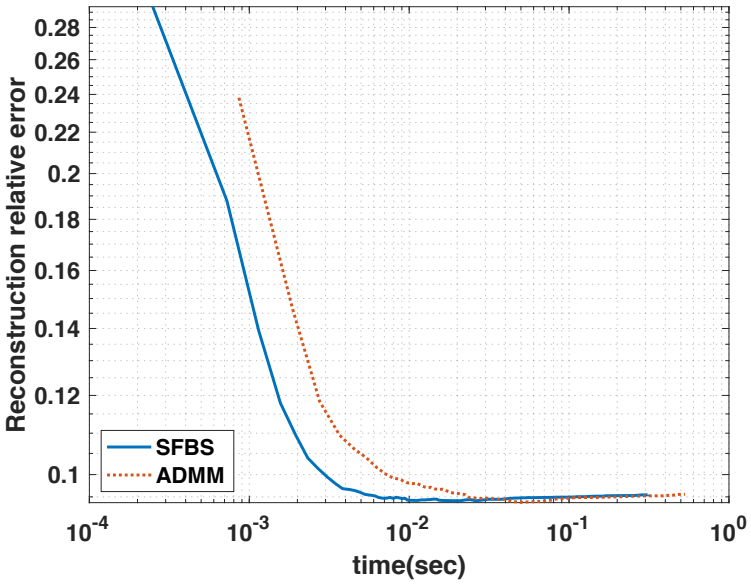

Figure 5.18: The reconstruction relative error in decomposing a tensor of dimension $10 \times 10 \times 10$, of rank $R = 3$ under the simplex set constraint over all the columns of all loading factors and over $\boldsymbol{\lambda}$, SNR $= 10$ with the following setting: $\epsilon_1 = 10^{-20}$, average number= 200, initialization number= 20, iterations max-number= 1000, $e = 1.9$.

and $\widehat{\boldsymbol{X}}$ via CorrIndex and Hungarian algorithm (which is the exact error) for the same experiment of Fig. 5.18. Both Fig. 5.18 and Fig. 5.19 show that SFBS performs better than AO-ADMM and it converges slightly faster.

Figure. 5.20 shows the gap between $\boldsymbol{\lambda}$ and $\widehat{\boldsymbol{\lambda}}$ via CorrIndex and Hungarian algorithm (which is the exact error) for the same experiment of Fig. 5.18. The same conclusion can be drawn as before, that is, SFBS performs and converges better and faster than AO-ADMM. It can be seen that the relative error in Fig. 5.20 increases slightly, but note that decreasing monotonically is essential for the relative reconstruction error (or objective function) not in the estimation of loading factors or $\boldsymbol{\lambda}$, which is the case in all the experiments such as the one expressed in Fig. 5.18.

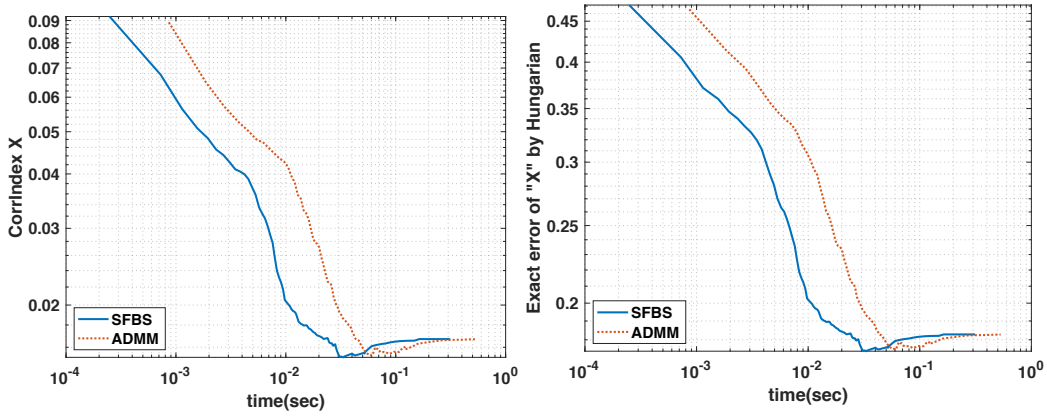

Figure 5.19: Compare the estimation of matrix $\boldsymbol{X}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.18.

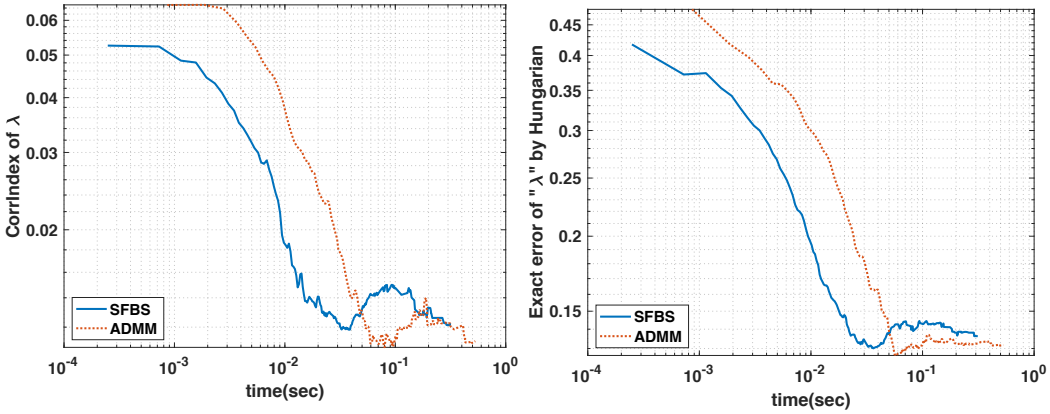

Figure 5.20: Compare the estimation of vector $\boldsymbol{\lambda}$ via CorrIndex (Left) and Hungarian (Right) in the same experiment as Fig. 5.18.

5.6.2 Synthetic data with hidden relations

In order to follow much more easily the computer experiments of this section, it is better to review briefly the notation of the data mining problem described in Section 3.3:

- h: a hidden variable (topic) encoded into a discrete variable taking K possible integer values, say in $\mathcal{H} = \{1, 2, \ldots, k, \ldots, K\}$ with the probability $\varphi(k) = \text{Prob}(h = k)$
- $\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d, \ldots, \boldsymbol{u}_D\}$: the encoded dictionary of cardinality D for the multi-view variables (words), x
- $f_k(d) = \text{Prob}(x = \boldsymbol{u}_d | h = k)$: the conditional probability of each word $\boldsymbol{u}_d$ of dictionary Ω, given a particular topic, $h = k$
- $\boldsymbol{a}_k$: a vector of dimension D contains the values of $f_k(d)$ for all d
- $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k, \ldots, \boldsymbol{a}_K]$.

The goal is to estimate the probabilities $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ by means of the decomposition of the third order moments tensor ($\boldsymbol{\mathcal{T}}$) of the observed multi-view variables (words), which can be employed for some targeted data mining tasks such as unsupervised clustering. As it is described in Section 5.4, some algorithms such as Robust tensor power method require also the second order moments ($\boldsymbol{P}$) for the decomposition of $\boldsymbol{\mathcal{T}}$. In addition, some methods of moment estimation, such as simple and standard averaging, have been expressed in Section 3.4.

5.6.2.1 The drawbacks of Robust tensor power method [AGH$^+$14]

In this section, we show experimentally the drawbacks of Robust tensor power method, namely the negative estimated probability (which is also reported in our paper [SCBZ19] but we shall explain it completely in the sequel) and the sensitivity to the additive noise. Figures 5.21 and 5.22 correspond to a simulation, which reveals that the estimated probabilities by Robust tensor

power method occasionally are not acceptable, since they include negative values. Table 5.2, Fig. 5.23 and Fig. 5.24 experimentally prove that Robust tensor power method is very sensitive to the additive noise.

Negative estimated probabilities. In the simulation of Fig. 5.21 and Fig. 5.22, we generated some arbitrary probabilities, *i.e.* $\boldsymbol{A}^{6\times 4}$ and $\boldsymbol{\varphi}^{4\times 1}$, which are depicted in Fig. 5.21 (left column) and Fig. 5.22 (top). Note that in Fig. 5.21 (left column), four plots correspond to each column (each hidden variable or topic, *i.e.* k) of the matrix $\boldsymbol{A}$, and in each plot, the six values are the conditional probabilities of multi-view variables (words), *i.e.* d, according to their hidden variable (topic). Similarly, Fig. 5.22 (top), contains the probabilities of four hidden variables (topics), *i.e.* k, in this simulation. Then, by considering these probabilities, synthetic data with hidden relation are generated via the generative procedure described in Section 3.4.1.1[13]. The method of moment estimation in this simulation to obtain empirical second and third order moments is simple averaging with $N_c = 10^5$ documents for averaging. Note that these documents construct a corpus of size N_c. Finally, we applied Robust tensor power method and the non-negative AO-ADMM on these empirical moments, and obtained the estimation of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$.

Middle and right column in Fig. 5.21 correspond to the estimation of the columns of matrix $\boldsymbol{A}$ with the non-negative AO-ADMM and Robust tensor power method, respectively. In order to compare the performances, the plots of each row of Fig. 5.21 should be compared with each other. In all the rows of Fig. 5.21 except the first row, the negative values can be seen in the estimation of Robust tensor power method, which are not acceptable as the estimated probabilities for the targeted data mining task.

The performances in estimating vector $\boldsymbol{\varphi}$ have been compared in Fig. 5.22, where the figures in the middle and at the bottom correspond to the non-negative AO-ADMM and Robust tensor power method, respectively. It is clear that the result of the non-negative AO-ADMM is closer to the original

[13]To calculate cumulative distributions, the MATLAB function `cumsum()` can be used.

vector $\boldsymbol{\varphi}$.

Sensitivity to additive noise. In the sequel, we shall experimentally show that Robust tensor power method does not perform well in presence of additive noise. In order to assess this, we generated some arbitrary matrices $\boldsymbol{A}^{10\times 3}$ and vectors $\boldsymbol{\varphi}^{3\times 1}$. Then, we calculated their original corresponding third and second order moments, *i.e.* $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$ (cf. (3.5) and (3.6)). At the end, we applied several tensor decomposition algorithms, including Robust tensor power method, on both the noiseless (Table 5.2) and the noisy (Fig. 5.23 and Fig. 5.24) versions of $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$.

Table 5.2 reports the relative reconstruction error and the relative exact error in estimating $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ (measured with Hungarian algorithm) from noiseless $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$. These results averaged over 200 realizations of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$[14]. The considered algorithms in this experiment are as follows:

1 - **Power method:** Robust tensor power method

2 - **Projected Power method:** as the result of Robust tensor power method is not in the simplex set, we modified this method by projecting its result to the simplex set.

3 - **Simplex AO-ADMM:** AO-ADMM with the simplex set constraint

4 - **Simplex SFBS:** SFBS with the simplex set constraint

5 - **Symmetric Simplex SFBS:** unlike Simplex SFBS, we take into account the symmetric property of the third order moments tensor. To be more precise, in calculating the objective function for the criterion to stop the iterations of SFBS, we utilized an average of all estimated loading factors.

[14]In this experiment, we work with true moments instead of their estimations. Therefore, we do not need realizations of documents. Note that we need realizations of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ to generate different values of $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$, but with realizations of documents, the empirical estimations in (3.8) and (3.7) are computed.

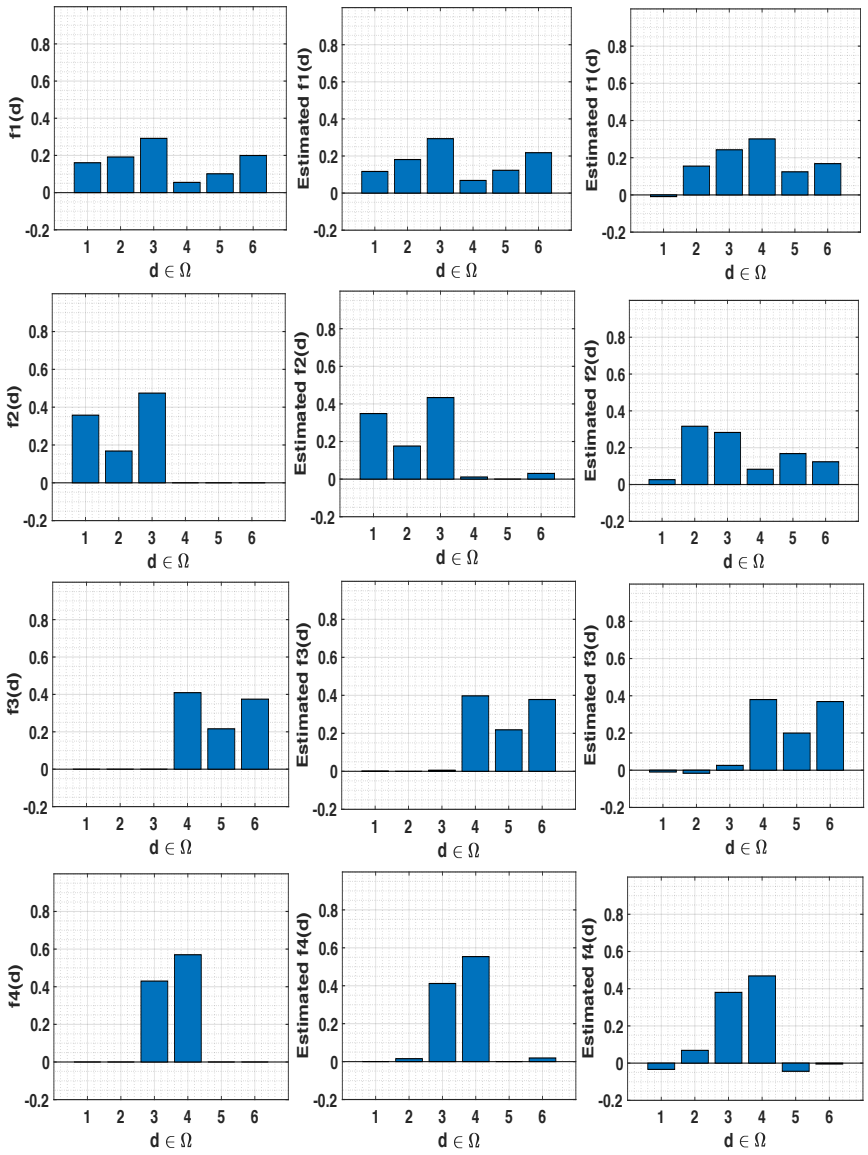

Figure 5.21: Left column: an arbitrary conditional probability, $\boldsymbol{A}^{6\times 4}$. Four plots correspond to each column (each hidden variable or topic, *i.e.* k) of the matrix $\boldsymbol{A}$, which contains 6 probability values for each multi-view variables (d). Middle column: the estimation of the matrix $\boldsymbol{A}$ with the non-negative AO-ADMM. Right column: the estimation of the matrix $\boldsymbol{A}$ with Robust tensor power method including some negative estimated probabilities, which are not acceptable.

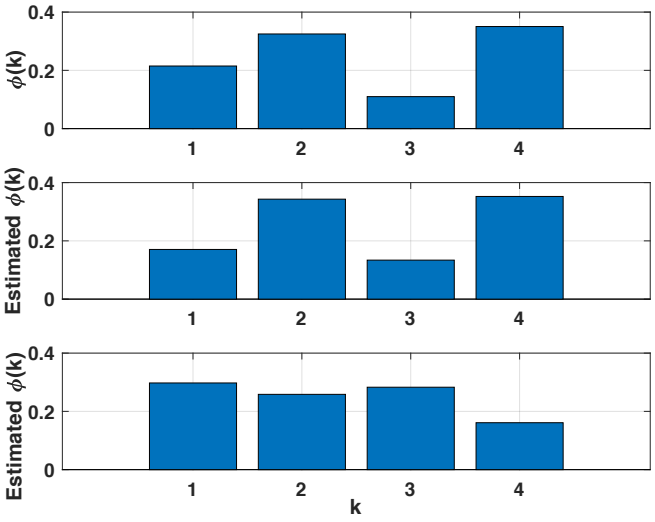

Figure 5.22: Top: an arbitrary probability distribution $\boldsymbol{\varphi}^{4\times 1}$ contains four values for each hidden variable (topic), k. Middle: the estimation of $\boldsymbol{\varphi}$ with non-negative AO-ADMM. Bottom: the estimation of $\boldsymbol{\varphi}$ with Robust tensor power method. Obviously, the result of non-negative AO-ADMM is much more closer than Robust tensor power method to the original probability distribution (top).

For the last three methods (*i.e.* Simplex AO-ADMM, Simplex SFBS and Symmetric Simplex SFBS), we set initialization number to 20, iterations max-number to 1000, $\epsilon_1 = 10^{-20}$ and $e = 1.9$. In addition, in order to have a fair comparison with Power method, we set "M_1 = initialization number = 20" and "$M_2 = 5 \times$ iterations max-number" in Algorithm 4, where 5 is due to the five times of doing the loop of line 6 in Algorithm 5 (the same loop in AO-ADMM is also done five times).

It can be inferred from Table 5.2 that all the considered methods perform well in noiseless case, specially Projected Power method. These results show that our implementation of Power method is reliable.

Although Power method and Projected power method perform very well in the noiseless case, Fig. 5.23 and Fig. 5.24 show that these algorithms are very sensitive to additive noise in $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$. Note that in these figures SFBS, SFBS-Symm, ADMM, Power and Power-proj refer to Simplex SFBS, Symmetric Simplex SFBS, Simplex AO-ADMM, Power method and Projected

Table 5.2: Generating some arbitrary matrices $\boldsymbol{A}^{10\times 3}$ and vectors $\boldsymbol{\varphi}^{3\times 1}$, then calculating their corresponding $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$, averaging the results of decomposing *noiseless* $\boldsymbol{\mathcal{T}}$ over 200 realizations of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$.

Algorithm	Reconstruction	Error of $\boldsymbol{A}$	Error of $\boldsymbol{\varphi}$
Power method	2.97 e–15	5.25 e–16	3.68 e–16
Projected Power method	9.19 e–16	5.10 e–16	7.70 e–16
Simplex AO-ADMM	4.11 e–5	1.50 e–4	2.40 e–4
Simplex SFBS	1.57 e–15	4.97 e–15	8.39 e–15
Symmetric Simplex SFBS	1.52 e–15	4.97 e–15	8.39 e–15

Power method, respectively.

All the settings in the experiment of Fig. 5.23 and Fig. 5.24 are the same as that of Table 5.2, except that $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{P}$ are perturbed by additional Gaussian noise at a specific SNR (cf. (5.20)). Figure 5.23 (Left) shows the relative reconstruction error of algorithms versus a range of SNR values, *i.e.* $[10, 20, 30, 40]$. As it can be seen, Power method is highly sensitive to the additive noise, and its *relative* reconstruction error can reach very large values (such as 500). In Fig. 5.23 (Right), so as to distinguish the performances of other algorithms, we remove Power method. Figure 5.23 (Right) expresses that even by projecting the result of Power method to the simplex set (Projected Power method), the performance of Power method cannot become acceptable.

The relative error of estimation of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ are depicted in Fig. 5.24 (Left) and Fig. 5.24 (Right), respectively. As it is demonstrated in Fig. 5.24, Power method and Projected Power method does not perform as well as others such as SFBS. Moreover, Simplex SFBS and Symmetric Simplex SFBS perform a bit better than AO-ADMM. Note that unlike reconstruction error, the relative error of the estimation of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ is limited in the range of $[0, 2]$, since we normalize the $\boldsymbol{\varphi}$ and the columns of $\boldsymbol{A}$ with L_2 norm before measuring the performance index.

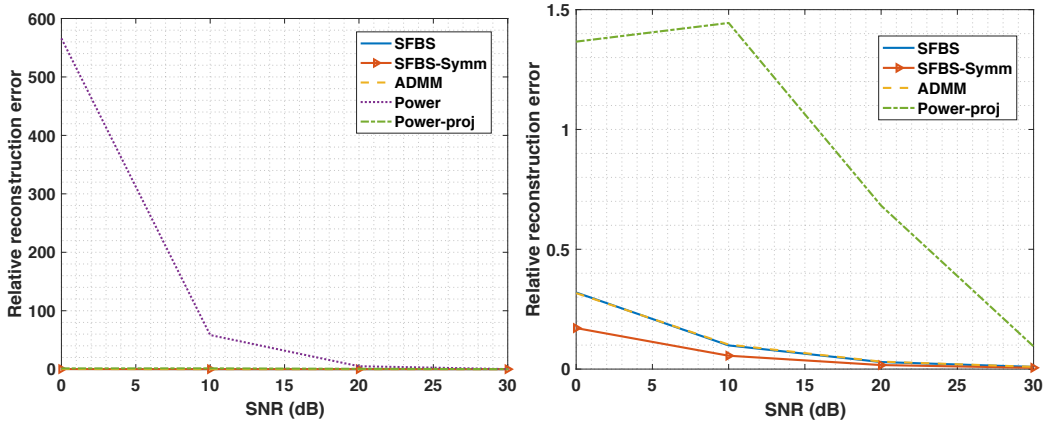

Figure 5.23: Left: The relative reconstruction error of decomposing third order moments including Power method. Right: zoom of the left one, excluding Power method. Averaging the results of decomposing *noisy* $\boldsymbol{\mathcal{T}}$ according to a range of SNR values, *i.e.* $[10, 20, 30, 40]$ over 200 realizations of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$, reveals that constrained algorithms such as SFBS perform much more better than Power method and its variants.

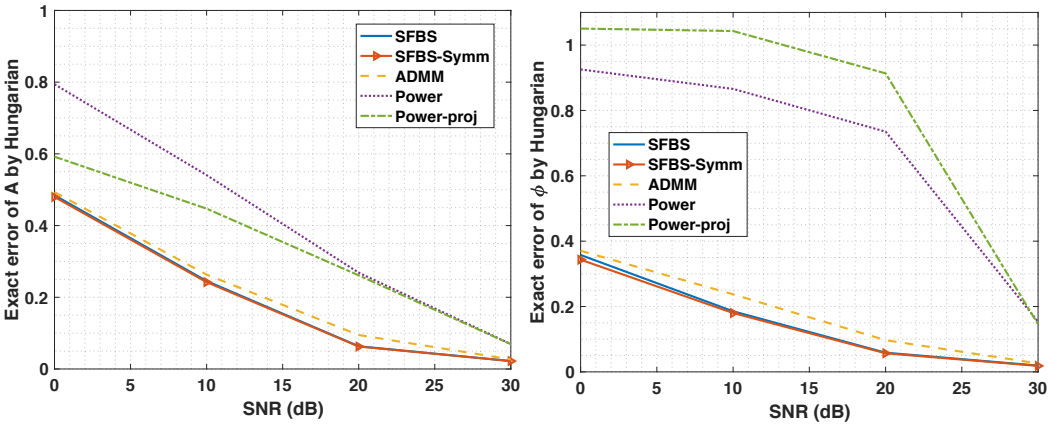

Figure 5.24: Left: The relative error of the estimation of $\boldsymbol{A}$ measured by Hungarian algorithm in the same experiment of Fig. 5.23. Right: The relative error of the estimation of $\boldsymbol{\varphi}$ measured by Hungarian algorithm in the same experiment of Fig. 5.23. Averaging the results of decomposing *noisy* $\boldsymbol{\mathcal{T}}$ according to a range of SNR values, *i.e.* $[10, 20, 30, 40]$ over 200 realizations of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$, reveals that constrained algorithms such as SFBS perform much more better than Power method and its variants. In addition, Symmetric Simplex SFBS and Simplex SFBS also perform a bit better than AO-ADMM.

5.6.2.2 Compare performances for synthetic data with hidden relation

In this section, we evaluate the performance of several algorithms in estimating probabilities for synthetic data with hidden relation. Note that in Section 5.6.2.1, we generated synthetic $\boldsymbol{A}$ and $\boldsymbol{\varphi}$, then we obtained their corresponding $\boldsymbol{\mathcal{T}}$ according to (3.6), and finally, we fed the algorithms with the noiseless or the noisy version of $\boldsymbol{\mathcal{T}}$.

Nevertheless, in this section, we generate some synthetic hidden (topics) and multi-view variables (words), then we estimate the third order moments tensor by means of moment estimators (cf. Section 3.4) and by observing the generated synthetic variables. Finally, we feed the algorithms with the estimated moments. The comparison of AO-ADMM and Robust tensor power method is also investigated in our paper [SCJBZ19].

As explained in Section 3.4.1, such a synthetic data can be generated by means of two different generative processes. In addition, as mentioned in Section 3.4.2, it can be easily shown that (3.3) and (3.4) are respectively equal to (3.5) and (3.6). Therefore, the sample estimates of (3.3) and (3.4) should converge to the true moments $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ defined in (3.5) and (3.6). To carry out this sample estimate, some moment estimators such as simple averaging are reviewed in Section 3.4.3, as well as our proposed estimator, standard averaging, in Section 3.4.4.

Hence, it is essential to check the consistency (cf. see Section 3.4.2 for a definition of consistency) of generated data along with moment estimator in order to make sure that sample moments indeed converge to the true joint probabilities.

The effect of constraint on the performance. In this experiment, we experimentally show the effect of considering a constraint on the precision of probability estimation. In other words, we shall compare the performance of non-constrained tensor decomposition such as Robust tensor power method with the constrained one, such as AO-ADMM and SFBS, in terms of their

precision in estimating probabilities.

As Robust tensor power method is introduced along with simple averaging, firstly, we investigate the consistency of simple averaging on a synthetic data generated by the generative process described in 3.4.1.1. To this end, we generated some corpuses of size N_c (*i.e.* each corpus contains N_c documents), where each document is constructed of synthetic words, $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma(\ell)}$, (in a way explained in 3.4.1.1) with the following parameter values: $K = 4, D = 8$ and different amounts of N_c. We chose a uniform distribution for φ, and K arbitrary distributions to build a synthetic matrix $\boldsymbol{A}$. Lastly, we calculated sample moments in (3.7) and (3.8) (*i.e.* simple averaging), and compared them to the true ones (3.5) and (3.6). Fig. 5.25 reports the discrepancy between both in terms of Euclidean norm. As can be seen in Fig. 5.25, the generated data in the way described in 3.4.1.1 along with simple averaging are consistent for $N_c > 10000$.

As mentioned before, the main goal is the estimation of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$. Next, we report in this section the comparison results obtained with only one dictionary size, $D = 8$, with a fixed number of topics, $K = 4$, and for various corpus size up to $N_c = 2^{17}$. Robust tensor power method is run with $M_1 = 10$ (the number of iterations over each mode in line 3 of Algorithm 4) and $M_2 = 5$ (the number of power iteration updates in line 5 of Algorithm 4) as proposed by the authors, and the non-negative AO-ADMM with initialization number set to 100. Moreover, the results are obtained for one particular realization of matrix $\boldsymbol{A}$ and vector $\boldsymbol{\varphi}$ (however similar results are obtained with other

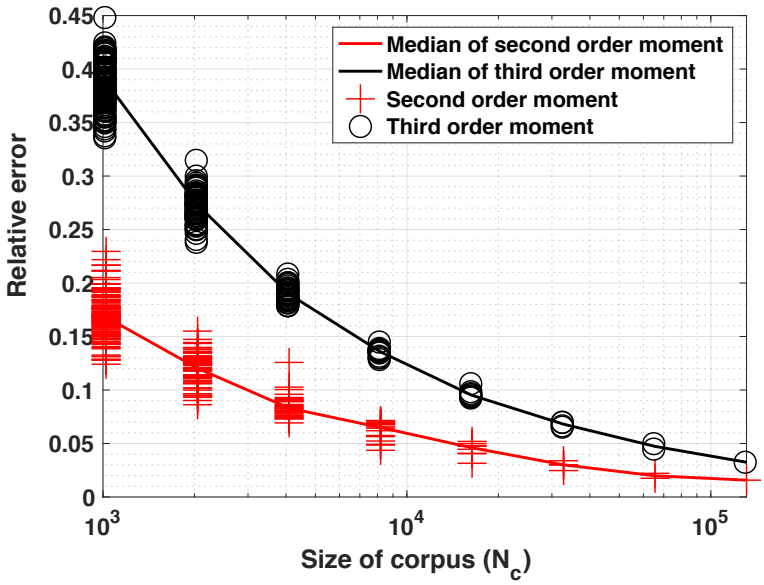

Figure 5.25: Moment consistency (cf. Section 3.4.2 for the definition of consistency). Each black circle (resp. red cross) corresponds to the discrepancy between sample third order moments (resp. sample second order moments) and true moments for a specific corpus of documents with a particular corpus size, N_c. The median among corpuses of documents is also plotted in solid line for every corpus size. Data generated by the generative process described in 3.4.1.1 are consistent for $N_c > 10000$. Sample moment computation is carried out via simple averaging. The relative error is reported according to (5.21), and it is not in the form of percentage.

choice of $\boldsymbol{A}$ and $\boldsymbol{\varphi}$):

$$\boldsymbol{A} = \begin{bmatrix} 0.162 & 0.211 & 0.000 & 0.000 \\ 0.082 & 0.130 & 0.000 & 0.000 \\ 0.022 & 0.235 & 0.000 & 0.403 \\ 0.196 & 0.423 & 0.000 & 0.038 \\ 0.174 & 0.000 & 0.276 & 0.119 \\ 0.104 & 0.000 & 0.133 & 0.439 \\ 0.113 & 0.000 & 0.119 & 0.000 \\ 0.147 & 0.000 & 0.473 & 0.000 \end{bmatrix}, \boldsymbol{\varphi} = \begin{bmatrix} 0.256 \\ 0.163 \\ 0.201 \\ 0.380 \end{bmatrix}.$$

In order to compare the performance of both methods, we run this simulation for increasing the corpus size (note that the larger the corpus size,

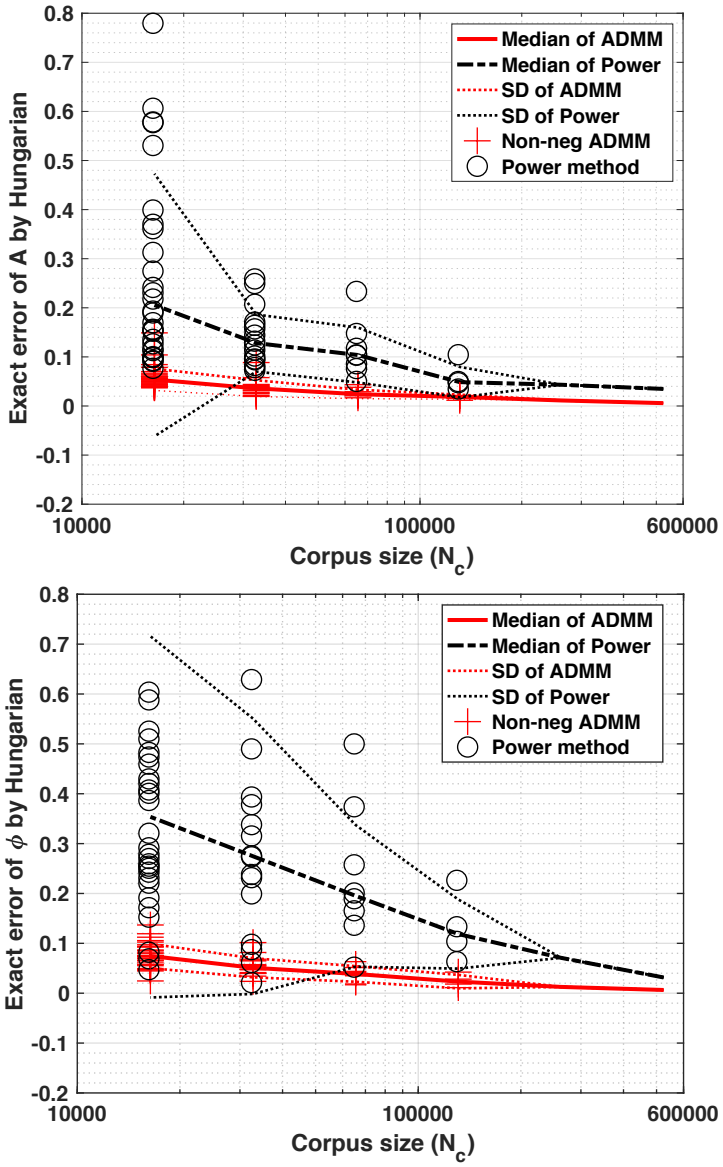

Figure 5.26: The comparison of performances of Robust tensor power method and non-negative AO-ADMM in estimating probabilities, *i.e.* $\boldsymbol{A}$ (top) and $\boldsymbol{\varphi}$ (bottom), versus corpus size up to $N_c = 2^{17}$, when the moment estimator is simple averaging. The number of words and topics are fixed to $D = 8$ and $K = 4$, respectively. Each black circle (resp. red cross) corresponds to the error of Robust tensor power method (resp. non-negative AO-ADMM) in estimating $\boldsymbol{\varphi}$ or $\boldsymbol{A}$ for a specific corpus of documents with a particular corpus size, N_c. The median (resp. standard deviation) among corpuses of documents is also plotted in solid (resp. dotted) line for every corpus size.

the fewer the number of documents, to keep a computational load constant). To be more precise, by keeping the values of parameters $D = 8, K = 4$, we increase exponentially the value of N_c from 2^{14} to 2^{17}. For each value of N_c, we generated several synthetic data sets of size N_c in the way that has been explained in Subsection 3.4.1, and then the methods have been applied to the obtained data to compare relative errors as a function of corpus size.

The results are reported in Fig. 5.26; each black circle (resp. red cross) corresponds to the error of Robust tensor power method (resp. non-negative AO-ADMM) in estimating $\boldsymbol{\varphi}$ or $\boldsymbol{A}$ for a specific corpus of documents with a particular corpus size, N_c. To ease comparison, the median (resp. standard deviation) among corpuses of documents is also plotted in solid (resp. dotted) line for every corpus size.

As Fig. 5.26 shows, the error of constrained method such as non-negative AO-ADMM converges faster to zero as the corpus size increases, and unlike Robust tensor power method, non-negative AO-ADMM appears to be much more robust because of its smaller standard deviation of the relative error. Moreover, Fig. 5.26 (bottom) reveals that the difference in the performance and standard deviation of Robust tensor power method is much larger in estimating $\boldsymbol{\varphi}$. Therefore, taking into account a proper constraint (such as non-negativity) in the decomposition is very effective, especially in the estimation of $\boldsymbol{\varphi}$.

Results for random $\boldsymbol{A}$ and $\boldsymbol{\varphi}$. When running extensive computer simulations, one can assume a uniform distribution for the hidden variables (topics), and K uniform distributions for the conditional probability of multi-view variables (words) given the topic, which are actually the columns of $\boldsymbol{A}$. Once $\boldsymbol{A}$ and $\boldsymbol{\varphi}$ are obtained, one can proceed exactly the same way as explained in Subsection 3.4.1 to generate synthetic datasets and compute sample moments via simple averaging.

In Table 5.3 the results obtained for a particular value of K and a few values of D, averaged over 20 independent trials. These extensive computer

experiments confirm the robustness and superiority of constrained algorithms such as non-negative AO-ADMM compared to Robust tensor power method.

Improve performance by standard averaging and simplex set constraint. We can modify the experiment of Fig. 5.26 from different aspects. As expressed in Section 3.4.5, the sample moment estimated by standard averaging requires a smaller corpus size compared to simple averaging to converge to the true moments. In addition, as we seek the probabilities, which belong to the simplex set, the decomposition would be more precise if it is carried out under the simplex set constraint for $\boldsymbol{\varphi}$ and the columns of $\boldsymbol{A}$.

Therefore, one can feed decomposition algorithms mentioned in page 118 by the sample moments approximated via standard averaging. Moreover, it is preferred to perform constrained algorithms such as AO-ADMM and SFBS under the simplex set constraint. These modifications have been done in the experiment of Fig. 5.27.

The experiment of Fig. 5.27 is performed for $K = 3$ (hidden variables (topics)), $D = 10$ (multi-view variables (words)) and various corpus size up to $N_c = 2^{10}$. The synthetic data for this experiment are generated via the generative process described in Subsection 3.4.1.2, which is based on multinomial distribution. The considered corpuses are consist of documents with minimum and maximum length of 3 and 100, respectively. We set initialization number to 20, iterations max-number to 1000, $\epsilon_1 = 10^{-20}$ and $e = 1.9$. In addition, in order to have a fair comparison with Power method, we set "M_1 = initialization number = 20" and "$M_2 = 5 \times$ iterations max-number" in Algorithm 4, where 5 is due to the five times of doing the loop of line 6 in Algorithm 5 (the same loop in AO-ADMM is also executed five times).

Figure 5.27 (top) compares the performances of algorithms in estimating $\boldsymbol{A}$ versus the corpus size. As it can be seen in Fig. 5.27 (top), the performances of all algorithms except Power method are the same, which shows that the projection to the simplex set or considering this constraint in the

Table 5.3: The median and standard deviation of error in estimating $\boldsymbol{A}$ and $\boldsymbol{\varphi}$, with a corpus of size $N_c = 2^{17} \approx 1.2 \times 10^6$, and $K = 4$. In each cell, top: non-negative AO-ADMM, bottom: Robust tensor power method. As it can be seen, the median and standard deviation of the error of non-negative AO-ADMM is always less than those of Robust tensor power method.

Median of relative error in estimating $\boldsymbol{A}$

D	6	7	8	9	10
Non-negative AO-ADMM	0.09	0.08	0.07	0.10	0.07
Robust tensor power method	0.38	0.18	0.24	0.18	0.12

Standard deviation of relative error in estimating $\boldsymbol{A}$

D	6	7	8	9	10
Non-negative AO-ADMM	0.10	0.12	0.08	0.10	0.12
Robust tensor power method	0.31	0.31	0.39	0.33	0.23

Median of relative error in estimating $\boldsymbol{\varphi}$

D	6	7	8	9	10
Non-negative AO-ADMM	0.14	0.10	0.11	0.10	0.13
Robust tensor power method	0.66	0.28	0.31	0.29	0.23

Standard deviation of relative error in estimating $\boldsymbol{\varphi}$

D	6	7	8	9	10
Non-negative AO-ADMM	0.22	0.12	0.21	0.28	0.31
Robust tensor power method	0.54	0.52	0.59	0.41	0.40

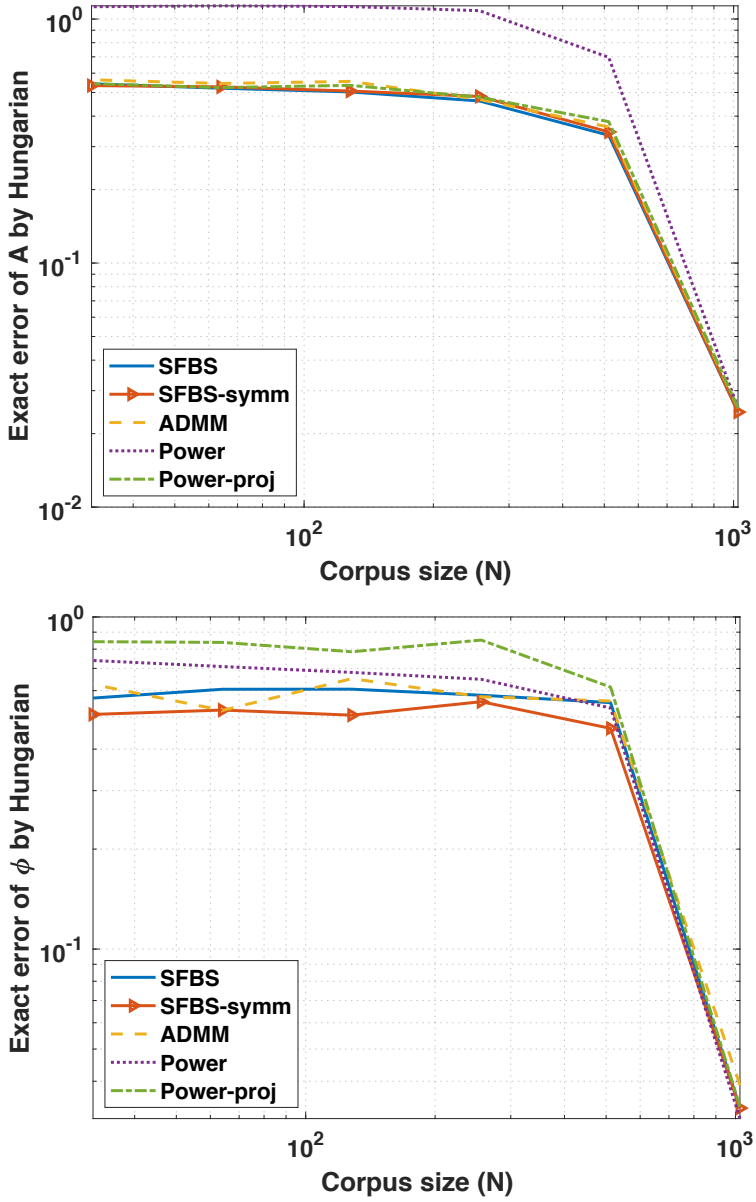

Figure 5.27: The comparison of performances in estimating probabilities, *i.e.* $\boldsymbol{A}$ (top) and $\boldsymbol{\varphi}$ (bottom), versus corpus size up to $N_c = 2^{10}$, when the moment estimator is standard averaging. The number of multi-view variables (words) and the number of hidden variables (topics) are fixed to $D = 10$ and $K = 3$, respectively. Robust tensor power method is run with $M_1 = 5000$ and $M_2 = 20$, and the initialization number and iterations max-number for constrained algorithms is 20 and 1000, respectively.

decomposition will end in more precise estimations compared to pure Power method.

Figure 5.27 (bottom) compares the performances of algorithms in estimating $\boldsymbol{\varphi}$ versus the corpus size. Unlike the estimation of $\boldsymbol{A}$, the performance of different algorithms are not the same. Actually, the performance of Projected Power method is the worst, while the relative error of $\boldsymbol{\varphi}$ by Symmetric Simplex SFBS is the least among other methods. Therefore, Symmetric Simplex SFBS outperforms other methods in decomposing the tensor of third order moments.

Let us remark two other points that can be inferred from Fig. 5.27. First, by comparing Fig. 5.26 and Fig. 5.27, we can observe easily the effect of moment estimators. In Fig. 5.27, which is based on standard averaging, a corpus of size 1024 is sufficient for the moment consistency, hence, for an acceptable estimation of probabilities. Nevertheless, in Fig. 5.26, a corpus of size $N_c > 10000$ is necessary to have a reasonable performance, since the moment estimator is simple averaging. Second, for the smallest considered corpus size in Fig. 5.27, *i.e.* $N_c = 32$, the SNR $= 7.62$, which shows that our assumption on the range of SNR in the synthetic data with hidden relation (Fig. 5.23 and Fig. 5.24) was reasonable.

5.6.3 Real data

In this section, we describe our experiment on a part of a well-known text data set, namely 20 Newsgroups, which consists of 11314 posts on 20 topics available online [Lan95a, Lan95b, Nig00].

It is vey common to do some pre-processing steps in order to extract the keywords of the corpus described above. standard steps of pre-processings are as follows: removing stop words (including me, you, a, the, which, where, ...), tokenizing (transform a text into a list of words), removing punctuations and unnecessary characters, lemmatizing (replacing with the infinitive, *e.g.* replacing "took" with "take"). At the end, according to the frequency of each word (called *term-document frequency*) in the corpus, a portion of

words are kept as the representatives of the corpus also known as vocabulary or dictionary.

By doing above mentioned pre-processing[15] steps on the whole data set and by keeping the words with more than 20% of the term-document frequency, we are left with a dictionary of size 14×14 ($D = 14$ words). Since, Power method and its variants cannot handle over-complete cases ($K > D$), we are forced to choose $K = 14$ topics rather than $K = 20$, though $K = 20$ is more suitable according to 20 topics exist in the considered data set ($N_c = 11314$ posts on 20 topics). Therefore, the resulted bag-of-words (cf. (3.9)) is a matrix of dimensions 14 by 11314.

After obtaining the bag-of-words, we fed the tensor decomposition algorithms listed in page 118 with the estimated $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ via standard averaging. In addition, we set the input rank of all the algorithms to $K = 14$ due to the limitation of Power method and its variants. Eventually, by each algorithm, a pair of estimated probabilities $(\boldsymbol{\varphi}^{14}, \boldsymbol{A}^{14\times 14})$ is acquired.

Although in this corpus, the number of documents for all the topics are almost the same (which means that $\boldsymbol{\varphi}$ is expected to have a distribution close to the uniform distribution), the estimated $\boldsymbol{\varphi}$ by Robust power method, *i.e.* $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$, is not only non-uniform, but also it is not a probability distribution at all, since it does not lie in the probability simplex:

$$\begin{aligned}\widehat{\boldsymbol{\varphi}}_{\text{Power}} = [&0.00, 0.02, 0.14, 0.03, 0.08, 0.26, 0.39,\\ &0.35, 0.65, \mathbf{302.89}, 1.01, 0.87, 8.30, 0.95]^T.\end{aligned}$$

As it can be observed in $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$, its tenth element dominates the others which could be the consequence of rounding errors. Even if one projects $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$ to the simplex set as mentioned before, and obtains $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$, it would be meaningless according to the fact that "$\boldsymbol{\varphi}$ is expected to have a distribution close to the uniform distribution", since we have:

$$\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}} = \Big[0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0\Big]^T.$$

[15] We did these steps by means of corresponding packages such as "`nltk`", libraries such as "`Gensim`" and an implemented example available online [Pra18].

On the other hand, the estimated $\boldsymbol{\varphi}$ by SFBS, *i.e.* $\widehat{\boldsymbol{\varphi}}_{\text{SFBS}}$, is as follows:

$$\widehat{\boldsymbol{\varphi}}_{\text{SFBS}} = [0.10, 0.08, 0.09, 0.09, 0.10, 0.04, 0.02, \\ 0.05, 0.06, 0.01, 0.06, 0.09, 0.08, 0.12]^T,$$

which is closer to uniform distribution than $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$. This again proves that applying constrained tensor decomposition is more reliable than non-constrained one.

In Section 5.6.2.1, we investigated two drawbacks of Robust tensor power method, namely negative estimated probabilities and sensitivity to the additive noise. Nevertheless, in this experiment, we detect a much more important drawback in employing Power method for real text data set, that is, the *dominant estimated element*.

Although Projected power method resolves the problem of negative estimated probabilities by projecting the estimated probabilities to the simplex set, it is incapable to solve the problem of the dominant estimated element. Therefore, utilizing non-constrained tensor decompositions such as Power method and its variants is not a good solution in practice, however these kinds of methods are considered as fast solutions for estimating probabilities.

In order to be able to compare the performances of tensor decomposition algorithms on this real text data set, we need the ground truth values of the probabilities of topics, *i.e.* $\boldsymbol{\varphi}$, and the conditional probabilities of each word given the topic, *i.e.* $\boldsymbol{A}$. To do this, for the sake of convenience, we worked with a portion of data set introduced at the beginning of this section. We selected documents about four different topics, namely "computer graphics" (labeled by `comp.graphics`), "baseball" (labeled by `rec.sport.baseball`), "cryptography" (labeled by `sci.crypt`) and "Christianity" (labeled by `soc.religion.christian`). Then, by applying above mentioned pre-processings on this portion of data, which contains 2375 documents, and by keeping the words with term-frequency between 20% and 50%, we had a dictionary with $D = 17$ words. As we desired to employ

Ruffini's estimator in this experiment, it was necessary to eliminate the documents including less than three words. By doing that, 1690 documents remained in the corpus. So as to calculate the ground truth values of $\boldsymbol{\varphi}$, we counted the number of documents belong to each topic (*i.e.* "computer graphics", "baseball", "cryptography" and "Christianity"). Then, the ground truth of $\boldsymbol{\varphi}$ is simply the division of the number of documents in each topic by 1690. To calculate the ground truth of $\boldsymbol{A}$, for each word in the obtained dictionary, we counted its occurrences in the documents belong to a particular topic, and then we divided it by the total number of words used in those documents (which belong to that particular topic).

As in the previous experiment, after obtaining the bag-of-words, we fed the tensor decomposition algorithms listed in page 118 with the estimated $\boldsymbol{P}$ and $\boldsymbol{\mathcal{T}}$ via the Ruffini's estimator. We set the input rank of all the algorithms to $K = 4$, and eventually, we computed the performance of each algorithm by comparing estimated probabilities $(\widehat{\boldsymbol{\varphi}}^4, \widehat{\boldsymbol{A}}^{17\times 4})$ with the ground truth values of $\boldsymbol{\varphi}$ and $\boldsymbol{A}$.

The performances of each algorithm of this experiment in estimating probabilities $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ based on CorrIndex are reported in Table 5.4. As it can be seen in Table 5.4, the performances of Power method and Projected Power method in estimating $\boldsymbol{\varphi}$ are much worse than other constrained algorithms. However, their performances in estimating $\boldsymbol{A}$ are a bit better than others. In addition, all the constrained algorithms such SFBS perform properly in both estimating $\boldsymbol{\varphi}$ and $\boldsymbol{A}$.

The fact that Power method performs better in estimating $\boldsymbol{A}$ than in estimating $\boldsymbol{\varphi}$ is probably due to the different manner of updating $\widehat{\boldsymbol{\varphi}}$ from updating $\widehat{\boldsymbol{A}}$ in Power method. As it can be observed in line 6 of Algorithm 4, the L_2 norm of the columns of $\widehat{\boldsymbol{A}}$ are stored as the estimation of $\boldsymbol{\varphi}$. In this way, a normalized matrix is reported as an estimation of $\boldsymbol{A}$, which is protected relatively from rounding errors. However, the non-normalized $\widehat{\boldsymbol{\varphi}}$ may suffer from rounding errors.

Table 5.4: Performances in estimating probabilities $\boldsymbol{\varphi}$ and $\boldsymbol{A}$ in terms of CorrIndex: the smaller the better. This experiment is carried out using a part of a well-known text data set, namely 20 Newsgroups. To be more precise, documents of the selected data set belong to four topics: "computer graphics", "baseball", "cryptography" and "Christianity".

Algorithm	CorrIndex($\boldsymbol{\varphi}, \widehat{\boldsymbol{\varphi}}$)	CorrIndex($\boldsymbol{A}, \widehat{\boldsymbol{A}}$)
Power method	0.577	0.077
Projected Power method	0.703	0.077
Simplex SFBS	0.106	0.097
Symmetric Simplex SFBS	0.106	0.097
Simplex AO-ADMM	0.102	0.089

As the words in the obtained dictionary of this experiment[16] are not informative enough, it seems impossible to do an unsupervised clustering based on the estimated probabilities $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{\varphi}}$. However, extracting informative words depends on employing proper pre-processing steps, which is out of the scope of this thesis.

5.7 Discussion

According to the performed experiments and theoretical arguments, in this section, we wrap up the advantages and disadvantages of each method compared to our proposed method, SFBS.

Concerning constrained algorithms, a theoretical drawback of AO-ADMM is that the convergence of AO is not guaranteed because of ignoring the proximal regularization (cf. Subsection 5.3.2). In addition, for the non-convex constraint ℓ_0, the convergence of ADMM has not yet been proved, whereas the complete convergence of SFBS is studied in Section 5.5.4. However, its

[16] also, come, even, find, get, give, go, good, make, need, nntp-poste, people, see, take, time, way, well.

performance is as well as that of SFBS in most cases, such as noiseless and noisy scenarios.

By comparing APG and SFBS, we can conclude that SFBS performs better than APG in noisy cases (cf. Fig. 5.9), while it is also computationally less expensive, due to not calculating any coefficient, unlike APG, which is based on FISTA. Moreover, some methods such as BC-VMFB forces the user to adjust several parameters that are very effective on the result, while the only parameter of SFBS is e, for which we provided a proper range ($[1.5, 1.9]$).

Furthermore, unlike other constrained algorithms, we discussed how SFBS can handle a variety of constraints, namely non-negativity, the simplex set (for all loading factors plus the vector of coefficient), sparsity (ℓ_0 pseudo-norm and ℓ_1 norm).

Concerning non-constrained algorithms, SFBS is much more robust against additive noise than Nway or Power method. Moreover, Power method and its variants (such as SS-HOPM) does not have any convergence guarantee for an odd-order tensor (including the tensor of third order moments), whereas the complete convergence of SFBS is studied in Section 5.5.4.

Focusing on data (text) mining based on the method of moments and by means of tensor decomposition, there is an intrinsic noise due to the discrepancy between true moments and sample moments. Therefore, Power method and its variants, which are not robust enough to the noise will face a problem. Moreover, it is also possible that Power method or its variants return non-acceptable negative values for estimated probabilities. Even if one projects the result of Power method to the simplex set (*i.e.* Projected power method), its performance in estimating $\boldsymbol{\varphi}$ would be the worst among others (cf. Fig. 5.27 (bottom)). More importantly, as it is shown experimentally in Section 5.6.3, another critical drawback of Power method in face of real text data set is the dominant estimated element (*i.e.* a very large element in $\boldsymbol{\varphi}$, which drops out $\boldsymbol{\varphi}$ from the simplex set), and this problem cannot be solved with projection to the simplex set by Projected Power method.

In addition, in order to decompose the tensor of third order moments,

Power method requires also second order moments for the whitening step. Furthermore, since after whitening, Power method performs a kind of orthogonal tensor decomposition, hence, one of its limitations is that it can only handle under-complete cases (*i.e.* when the number of hidden variables, K, is smaller than that of multi-view ones, D).

However, SFBS is robust against additive noise (especially in estimating $\boldsymbol{\varphi}$, cf. Fig. 5.27 (bottom)), returns the estimated probabilities in the simplex set thanks to its constraint, requires only the tensor of third order moments as input, and is able to handle over-complete cases ($K > D$), which is necessary in some applications mentioned in [AHJK15]. The only drawback of constrained algorithms such as SFBS compared to the Power method (and especially SVTD, which is faster than the Power method) is that they are more time consuming than Power method.

As a last point, the role of moment estimator should always be taken into account, since it has a significant effect on decreasing the intrinsic noise and decreasing the required data for consistency. Even the best decomposition algorithm may not return acceptable results without being provided a good estimation of moments as its input.

5.8 Conclusion and perspective

In this chapter, we investigated constrained and unconstrained tensor decompositions. As mentioned before, the main goal is to estimate probabilities by decomposing the third order moments tensor, and in the literature, usually some unconstrained tensor decompositions such as Robust tensor power method are employed for this purpose.

We showed theoretically and experimentally that such a unconstrained algorithms are not proper, since they probably return negative values, which are not acceptable as probabilities (in this vein, we proposed Projected Power method to solve this problem by projecting the results of Power method to the simplex set); or they are very sensitive to additive noise and are unable

to handle over-complete cases, where the number of hidden variables is more than the number of multi-view ones. More importantly, we experimentally showed that in employing Robust tensor power method on real text data set, a very challenging problem arises, namely the dominant estimated element drops out estimated probability vector ($\boldsymbol{\varphi}$) from the simplex set. Moreover, this issue cannot be solved by projecting onto the simplex set.

On the other hand, we showed that constrained tensor decompositions return more acceptable estimations, since some constraints are taken into account, such as non-negativity and simplex set. We also introduced our constrained tensor decomposition, SFBS, which is based on Forward-Backward Splitting.

We compared theoretically and experimentally SFBS with other algorithms, and we concluded that SFBS performs either better than or as well as AO-ADMM. Furthermore, there exists a complete convergence proof for SFBS, which is not the case for AO-ADMM due to ignoring the proximal regularization (which is essential for the convergence guarantee of AO) and due to the lack of theoretical proof for the non-convex constraint ℓ_0.

We also explained that SFBS performs better than APG in noisy cases and computationally costs less than APG because of not calculating any coefficient, unlike APG. In addition, we discussed that not only BC-VMFB performs poorly in decomposing tensors, but also it forces the user to adjust several parameters, which are so effective on the result of this method. Yet, SFBS consists of just one parameter (e), which is easy to set according to the convergence condition and the spectral norm of the corresponding matrix (*e.g* for the experiments of this section we proposed a range of $[1.5, 1.9]$).

Compared to unconstrained algorithms such as Robust tensor power method, not only SFBS performs better, but also it has the theoretical convergence guarantee. Moreover, concerning the simplex set constraint, one can assure that probabilities estimated by SFBS always lie in the probability simplex. Further, SFBS is much more robust against additive noise, which is a very important feature in data (text) mining where there exists

intrinsic noise due to the discrepancy between true moments and sample moments. However, non-constrained algorithms, especially SVTD, converge much more faster than iterative algorithms such as SFBS.

In addition to the intrinsic noise, working with large dimension third order moments tensors is another big challenge of data mining by means of tensor decomposition. Recall that the dimension of this tensor depends on the number of multi-view variables (words) in the dictionary, which can be very large in a real world text mining problem (around 5000). Unfortunately, most of iterative constrained algorithms such as SFBS are very time consuming, hence, intractable in practice.

As the most time consuming part in tensor decomposition algorithms is the calculation of Khatri-Rao product of loading matrices, several methods have been proposed to reduce this computational complexity by sketching [KBGP18] the data required for the calculation of Khatri-Rao product in each iteration. One of the most successful methods in this vein is CPRAND [BBK18], which is a non-constrained CP decomposition and is proper for large dimensions, since it employs sketching data for the calculation of Khatri-Rao product, called sample Khatri-Rao.

Recently, a constrained CP decomposition for large dimensions based on Forward-Backward Splitting and the sketching concept has been introduced in [FIW$^+$20]. Although, it is claimed that this method is capable of handling various types of constraints, but we find it inefficient for the simplex set constraint, in spite of using the implementation provided by the authors.

Therefore, a future extension of SFBS would be employing CPRAND idea (sample Khatri-Rao) in each iteration of SFBS. In this way, we will have a constrained CP decomposition dedicated for large dimension tensors, which handles properly constraints such as the simplex set constraint.

6 Conclusion and Perspectives

Vectors and matrices as one-way and two-way arrays, respectively, can be generalized to multi-way arrays or *tensors*. Tensors are known to be useful to identify parameters, thanks to the mild uniqueness conditions required in Canonical Polyadic tensor decomposition (3.6).

In this thesis, we focused on a particular data model, called single topic (cf. Fig. 3.1), consisting of some multi-view variables which are related to each other via a hidden/latent variable. A document can be described properly by this model when its words role as some multi-view variables, which are related to each other according to the topic of that document as a hidden variable.

Estimating the probabilities of hidden variables, and also the conditional probabilities of multi-view ones, are in the center of interest, since they can be utilized for an ultimate data/text mining task such as unsupervised clustering. On the other hand, these probabilities correspond to the tensor of third order moments of observed data (*e.g.* words) via a particular tensor format, called Canonical Polyadic (CP); see (3.6) for details.

In this thesis, generally, we investigated CP decomposition applications in estimating the probabilities of hidden variables and the conditional probabilities of multi-view variables.

In chapter 3, we explained that there are two main steps in estimating the probabilities via a tensor approach: firstly, estimate the third order moments tensor by observing data which obeys the single topic model, secondly, decompose the estimated third order moments tensor by means of proper tensor decomposition algorithms to obtain some reliable and accu-

rate estimations of probabilities that can be used efficiently for an ultimate data/text mining task such as unsupervised clustering.

At the end of chapter 3, we reviewed some existing moment estimators, and criticized simple averaging, as a relevant moment estimator in the literature, by experimentally showing its weakness in providing moment consistency even on sufficiently large size corpuses.

As our moment estimator, we proposed standard averaging, which performs much better than simple averaging, due to taking into account all the words of each document as well as the length of each document. In this way, we control the importance of each document in the estimation. Further, compared to the state-of-the-art, standard averaging is based on a simpler concept (weighted averaging), and is computationally less expensive in terms of number of multiplications for estimating third order moments. In addition, the number of required additions for estimating second order moments does not depend on the dictionary size.

In Chapter 4 and 5, we discussed about the second stage, *i.e.* tensor decomposition algorithms, and also about required performance indices to evaluate these different methods of decompositions. Recall that permutation and scale ambiguities are inherent in tensor representations, by definition of tensors [Com14]. Therefore, in order to cancel the effect of these ambiguities in measuring the performance, one needs to employ proper performance indices.

Chapter 4 is devoted to performance indices in which we carried out a critical survey about the state-of-the-art. We showed theoretically and experimentally that existing indices are either greedy (optimistic or pessimistic) or computationally expensive. Moreover, we proposed a new performance index belonging to the class of invariant indices, called CorrIndex, whose upper and lower bounds are easy to interpret, while being computationally cheap. We compared tensor decomposition algorithms in Chapter 5 by means of CorrIndex.

In Chapter 5, we investigated constrained and unconstrained tensor de-

compositions to discover their advantages and disadvantages especially in decomposing the tensor of third order moments to estimate probabilities. We experimentally showed that although Power method and its variants are relevant methods for this purpose, they are sensitive to additive noise and not capable of handling over-complete cases, where the number of hidden variables is larger than multi-view variables. More importantly, the probabilities estimated by these unconstrained algorithms may include negative values or dominant elements, which exclude estimated probabilities out of the simplex set. One may think of projecting the estimated probabilities to eliminate negative values but this solution is not useful for the problem of dominant estimated element.

On the other hand, we proposed to employ constrained tensor decompositions such as AO-ADMM under non-negative or simplex set constraints in order to have much more accurate estimations of probabilities. In this vein, we proposed our constrained tensor decomposition, called Simple Forward-Backward Splitting (SFBS), using Forward-Backward Splitting, a minimization procedure based on proximal concept. The term "simple" shows two aspects of our algorithm: first, compared to state-of-the-art, it is less computationally costly, second, its algorithm is much easier to understand and implement. Furthermore, we described how SFBS can handle a variety of constraints. Moreover, a complete convergence analysis is provided, which is not possible for AO-ADMM. More importantly, SFBS performs better than state-of-the-art in noisy scenarios while being computationally cheap.

In summary, we investigated each required step of probability estimation via a tensor approach. We studied theoretically and experimentally the challenges of state-of-the-art in moment estimations and tensor decompositions. Moreover, we proposed our moment estimator (standard averaging), our performance index (CorrIndex) and our tensor decomposition algorithm (SFBS), which bring some advantages over existing methods and indices.

As mentioned at the end of Chapter 5, the main disadvantage of constrained tensor decomposition algorithms for large dimension (typically of

the order $D > 50$) is that they are time consuming due to the calculation of Khatri-Rao product of loading matrices. Several methods have been proposed to reduce this computational complexity by sketching the data required for the calculation of Khatri-Rao product in each iteration. One of the most successful methods in this vein is CPRAND [BBK18], which is a non-constrained CP decomposition and is proper for large dimensions, since it employs sketching data for the calculation of Khatri-Rao product, called sample Khatri-Rao.

Therefore, a future extension of SFBS would be to follow the CPRAND idea (sample Khatri-Rao) in each iteration of SFBS. In this way, we would have a constrained CP decomposition dedicated to tensors of large dimension, which handles properly constraints such as the simplex set constraint.

It is worth mentioning that recently some constrained decomposition algorithms for large dimension tensors are proposed in [FIW$^+$20]. However, we found them improper under the simplex set constraint. Hence algorithms in [FIW$^+$20] could be adapted for this constraint, which is important in estimating probabilities.

The procedure described in Sections 5.5.2.3 and 5.5.2.4 is useful when sparsity constraints are applied on each element of loading factors. However, if sparsity constraints are required for each column of loading factors (as in co-clustering application [PS11]), ℓ_0 and ℓ_1 should be replaced by mixed norms [Kow09] to ensure the sparsity of each column.

Although we focused on single topic model in this thesis, the content can be extended to other data models such as LDA [BNJ03] and Markov models [AGH$^+$14].

A Lipschitz constant of the gradient of the fidelity term in (5.12)

Let us define $h(\boldsymbol{A}^{(n)})$ and $g(\boldsymbol{A}^{(n)})$ as follows:

$$\begin{aligned} h(\boldsymbol{A}^{(n)}) &\triangleq \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2 \\ g(\boldsymbol{A}^{(n)}) &\triangleq i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}), \end{aligned}$$

where $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$. Note that the objective function of (5.12) can be written as $h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)})$.

Since $h(\boldsymbol{A}^{(n)})$ has a quadratic form, the "Lipschitz" constant of its gradient can be calculated. The gradient of $h(\boldsymbol{A}^{(n)})$ is computed as follows:

$$\begin{aligned} h(\boldsymbol{A}^{(n)}) &= \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2 \\ &= \frac{1}{2}\,\mathrm{trace}\{(\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W})^T(\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W})\} \\ &\Rightarrow \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) = -\boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T} + \boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T \end{aligned}$$

In calculating the gradient, the following relations have been used [Duc07]:

$$\nabla_{\boldsymbol{A}}\,\mathrm{trace}\{\boldsymbol{A}\boldsymbol{B}\} = \boldsymbol{B}^T,$$

$$\nabla_{\boldsymbol{A}}\,\mathrm{trace}\{\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{C}\} = \boldsymbol{C}^T\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}.$$

Now, the Lipschitz constant of $\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)})$ can be calculated as follows:

$h(\boldsymbol{A}^{(n)})$ is β-Lipschitz gradient, *i.e.*,

$$\begin{aligned}
&\|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y})\|_F \leq \beta \|\boldsymbol{X} - \boldsymbol{Y}\|_F \\
&\Rightarrow \beta = \max \frac{\|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y})\|_F}{\|\boldsymbol{X} - \boldsymbol{Y}\|_F} \\
&\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y}) = (\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T \\
&\Rightarrow \beta = \max \frac{\|(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T\|_F}{\|\boldsymbol{X} - \boldsymbol{Y}\|_F}.
\end{aligned}$$

Note that the definition of "spectral norm" of a matrix is:

$$\|\boldsymbol{A}\|_\sigma \triangleq \max_{\|\boldsymbol{X}\|_F \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{X}\|_F}{\|\boldsymbol{X}\|_F},$$

which is equal to the maximum singular value of $\boldsymbol{A}$ with $\boldsymbol{X}$ as a matrix. Therefore, we have:

$$\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma.$$

B Résumé en Francais

B.1 Introduction

La plupart des événements, qui sont au centre de l'intérêt de l'étude, sont de nature multimodale. Les entités qui relient et affectent ensemble ces données multimodales sont des aspects cachés (latents), qui sont appelés variables cachées (latentes) [AGH+14]. Les modèles multi-vues ou modèles à variables latentes (MVL) font référence à une vaste gamme de modèles dans lesquels une ou plusieurs variables cachées existent [RCG18].

Tout au long de cette thèse, nous considérons un modèle de mélange particulier, appelé modèle à sujet unique [BNJ03] représenté sur la Fig. B.1, qui peut être utilisé pour décrire une relation cachée parmi les données observées (voir la Section 3.2 pour une définition plus détaillée du modèle à sujet unique). Par exemple, un corpus de documents peut être décrit par un modèle à sujet unique : le sujet de chaque document est considéré comme une variable cachée et les mots de ce document particulier sont ses variables multi-vues générées en fonction de son sujet.

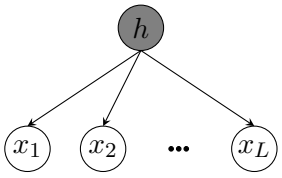

Figure B.1: Le modèle à sujet unique avec ses variables multi-vues (x_i) et leur variable cachée correspondante (h).

Soit L le nombre de mots dans un document donné, $\boldsymbol{x}_\ell$ les mots observés, $\ell \in \mathcal{L} = \{1, 2, \ldots, L\}$, et h un sujet codé par une variable discrète prenant K valeurs entières possibles, disons de $\mathcal{H} = \{1, 2, \ldots, K\}$ avec la probabilité $\varphi(k) = \text{Prob}(h = k)$. Tous les mots appartiennent à un dictionnaire codé connu $\Omega = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D\}$ de cardinalité D. Nous désignons la probabilité conditionnelle de chaque mot $\boldsymbol{u}_d$ du dictionnaire Ω, étant donné un sujet particulier, $h = k$, par $f_k(d) = \text{Prob}(\boldsymbol{x} = \boldsymbol{u}_d | h = k)$.

Comme mentionné précédemment, l'objectif principal est d'estimer la probabilité de la variable cachée (sujet) et la probabilité conditionnelle des variables multi-vues (mots).

À cette fin, certaines hypothèses sont nécessaires et sont énumérées ci-dessous [AGH+14, AHK12a] :

- Les probabilités conditionnelles ne dépendent pas de l'ordre des mots (échangeabilité), *e.g.*

$$\text{Prob}(\boldsymbol{u}_{\gamma_e(p)}, \boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{u}_{\gamma_e(r)} | h) = \text{Prob}(\boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{u}_{\gamma_e(r)}, \boldsymbol{u}_{\gamma_e(p)} | h)$$

- Les mots du sujet sont conditionnellement indépendants compte tenu du sujet, *i.e.*

$$(\boldsymbol{u}_{\gamma_e(p)} | h) \perp\!\!\!\perp (\boldsymbol{u}_{\gamma_e(q)} | h)$$

- Les mots du sujet ont la même distribution conditionnelle étant donné le sujet, *i.e.*

$$(\boldsymbol{u}_{\gamma_e(p)} | h) \sim (\boldsymbol{u}_{\gamma_e(q)} | h).$$

De plus, $\boldsymbol{x}_\ell$ est codé en $\boldsymbol{u}_d$, et comme dans [AGH+14], on choisit pour vecteurs $\boldsymbol{u}_d$ les vecteurs de la base canonique, c'est à dire les colonnes de la matrice identité $D \times D$.

Dans la suite, les moments d'ordre deux et trois seront nécessaires :

$$\boldsymbol{P} \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q\}, \tag{B.1}$$

$$\boldsymbol{\mathcal{T}} \overset{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\{\boldsymbol{x}_p \otimes \boldsymbol{x}_q \otimes \boldsymbol{x}_r\}, \tag{B.2}$$

où $\otimes$ désigne le produit tensoriel, $\boldsymbol{P}$ est une matrice symétrique $D \times D$ et $\boldsymbol{\mathcal{T}}$ un tenseur symétrique $D \times D \times D$.

En raison de ce choix pour $\boldsymbol{u}_d$, ces moments présentent les relations suivantes :

$$\boldsymbol{P} = \sum_{k=1}^{K} \varphi_k \, \boldsymbol{a}_k \otimes \boldsymbol{a}_k, \tag{B.3}$$

$$\boldsymbol{\mathcal{T}} = \sum_{k=1}^{K} \varphi_k \, \boldsymbol{a}_k \otimes \boldsymbol{a}_k \otimes \boldsymbol{a}_k, \tag{B.4}$$

où $\boldsymbol{a}_k$ est la $k$ième colonne d'une matrice notée $\boldsymbol{A}$. Notez que $\boldsymbol{a}_k$ contient les valeurs de $f_k(d)$ pour tout d et chaque k.

Par conséquent, une fois que l'approximation empirique des moments dans (B.1) et (B.2) est obtenue, la probabilité de la variable cachée (sujet) et les probabilités conditionnelles des variables multi-vues (mots), *i.e* $\boldsymbol{\varphi}$ et $\boldsymbol{A}$, peuvent être estimées via la décomposition tensorielle selon (B.4).

Dans le reste de cet appendice, nous étudions les estimations de moments dans la section B.2. Les deux sections suivantes sont consacrées aux décompositions tensorielles et aux indices de performance requis pour évaluer les algorithmes de décomposition.

B.2 Estimation des moments

B.2.1 L'état de l'art

B.2.1.1 Moyenne simple [AFH+12]

Cet estimateur est simplement la moyenne d'un grand nombre de réalisations (un *corpus* de documents), disons N_c, pour obtenir une approximation acceptable de $\boldsymbol{P}$ et $\boldsymbol{\mathcal{T}}$. Dans chaque réalisation, un sujet codé aléatoire, k, est tiré de la manière décrite dans la première étape des processus génératifs dans les sections 3.4.1.1 et 3.4.1.2. Ensuite, en fonction du thème choisi, *trois* mots codés aléatoires (on suppose qu'un document est constitué d'au moins trois mots) sont tirés, $\{\boldsymbol{x}_p = \boldsymbol{u}_{\gamma_e(p)}, \boldsymbol{x}_q = \boldsymbol{u}_{\gamma_e(q)}, \boldsymbol{x}_r = \boldsymbol{u}_{\gamma_e(r)}\}$, en se basant sur la deuxième étape des processus génératifs mentionnés dans les sections 3.4.1.1 et 3.4.1.2. A la fin, en utilisant les moyennes suivantes, une

approximation *empirique* des moments d'ordre deux et trois est obtenue avec les exemples de moments ci-dessous :

$$\boldsymbol{P}_e = \frac{1}{N_c} \sum_{n=1}^{N_c} \boldsymbol{u}_{\gamma_e(p)} \otimes \boldsymbol{u}_{\gamma_e(q)}, \tag{B.5}$$

$$\boldsymbol{\mathcal{T}}_e = \frac{1}{N_c} \sum_{n=1}^{N_c} \boldsymbol{u}_{\gamma_e(p)} \otimes \boldsymbol{u}_{\gamma_e(q)} \otimes \boldsymbol{u}_{\gamma_e(r)}. \tag{B.6}$$

Notez que dans cette méthode d'estimation, la longueur de chaque document (L_n) n'a pas d'importance, et que seuls trois mots (pouvant être le premier, celui du milieu et le dernier mot) de chaque document participent à l'estimation du moment de l'échantillon. En raison de ces faits, il est très difficile de fournir une cohérence de moment raisonnable avec une simple moyenne et cela nécessite un corpus de grande taille, *i.e* une grande valeur de N_c dans (B.5) et (B.6).

B.2.1.2 L'estimateur de Ruffini [RCG18]

Supposons qu'un document soit constitué de L_n mots ($\boldsymbol{x}_\ell, \quad \ell = 1, \ldots, L_n$) qui sont codés en $\boldsymbol{u}_d = \boldsymbol{u}_{\gamma_e(\ell)}$. Alors, le sac de mots d'un tel document est un vecteur de dimension D, et est défini comme suit:

$$\boldsymbol{b}_n \overset{\mathsf{def}}{=} \sum_{\ell=1}^{L_n} \boldsymbol{u}_{\gamma_e(\ell)}, \tag{B.7}$$

où $b_n(d)$ indique combien de fois le mot d^{th} apparaît dans le document n.

Les estimateurs de Ruffini, *i.e.* $\boldsymbol{P}_{\text{Ruffini}}$ et $\boldsymbol{\mathcal{T}}_{\text{Ruffini}}$, sont définis comme suit ($i < j$) :

$$P_{\text{Ruffini}}(i,j) = \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j) - \delta_{i=j})}{\sum_{n=1}^{N_c} L_n(L_n - 1)} \tag{B.8}$$

$$\begin{aligned} T_{\text{Ruffini}}(i,j,k) = {} & \delta_{i<j<k} \frac{\sum_{n=1}^{N_c} b_n(i) b_n(j) b_n(k)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)} \\ & + \delta_{(i=j<k)\vee(i<j=k)} \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j)-1) b_n(k)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)} \\ & + \delta_{i=j=k} \frac{\sum_{n=1}^{N_c} b_n(i)(b_n(j)-1)(b_n(k)-2)}{\sum_{n=1}^{N_c} L_n(L_n-1)(L_n-2)}. \end{aligned} \tag{B.9}$$

En employant le sac de mots de chaque document du corpus, les nominateurs des expressions ci-dessus correspondent à la mise en œuvre des moments (sommation et production de variables aléatoires), et leurs dénominateurs sont les coefficients, qui rendent cet estimateur sans biais.

Notez qu'en calculant la partie triangulaire supérieure de $\boldsymbol{P}_{\text{Ruffini}}$ et de $\boldsymbol{\mathcal{T}}_{\text{Ruffini}}$, puis, en fixant la partie triangulaire inférieure identique à la partie supérieure, la symétrisation serait parfaite.

B.2.2 Proposé : Moyenne standard

En prenant en compte la longueur et tous les mots (et pas seulement trois) de chaque document, nous proposons un algorithme, que nous appelons *standard averaging*, et qui peut être vu comme une moyenne pondérée basée sur la longueur de chaque document.

L'algorithme 6 décrit en détail le calcul de la moyenne standard. Notez que la symétrisation de la ligne 12 est exécutée en calculant $\frac{\boldsymbol{P}+\boldsymbol{P}^T}{2}$. Nous avons observé expérimentalement que cette méthode de symétrisation montre une meilleure performance comparée à celle de l'estimateur de Ruffini dans lequel les valeurs de la partie triangulaire inférieure seront fixées identiques à la partie supérieure à la fin de l'algorithme. L'algorithme pour les moments d'ordre 3 est similaire, sauf qu'à la ligne 6, tous les *triplets* possibles seront considérés. Notez que la symétrisation d'un tenseur d'ordre 3 (cf. ligne 12) nécessite de faire la moyenne sur six permutations d'indices.

Les simulations de la Section B.2.3 montrent que la moyenne standard est plus performante que l'estimateur de Ruffini, lorsque la dimension du dictionnaire, D, n'est pas très grande. Plus de détails sur les performances sont donnés dans la section 3.4.5. De plus, la moyenne standard présente certains avantages en termes de complexité de calcul :

- la moyenne standard nécessite moins de multiplications dans l'estimation de $\boldsymbol{\mathcal{T}}$,
- le nombre d'additions requises dans l'estimation de $\boldsymbol{P}$ ne dépend pas

Algorithm 6 Moyenne standard pour l'estimation des moments du second ordre

Entrée: D, un corpus de N_c documents de longueur $[L_1, \ldots, L_{N_c}]$

Sortie: $\boldsymbol{P}_{\text{Stand-Ave}}$

$\boldsymbol{P}_{\text{Stand-Ave}} = \text{zeros}(D,D)$
for $n = 1, 2, \ldots, N_c$ **do**
  Sauvegarder l'index des mots du document "n".
  $\boldsymbol{P}_{\text{temp}} = \text{zeros}(D,D)$
  count-pair $= 0$
  for toutes les paires de mots possibles, ex. $(\text{word}_1, \text{word}_2)$ **do**
    $\boldsymbol{P}_{\text{temp}}(\text{word}_1, \text{word}_2) = 1 + \boldsymbol{P}_{\text{temp}}(\text{word}_1, \text{word}_2)$
    count-pair $= 1 +$ count-pair
  end for
  $\boldsymbol{P}_{\text{Stand-Ave}} = \boldsymbol{P}_{\text{Stand-Ave}} + \frac{L_n}{\sum_{n=1}^{N_c} L_n} \frac{\boldsymbol{P}_{\text{temp}}}{\text{count-pair}}$
end for
Symétriser $\boldsymbol{P}_{\text{Stand-Ave}}$

de la taille du dictionnaire (D), ce qui est un avantage dans le cas de dictionnaires de grande dimension.

B.2.3 Résultats de la simulation

Toutes les expériences informatiques rapportées dans cette section ont été exécutées soit sur un ordinateur portable avec un processeur Intel Core i5 de 3,1 GHz, 16 Go de RAM, soit sur un PC avec un processeur Intel Core i5 de 3,2 GHz, 8 Go de RAM, tous deux exécutant macOS Mojav et MATLAB 2019a.

B.2.3.1 L'effet de la taille du corpus sur la cohérence du moment

Comme mentionné précédemment, le simple calcul de la moyenne ne permet pas d'obtenir une cohérence des moments acceptable et une façon naïve de compenser cet inconvénient est d'augmenter la taille du corpus (le nombre de

documents ou de réalisations du sujet/de la variable cachée et des mots/des variables multi-vues).

Afin de montrer ce défi en pratique, nous avons généré N_c réalisations de notre ensemble de données synthétiques, $\boldsymbol{x}_\ell = \boldsymbol{u}_{\gamma_e(\ell)}$, (selon la méthode expliquée dans la section 3.4.1.2) avec les valeurs de paramètres suivantes : le nombre de variables cachées (sujets), $K = 4$; le nombre de variables multi-vues (mots), $D = 8$; le nombre de réalisations pour l'estimation des moments (la taille du corpus, *i. e.* le nombre de documents), $N_c = [2^9, 2^{10}, \ldots, 2^{17}]$. Les longueurs minimale et maximale des documents générés sont respectivement de 3 et 100.

Enfin, nous avons calculé les moments de l'échantillon avec les estimateurs de moments décrits dans la section B.2.1 et B.2.2, puis nous les avons comparés aux moments réels (B.3) et (B.4). La Fig. B.2 rapporte l'écart entre les deux en termes de norme euclidienne. Comme on peut le voir sur la Fig. B.2, l'estimation par moyenne simple, de la manière décrite dans la section B.2.1.1, est cohérente, si elle fonctionne sur un corpus de taille $N_c > 10000$. L'estimateur de Ruffini et la moyenne standard ont la même erreur relative (ils sont superposés dans la Fig. B.2), cependant, nous verrons la différence entre les performances de l'estimateur de Ruffini et de la moyenne standard dans la Section B.2.3.2 en étudiant l'effet de la cardinalité du dictionnaire.

B.2.3.2 Effet de la cardinalité du dictionnaire sur la cohérence des moments

Dans cette section, nous étudions l'effet de la cardinalité du dictionnaire, D, sur les performances des estimateurs de moment. Pour toutes les expériences, nous considérons un corpus de $N_c = 100$ documents dont la longueur L_n est choisie aléatoirement entre une longueur minimale ($L_{\min} = 3$) et une longueur maximale ($L_{\max} = 100$). Les sujets et les mots de chaque document sont générés selon le processus génératif décrit dans la section 3.4.1.2.

Les figures B.3 et B.4 montrent l'erreur relative dans l'estimation de $\boldsymbol{P}$

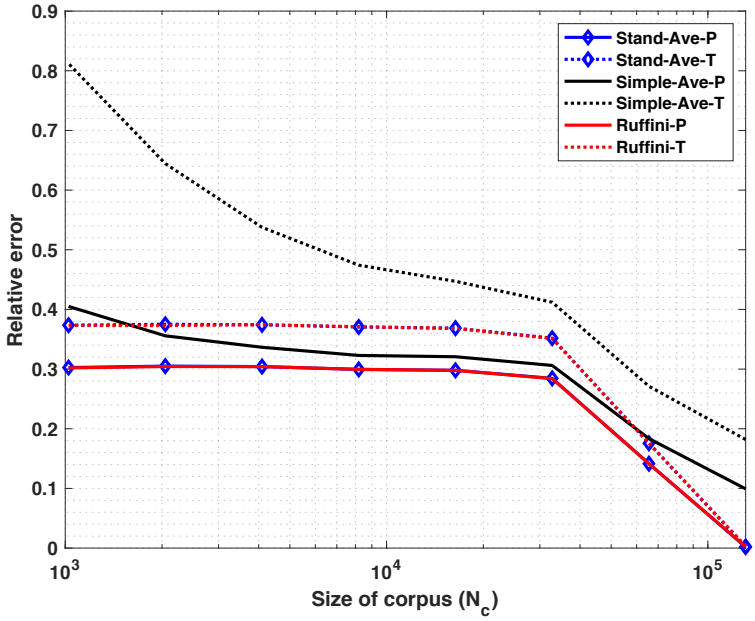

Figure B.2: La cohérence des estimations des moments. L'erreur relative dans l'estimation des moments d'ordre deux ($\boldsymbol{P}$) et d'ordre trois ($\boldsymbol{\mathcal{T}}$) est tracée en ligne continue et en pointillé, respectivement. Notez que les tracés du calcul de la moyenne standard et de l'estimateur de Ruffini sont superposés. Dans cette figure, le nombre de corpus utilisés est fonction de leur taille: plus la taille du corpus est grande, moins il y a de corpus; l'intérêt de ce choix délibéré est de maintenir constante la longueur totale du bloc de données (et donc la charge de calcul) pour chaque point en abscisse.

et $\boldsymbol{\mathcal{T}}$ par rapport à une gamme de valeurs de D, lorsque K est fixé à 10. Puisque dans la Fig. B.3, la différence entre le calcul de la moyenne standard et l'estimateur de Ruffini n'est pas très évidente, nous traçons les résultats du calcul de la moyenne standard et de l'estimateur de Ruffini séparément dans la Fig. B.4.

Comme on peut le voir sur les figures B.3 et B.4, non seulement la moyenne standard est un estimateur bien meilleur que la moyenne simple, mais elle est également légèrement plus performante que l'estimateur de Ruffini lorsque la cardinalité du dictionnaire, D, est inférieure à 50.

B.3 Indices de performance

Les ambiguïtés de permutation et d'échelle sont des problèmes pertinents dans des applications telles que la décomposition tensorielle [Com14] et la séparation aveugle de sources (BSS) [CJ10]. La mise à l'échelle fait référence à la multiplication par une matrice diagonale à composantes non nulles, qui peut être complexe dans le cas le plus général, et la permutation fait référence à la permutation des colonnes d'une matrice, ce qui est équivalent à la multiplication par une matrice de permutation.

Soit $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N] \in \mathbb{C}^{M\times N}$ et $\widehat{\boldsymbol{A}} = [\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2, \ldots, \hat{\boldsymbol{a}}_N] \in \mathbb{C}^{M\times N}$ sont respectivement les matrices originales et estimées, où $\mathbb{C}^{M\times N}$ représente l'ensemble des matrices à valeurs complexes M par N. Désignons l'ensemble des permutations de N éléments par $\mathrm{Perm}(N)$, et désignons par $\boldsymbol{P}_\sigma$ la matrice associée à la permutation $\sigma \in \mathrm{Perm}(N)$. Si les colonnes de $\boldsymbol{A}$ et de $\widehat{\boldsymbol{A}}$ sont normalisées par leurs normes L_2, l'ambiguïté d'échelle se réduit à la post-multiplication par une matrice diagonale $\boldsymbol{\Lambda}$ avec des composantes de module unité (par exemple dans $\mathbb{R}$, les valeurs sont égales a ± 1).

Supposons que $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda} + \boldsymbol{W}$, où les colonnes de $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$ sont normalisées, $\boldsymbol{\Lambda}$ est une matrice diagonale avec des composantes de module unité et $\boldsymbol{W}$ est l'erreur d'estimation de $\widehat{\boldsymbol{A}}$, qui peut être modélisée comme un bruit additif. Plus formellement, l'objectif est de mesurer l'écart défini ci-dessous:

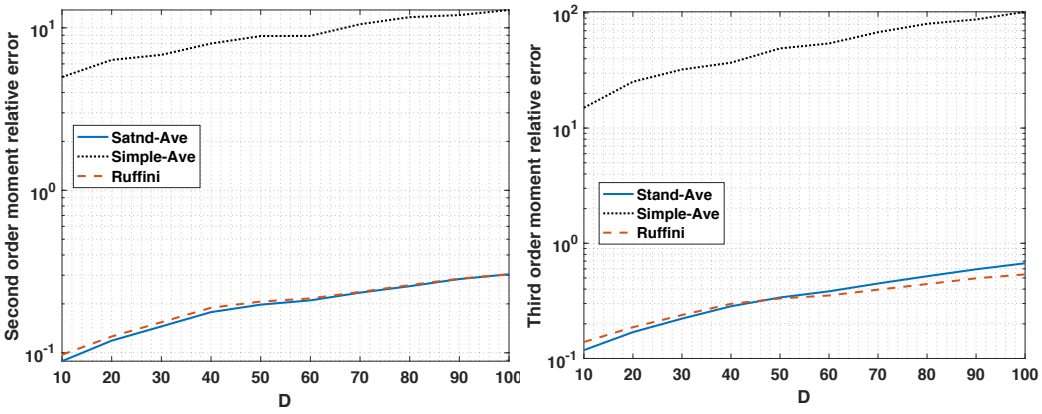

Figure B.3: Comparaison entre tous les estimateurs mentionnés pour l'estimation de $\boldsymbol{P}$ et de $\boldsymbol{\mathcal{T}}$ avec $K = 10$, $D = [10, 20, 30, \ldots, 100]$, $N = 100$, $L_{\min} = 3$, $L_{\max} = 100$, et calcul de la moyenne sur 50 de probabilités différentes $\boldsymbol{\varphi}$, $\boldsymbol{A}$, Gauche : L'erreur relative de l'estimation des moments du second ordre, Droite : L'erreur relative de l'estimation des moments du troisième ordre.

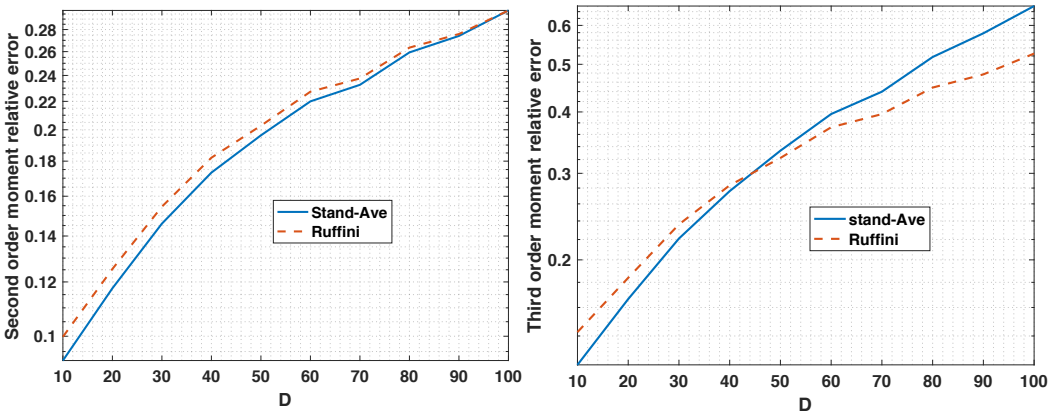

Figure B.4: Zoom de la Fig. B.3 pour montrer les détails.

$$\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \min_{\sigma, \boldsymbol{\Lambda}} \|\boldsymbol{A}\boldsymbol{P}_\sigma\boldsymbol{\Lambda} - \widehat{\boldsymbol{A}}\|_F^2 \tag{B.10}$$

Cet écart peut être calculé avec ou sans estimation explicite de la permutation σ. La recherche de la permutation optimale σ peut être écrite comme le problème d'optimisation suivant :

$$\underset{\sigma}{\operatorname{argmin}} \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{a}_n - \hat{\boldsymbol{a}}_{\sigma(n)}\|_2^2 = \underset{\sigma}{\operatorname{argmax}} \sum_{n=1}^{N} |\boldsymbol{a}_n^H \hat{\boldsymbol{a}}_{\sigma(n)}|. \tag{B.11}$$

B.3.1 L'état de l'art

B.3.1.1 Méthodes basées sur la matrice de corrélation

Soit $C_{ij} = |\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j|$, et désignons par $\boldsymbol{C}$ la matrice de corrélation dont les composantes sont C_{ij}. Si les colonnes de $\boldsymbol{A}$ et de $\widehat{\boldsymbol{A}}$ sont normalisées par leurs normes L_2, nous avons $0 \leq C_{ij} \leq 1$. Dans la suite, nous passons en revue de manière critique deux méthodes gloutonnes basées sur la matrice de corrélation, et nous montrons, à l'aide d'un exemple simple, pourquoi ces méthodes sont gloutonnes et peu fiables.

Approche gloutonne de [FIW+20] Dans cette approche, $\hat{\boldsymbol{a}}_j$ est attribué à $\boldsymbol{a}_i$ si C_{ij} a la valeur maximale dans la j^{ime} colonne de $\boldsymbol{C}$. L'exemple numérique jouet suivant illustre les effets pervers du caractère glouton de cette approche simple.

Supposons que, dans une expérience, la matrice $\boldsymbol{C}$ est:

$$\boldsymbol{C} = \begin{bmatrix} 0.8 & 0.3 & 0.1 \\ 0.85 & 0.9 & 0.5 \\ 0.5 & 0.2 & 0.7 \end{bmatrix}. \tag{B.12}$$

L'affectation proposée par cette méthode est $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$, ce qui n'est évidemment pas acceptable car la colonne $\boldsymbol{a}_2$ est sélectionnée deux fois. Si on calcule l'erreur quadratique via $\frac{1}{2}\sum_{n=1}^{3} \|\boldsymbol{a}_n - \hat{\boldsymbol{a}}_{\sigma(n)}\|_2^2$ en considérant l'hypothèse de normalisation de $\boldsymbol{a}_n$ et $\hat{\boldsymbol{a}}_{\sigma(n)}$ par rapport à la norme L_2

et en substituant les valeurs de $|\boldsymbol{a}_i^H \hat{\boldsymbol{a}}_j|$ de C_{ij}, on obtient $3-0.85-0.9-0.7 = 0.55$, ce qui est inférieur à l'erreur exacte, $3-0.8-0.9-0.7 = 0.60$ (l'erreur exacte est donnée dans la section 4.3.3 avec la permutation optimale). Cet exemple montre que cet algorithme produit une matrice $\boldsymbol{P}_p$ qui peut ne pas être une permutation. Cet indice est toujours *optimiste* (cf. Section 4.3.1.1).

Approche gloutonne de [CLDA09, CKAC14] Afin d'éviter une affectation non acceptable, après avoir détecté la valeur maximale de chaque colonne de $\boldsymbol{C}$, sa ligne et sa colonne peuvent être supprimées pour le reste de l'algorithme. Il s'agit d'une approche *gloutonne*, puisque l'indice dépend de l'ordre de choix des valeurs maximales. Par exemple, si cet algorithme glouton est appliqué à la matrice $\boldsymbol{C}$ exprimée en (B.12), l'affectation résultante sera $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_1)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$ à condition que les colonnes soient balayées de gauche à droite. En revanche, si les colonnes sont balayées dans le sens inverse, l'affectation sera $(\hat{\boldsymbol{a}}_1, \boldsymbol{a}_1)$, $(\hat{\boldsymbol{a}}_2, \boldsymbol{a}_2)$, $(\hat{\boldsymbol{a}}_3, \boldsymbol{a}_3)$. Par rapport à l'indice optimiste, l'erreur produite par cette approche gloutonne en balayant de gauche à droite, $3-0.85-0.3-0.7 = 1.15$, est plus grande que l'erreur exacte, $3-0.8-0.9-0.7 = 0.60$, tandis qu'en balayant de droite à gauche, l'erreur rapportée est égale à l'erreur exacte 0.6. Par conséquent, l'erreur signalée est toujours supérieure ou égale à l'erreur exacte sur la base de l'affectation optimale; cet indice est donc pessimiste.

B.3.1.2 Méthodes basées sur le transport optimal

Problème d'affectation optimale Le problème d'assignation est un problème d'optimisation combinatoire ancien, bien connu et fondamental [Gal83, Mun57, PC$^+$19]. Le premier algorithme en temps polynomial pour les problèmes d'affectation optimale est la "méthode hongroise" [Kuh55] également connue sous le nom de "Kuhn-Munkres" [Mun57, TK04], et la complexité de l'algorithme est approximativement $O(N^4)$ flops [Mun57]. Cet algorithme a été utilisé dans [TK04] pour un appariement optimal des sources dans BSS.

Matching pondéré maximum (MWM[1]) Le problème de l'affectation optimale peut également être considéré comme un cas particulier de l'matching pondéré maximum (MWM), qui est un problème bien connu en théorie des graphes, pour lequel plusieurs algorithmes en temps polynomial existent [Gal83]. Les meilleurs algorithmes de MWM exacts [DP14, DK69, D$^+$59] et approximatifs [GT89] coûtent environ $8N^3$ et N^2 flops, respectivement.

Programmation linéaire La recherche de la permutation optimale σ, *i.e* pour la matrice de permutation optimale $\boldsymbol{P}^\star$, peut être formulée ceci comme le problème d'optimisation suivant [PC$^+$19] :

$$\boldsymbol{P}^\star = \underset{\boldsymbol{P}_p \in \mathbb{R}_+^{N\times N}}{\operatorname{argmin}} \sum_{i,j} D_{ij} P_{p_{ij}} \quad \text{s.t.} \quad \boldsymbol{P}_p \mathbb{1}_N = \boldsymbol{P}_p^T \mathbb{1}_N = \mathbb{1}_N, \tag{B.13}$$

où $\boldsymbol{D} = -\boldsymbol{C}$, $\mathbb{1}_N$ est un vecteur de 1 de dimension N et l'exposant $\star$ désigne la solution optimale.

En vectorisant (concaténation des colonnes) $\boldsymbol{P}_p$ et $\boldsymbol{D}$ en vecteurs $\boldsymbol{d}$ et $\boldsymbol{p}$, (B.13) peut être réécrit sous la forme standard de programmation linéaire [PC$^+$19, Sec. 3.1]:

$$\boldsymbol{p}^\star = \underset{\boldsymbol{p}\in \mathbb{R}_+^{N^2}}{\operatorname{argmin}} \boldsymbol{d}^H \boldsymbol{p} \quad \text{s.t.} \quad \boldsymbol{Q}\boldsymbol{p} = \mathbb{1}_{2N}, \tag{B.14}$$

où $\boldsymbol{Q} = [\mathbb{1}_N^T \boxtimes \boldsymbol{I}_N, \boldsymbol{I}_N \boxtimes \mathbb{1}_N^T]^T \in \mathbb{R}^{2N\times N^2}$, $\boldsymbol{I}_N$ et $\boxtimes$ désignent respectivement la matrice identité de taille N et le produit de Kronecker.

Récemment, en améliorant certaines étapes de calcul requises, le temps d'exécution des algorithmes de programmation linéaire a été réduit à environ $q^{2+\frac{1}{6}}$ [CLS21] et $q^{2+\frac{1}{18}}$ [LSZ19] flops, où q est la taille du vecteur inconnu dans le problème de programmation linéaire. Par conséquent, comme $q = N^2$ dans (B.14), la complexité la plus faible pour trouver la permutation optimale du problème décrit au début de la section 4.3 au moyen de la programmation linéaire est approximativement de N^4 flops.

[1]Maximum Weighted Matching

B.3.1.3 Indices invariants à la permutation

Cependant, dans la littérature sur la séparation des sources [CJ10], certains indices ont été proposés pour mesurer l'écart (basé sur une définition spécifique de l'écart) entre les matrices de mélange originales et estimées sans chercher à trouver la permutation correspondante [Com94, MM94, ACY$^+$96]. Les indices proposés dans [Com94, MM94, ACY$^+$96] sont basés sur $\boldsymbol{S} = \boldsymbol{A}^{-1}\widehat{\boldsymbol{A}}$ (ou $\boldsymbol{S} = \boldsymbol{A}^{\dagger}\widehat{\boldsymbol{A}}$ pour les matrices $\boldsymbol{A}$ non carrées). Les détails de ces indices sont les suivants.

Indice de Comon [Com94] L'indice de Comon est une combinaison des normes L_1 et L_2, et se calcule comme suit :

$$\epsilon_1(\boldsymbol{S}) = \sum_{i=1}^{N}\left|\sum_{j=1}^{N}|S_{ij}| - 1\right|^2 + \sum_{j=1}^{N}\left|\sum_{i=1}^{N}|S_{ij}| - 1\right|^2, \\ \sum_{i=1}^{N}\left|\sum_{j=1}^{N}|S_{ij}|^2 - 1\right| + \sum_{j=1}^{N}\left|\sum_{i=1}^{N}|S_{ij}|^2 - 1\right|.$$

La valeur de ϵ_1 peut augmenter énormément, en fonction des valeurs de la matrice $\boldsymbol{S}$, par conséquent, cet indice n'est pas borné supérieurement.

Indice de Moreau-Macchi [MM94] L'indice proposé dans [MM94] mesure un écart entre la matrice $\boldsymbol{S}$ et une matrice de permutation. Il est défini comme suit :

$$\epsilon_2(\boldsymbol{S}) = \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|S_{ij}|^2}{(\max_k|S_{ik}|)^2} - 1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|S_{ij}|^2}{(\max_k|S_{kj}|)^2} - 1\right).$$

La division par la valeur maximale (*e.g.* $(\max_k|S_{ik}|)^2$) fournit une borne supérieure pour ϵ_2 contrairement à ϵ_1.

Indice de Amari [ACY$^+$96] Cet indice de performance prend la forme :

$$\epsilon_3(\boldsymbol{S}) = \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|S_{ij}|}{\max_k|S_{ik}|} - 1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|S_{ij}|}{\max_k|S_{kj}|} - 1\right).$$

La seule différence entre l'indice d'Amari et celui de Moreau-Macchi est la puissance 2, qui existe dans ϵ_2. Par conséquent, le calcul de ϵ_3 est moins coûteux que celui de ϵ_2.

Une étude précise des indices examinés dans cette section révèle que ϵ_1 n'est pas borné supérieurement. De plus, les bornes supérieures de ϵ_2 et ϵ_3 n'ont pas été étudiées dans [MM94, ACY+96], de sorte que leur borne supérieure ne peut être facilement interprétée. Même si l'on normalise les indices d'Amari et de Moreau-Macchi, les limites supérieures obtenues sont atteintes lorsque $\boldsymbol{A}$ a des colonnes équivalentes et que $\widehat{\boldsymbol{A}}$ est la matrice identité. Par conséquent, les limites supérieures des indices d'Amari et de Moreau-Macchi ne correspondent pas au plus grand écart angulaire possible entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$.

B.3.2 Notre proposition d'index : CorrIndex

Rappelons que nous définissons $\boldsymbol{C} = |\boldsymbol{A}^H \widehat{\boldsymbol{A}}|$, où $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ et $\widehat{\boldsymbol{A}} \in \mathbb{C}^{M \times N}$ et le module est appliqué pour chaque élément. En outre, nous supposons que les colonnes de $\boldsymbol{A}$ et de $\widehat{\boldsymbol{A}}$ sont normalisées par leurs normes L_2.

Fondamentalement, si $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$, N composantes de $\boldsymbol{C}$ sont égales à 1, puisque $|\boldsymbol{a}_n| = |\hat{\boldsymbol{a}}_n|$ et les colonnes de $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$ sont normalisées par leurs normes L_2. Rappelons que l'on souhaite qu'un indice de performance soit nul si et seulement si $\epsilon_0(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = 0$. Afin de satisfaire ces exigences de base dans le cas de la matrice, *i.e.* $M > 1$, CorrIndex est défini comme suit :

$$\text{CorrIndex}(\boldsymbol{C}) = \frac{1}{2N}\left[\sum_{i=1}^{N} |\max_k C_{ik} - 1| + \sum_{j=1}^{N} |\max_k C_{kj} - 1|\right]. \quad \text{(B.15)}$$

Le coefficient $\dfrac{1}{2N}$ maintient les valeurs de CorrIndex dans l'intervalle $[0, 1]$. De plus, le module $|.|$ garantit une distance nulle entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$ si CorrIndex = 0 (cf. Proposition 2). De plus, selon (B.15), si la distance entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$ est nulle, alors CorrIndex = 0. Par conséquent, les deux exigences mentionnées ci-dessus sont simplement satisfaites par (B.15). De plus, CorrIndex est invariant à la permutation et au changement d'échelle

(cf. Proposition 1).

B.3.3 Résultats de la simulation

L'indice et le temps de calcul de chaque indice dans une expérience numérique sont rapportés dans le Tableau B.1 pour évaluer les méthodes de manière pratique. Cette expérience est exécutée sur un ordinateur portable avec un processeur Intel Core i5 de 3,1 GHz, 16 Go de RAM, exécutant macOS Mojave et MATLAB 2019a.

Afin de montrer les inconvénients des méthodes gourmandes, cette expérience est réalisée sur certaines matrices, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, dont les colonnes sont fortement corrélées. Dans l'expérience du Tableau B.1, $M = 150, N = 100$, avec une cohérence mutuelle $\gamma = 0.75$, $\delta = 0.1$ (ce qui est équivalent à SNR $= -1.76$ dB), et $\boldsymbol{U}$ est une matrice orthogonale obtenue en concaténant les N premiers vecteurs singuliers de gauche d'une matrice aléatoire dont les composantes sont choisies aléatoirement à partir d'une distribution uniforme sur $(0, 1)$. Les valeurs indiquées sont des moyennes sur 50 de réalisations.

Comme on peut le voir dans le Tableau B.1, CorrIndex est l'indice le plus rapide. De plus, contrairement aux indices de Comon, Moreau-Macchi et Amari, CorrIndex retourne une valeur dans l'intervalle borné $[0, 1]$. On pourrait normaliser les indices de Moreau-Macchi et d'Amari afin d'obtenir des valeurs bornées, mais leur borne supérieure ne peut être facilement interprétée car elle ne correspond pas au plus grand écart angulaire possible entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$.

CorrIndex est basé sur (B.11), qui tente de minimiser l'erreur des moindres carrés entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$. Par conséquent, si la distance entre $\boldsymbol{A}$ et $\widehat{\boldsymbol{A}}$ augmente en raison du bruit additif dans $\widehat{\boldsymbol{A}}$, CorrIndex retournera une valeur plus grande. Pour montrer ce fait en pratique, nous avons réalisé une expérience dont le résultat est représenté sur la Fig. B.5. En générant une matrice aléatoire $\boldsymbol{A}$ de dimensions 6×4, on obtient $\widehat{\boldsymbol{A}}$ en permutant ses colonnes et en ajoutant une matrice de bruit, $\boldsymbol{W}$, de même taille que $\boldsymbol{A}$ avec

Table B.1: Comparaison numérique des méthodes de mesure de la distance entre $\boldsymbol{A}^{150\times100}$ avec une cohérence mutuelle $\gamma = 0,75$ et sa version bruitée permutée $\widehat{\boldsymbol{A}}$ avec SNR $= -1,76$ dB en moyenne sur 50 de réalisations. L'indice dans les cinq premières lignes du tableau est l'erreur relative. En revanche, les quatre derniers indices du tableau sont définis différemment et ne sont donc pas comparables.

Méthode	Index	Temps de calcul (ms)	Significativité de la limite supérieure
Greedy of [FIW+20]	0.37	0.9	Non
Greedy of [CLDA09, CKAC14]	1.05	5.3	Non
Hungarian [Mun57]	0.86	4.5	Oui
MWM [DP14]	0.86	2.9	Oui
Linprog [PC+19]	0.86	1310	Oui
Comon [Com94]	1.8e4	3.3	Non
Moreau-Macchi [MM94]	897.91	3.3	Non
Amari [ACY+96]	3.2e3	2.9	Non
CorrIndex	0.36	0.4	Oui

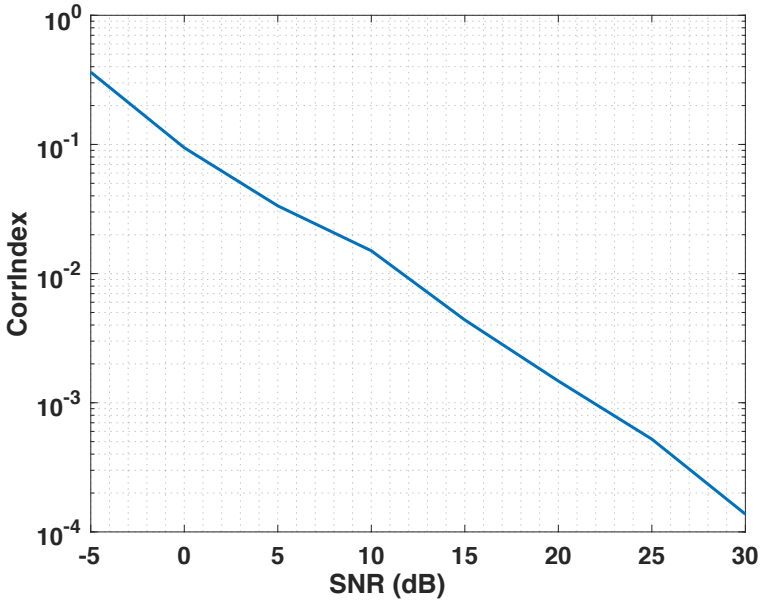

Figure B.5: CorrIndex et bruit. CorrIndex d'une matrice aléatoire $\boldsymbol{A}^{6\times4}$ et de sa version bruitée permutée $\widehat{\boldsymbol{A}}$. Cette figure confirme le fait que plus le bruit est grand, plus le CorrIndex est grand.

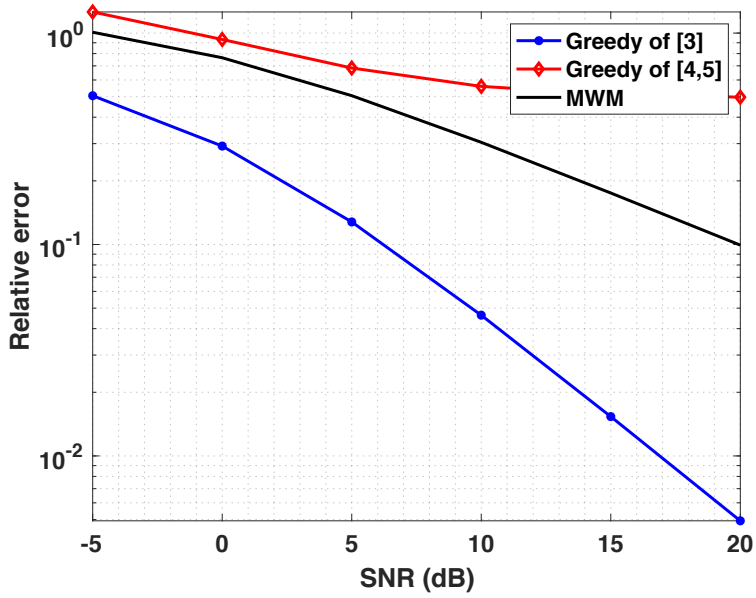

Figure B.6: Inconvénients des méthodes gloutonnes de [FIW+20, CLDA09, CKAC14]. Comparez l'erreur relative entre une matrice aléatoire $\boldsymbol{A}^{150\times100}$ avec la cohérence mutuelle $\gamma = 0.75$ et sa version permutée bruitée $\widehat{\boldsymbol{A}}$ en fonction du rapport signal/bruit rapporté par les méthodes avgloutonnes de [FIW+20,CLDA09, CKAC14] et par l'un des indices exacts, *i.e* MWM moyenné sur 50 réalisations.

des composantes indépendantes et identiquement distribuées (i.i.d.) de distribution gaussienne de moyenne nulle et de variance unitaire, et pondérées par le paramètre δ. La variance δ^2 du bruit additif est ajustée de manière à atteindre le SNR souhaité.

La figure B.5 confirme le fait que plus le RSB est faible, plus le CorrIndex est grand. Par conséquent, lors de l'évaluation de différentes méthodes de décomposition, celle qui présente le CorrIndex le plus faible sera la plus performante.

La figure B.6 montre l'erreur relative entre $\widehat{\boldsymbol{A}}$ et $\boldsymbol{A}$. Comme l'erreur produite par MWM est égale à l'erreur théorique, la différence entre les erreurs produites par les méthodes gloutonnes de [FIW+20,CLDA09,CKAC14] et celle de MWM montre l'inexactitude de [FIW+20, CLDA09, CKAC14]. Comme on peut s'y attendre, l'erreur relative de la méthode gloutonne

[FIW+20] (resp. [CLDA09,CKAC14]) est optimiste (resp. pessimiste), puisque son erreur rapportée est plus petite (resp. grande) que l'erreur exacte. De plus, lorsque le SNR augmente, cette erreur devient plus grande, ce qui démontre qu'en diminuant le bruit additif, l'influence de la cohérence mutuelle devient plus visible sur le résultat des méthodes gloutonnes.

B.4 Décomposition tensorielle

B.4.1 Décomposition tensorielle Canonique Polyadique (CP)

Un tenseur décomposable d'ordre N est un produit tensoriel de N vecteurs [Com14], *i.e.*, $\boldsymbol{\mathcal{D}} = \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(2)} \ldots \otimes \boldsymbol{a}^{(N)}$. Selon [Hit27], tout tenseur peut être écrit comme une combinaison linéaire d'un nombre fini de tenseurs décomposables,

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \lambda_r \boldsymbol{\mathcal{D}}(r), \tag{B.16}$$

où $\boldsymbol{\mathcal{T}}$ est un tenseur d'ordre N, et $\boldsymbol{\mathcal{D}}(r) = \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{a}_r^{(2)} \ldots \otimes \boldsymbol{a}_r^{(N)}$. La décomposition décrite dans (B.16) s'appelle la *décomposition polyadique* [Hit27], et si elle est unique, elle est appelée la *décomposition Canonique Polyadique (CP)* ou aussi *CANDECOMP* ou *PARAFAC* [Com14]. Sous forme compacte, (B.16) peut être représentée par $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, dans lequel $\boldsymbol{\lambda}$ est un *vecteur de coefficients* de taille R contenant les valeurs λ_r, et $\boldsymbol{a}_r^{(n)}$ désigne la $r$ième colonne de la matrice $\boldsymbol{A}^{(n)}, 1 \leq n \leq N$. La matrice $\boldsymbol{A}^{(n)}$ est la *matrice facteur* du mode n [CMDL+15].

Le problème de la décomposition CP d'ordre N sous contrainte peut être exprimé comme la minimisation suivante :

$$\min_{\boldsymbol{\lambda}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2$$
$$\text{s.t. } \boldsymbol{\mathcal{C}}_{\boldsymbol{\lambda}}, \boldsymbol{\mathcal{C}}_{\boldsymbol{A}^{(n)}},\ 1 \leq n \leq N$$

où $\boldsymbol{\mathcal{T}}$ est un tenseur d'ordre N de dimensions $n_1 \times n_2 \ldots \times n_N$ et de rang R, le vecteur de coefficient $\boldsymbol{\lambda}$ est de dimensions $R \times 1$, et les N matrices

$\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$ sont de dimensions $n_1 \times R, n_2 \times R, \ldots, n_N \times R$, respectivement. En outre, $\mathcal{C}_{\boldsymbol{\lambda}}, \mathcal{C}_{\boldsymbol{A}^{(n)}}$ représentent la contrainte sur le vecteur de coefficient $\boldsymbol{\lambda}$ et la matrice $\boldsymbol{A}^{(n)}$, respectivement.

Puisque la minimisation de la fonction de coût de la décomposition CP sur toutes les matrices-facteurs est un problème non-convexe, une stratégie commune pour le transformer en une séquence de problèmes convexes (si les contraintes sont déjà convexes) est le cadre d'Optimisation Alternée (AO) [SDLF+17] ou Block Coordinate Descent (BCD) [Tse01]. Dans l'AO, en fixant toutes les matrices-facteurs (par initialisation ou en utilisant leur estimation précédente) sauf une, on essaie de minimiser la fonction de coût sur une seule matrice-facteur

Par exemple, les étapes ALS pour une décomposition tensorielle contrainte d'ordre trois de dimensions $n_1 \times n_2 \times n_3$ et de rang R, $\boldsymbol{\mathcal{T}} = [\![\mathbb{1}_R; \boldsymbol{A}^{(1)}_{n_1 \times R}, \boldsymbol{A}^{(2)}_{n_2 \times R}, \boldsymbol{A}^{(3)}_{n_3 \times R}]\!]$, sous une forme dépliante, sont les suivants :

$$\begin{aligned}
\boldsymbol{A}^{(1)}_{k+1} &= \underset{\boldsymbol{A}_{n_1 \times R}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{\mathcal{T}}^{(1)}_{n_1 \times n_2 n_3} - \boldsymbol{A}(\boldsymbol{A}^{(3)}_k \odot \boldsymbol{A}^{(2)}_k)^T\|^2_F \text{ s.t. } \mathcal{C}_{\boldsymbol{A}}(\boldsymbol{A}) \\
\boldsymbol{A}^{(2)}_{k+1} &= \underset{\boldsymbol{B}_{n_2 \times R}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{\mathcal{T}}^{(2)}_{n_2 \times n_1 n_3} - \boldsymbol{B}(\boldsymbol{A}^{(3)}_k \odot \boldsymbol{A}^{(1)}_{k+1})^T\|^2_F \text{ s.t. } \mathcal{C}_{\boldsymbol{B}}(\boldsymbol{B}) \\
\boldsymbol{A}^{(3)}_{k+1} &= \underset{\boldsymbol{C}_{n_3 \times R}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{\mathcal{T}}^{(3)}_{n_3 \times n_1 n_2} - \boldsymbol{C}(\boldsymbol{A}^{(2)}_{k+1} \odot \boldsymbol{A}^{(1)}_{k+1})^T\|^2_F \text{ s.t. } \mathcal{C}_{\boldsymbol{C}}(\boldsymbol{C})
\end{aligned} \tag{B.17}$$

où $\mathcal{C}_{\cdot}(\cdot)$ représente la contrainte souhaitée sur les matrices facteurs.

B.4.2 Séparation Avant-Arrière

Dans cette section, nous expliquons l'opérateur proximal et les méthodes telles que le Séparation Avant-Arrière[2], qui est utilisé dans la méthode que nous proposons dans la Section B.4.4.

Opérateur de proximité [CP11] Pour tout $\boldsymbol{x} \in \mathbb{R}^N$, l'unique solution du problème de minimisation suivant :

$$\underset{\boldsymbol{y} \in \mathbb{R}^N}{\operatorname{argmin}} f(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2_2, \tag{B.18}$$

[2]Forward-Backward Splitting

est défini comme l'opérateur de proximité de la fonction $f \in \Gamma_0(\mathbb{R}^N)$, et il est désigné par $\text{prox}_f(\boldsymbol{x})$.

Dans de nombreuses applications de traitement du signal, la fonction de coût à minimiser est la somme de deux fonctions dont l'une est généralement non-différentiable ou même non-convexe. En suivant une approche proximale, ces types de problèmes peuvent être résolus à l'aide d'un algorithme particulier appelé *Séparation Avant-Arrière*.

Theorem 3 (Séparation Avant-Arrière [CP11])**.** *Supposons que* $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ *est une fonction propre*[3]*, semi-continue inférieurement, qui possède la propriété KL et est bornée par le bas. Si* f *peut être décomposée en deux parties sous la forme* $f = h + g$*, où* g *est semi-continue inférieurement et* $h : \mathbb{R}^N \mapsto \mathbb{R}$ *est une fonction différentiable à valeur finie avec un gradient continu* β*-Lipschitz,* i.e, $\exists \beta$ *tel que :*

$$\|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\|_2 \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|_2,$$

alors on peut montrer [CW05] que le minimiseur de f *satisfait l'équation de point fixe suivante :*

$$\boldsymbol{x} = prox_{\gamma g}(\boldsymbol{x} - \gamma \nabla h(\boldsymbol{x})), \tag{B.19}$$

où $\gamma \in (0, +\infty)$.

Plusieurs variantes d'implantation de l'agoritme de Séparation Avant-Arrière existent, et sont rapportées dans [CP11]. Deux d'entre elles sont reprises (Algorithme 7 et Algorithme 8) auxquelles nous ferons référence dans la suite de cette thèse.

L'algorithme 8 est basé sur l'algorithme Fast Iterative Shrinkage Thresholding (FISTA) proposé dans [BT09a, BT09b] et peut être considéré comme un algorithme de gradient proximal. En fait, au lieu de la fonction de rétrécissement dans FISTA, on utilise l'opérateur de proximité.

[3]Une fonction est propre si son domaine n'est pas un ensemble nul.

Algorithm 7 Séparation Avant-Arrière [CW05, CP11, Algorithm 10.5]

Entrée: La fonction $f = h + g$ telle que définie dans le Théorème 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Sortie: Le minimiseur de f

1: Fixer $\epsilon \in (0, \min\{1, \frac{1}{\beta}\})$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$
4: $\quad \boldsymbol{y}_k = \boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)$
5: $\quad \alpha_k \in [\epsilon, 1]$
6: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\text{prox}_{\gamma_k g}(\boldsymbol{y}_k) - \boldsymbol{x}_k)$
7: **end for**

La convergence de l'algorithme de Séparation Avant-Arrière dans le théorème 1 est prouvée dans [ABS13], et l'analyse de convergence fournie n'est pas seulement applicable pour des fonctions de coût continues et convexes, mais est également utilisable pour des fonctions non-lisses et non-convexes (ou un ensemble de contraintes non-convexes).

B.4.3 L'état de l'art

B.4.3.1 Optimisation Alternée - Méthode des Multiplicateurs à Direction Alternée (AO-ADMM) [HSL16]

L'algorithme AO-ADMM[4] tente de minimiser chaque étape de (5.9) par ADMM. Les détails de l'AO-ADMM pour la décomposition CP sous contrainte peuvent être trouvés dans [HSL16]. La convergence d'AO est brièvement examinée dans [HSL16] sur la base des travaux de [Tse01, RHL13]. La convergence de chaque étape, qui est un algorithme ADMM, peut être démontrée pour les fonctions convexes [HSL16, BPC+11] et aussi récemment pour certaines fonctions non-convexes telles que $\ell_q, \quad 0 < q < 1$ [WYZ19], mais pas encore pour certaines autres telles que ℓ_0.

[4] Alternating Optimization - Alternating Direction Method of Multipliers

Algorithm 8 Algorithme du gradient proximal de Beck-Teboulle [CP11, Algorithm 10.7] basé sur FISTA [BT09b]

Entrée: La fonction $f = h + g$ telle que définie dans le Théorème 1, β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Output: Le minimiseur de f

1: Set $\boldsymbol{z}_0 = \boldsymbol{x}_0$ and $t_0 = 1$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \boldsymbol{y}_k = \boldsymbol{z}_k - \beta^{-1}\nabla h(\boldsymbol{z}_k)$
4: $\quad \boldsymbol{x}_{k+1} = \text{prox}_{\beta^{-1}g}(\boldsymbol{y}_k)$
5: $\quad t_{k+1} = \dfrac{1 + \sqrt{4t_k^2 + 1}}{2}$
6: $\quad \lambda_k = 1 + \dfrac{t_k - 1}{t_{k+1}}$
7: $\quad \boldsymbol{z}_{k+1} = \boldsymbol{x}_k + \lambda_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$
8: **end for**

B.4.3.2 Gradient Proximal Alternatif (APG) [XY13b]

Dans [XY13b], la convergence d'un algorithme BCD général est étudiée, où pour mettre à jour chaque bloc de variables inconnues, trois types de mises à jour sont étudiés, à savoir : *original*, *proximal* et *prox-linear*.

D'après la comparaison effectuée dans [XY13b], la mise à jour prox-linéaire (i) donne de meilleures valeurs de la fonction objectif que la mise à jour originale et proximale, (ii) est plus facile à calculer et (iii) permet souvent des solutions sous forme explicite.

L'algorithme APG[5] ne suit pas explicitement la procédure de Séparation Avant-Arrière; néanmoins, pour satisfaire la contrainte de non-négativité, il se termine par une projection sur l'orthant non-négatif ainsi que par la mise à jour d'un coefficient particulier comme à la ligne 5 de l'Algorithme 8.

[5] Alternating Proximal Gradient

B.4.3.3 Factorisation tensorielle non-négative rapide-APG (FastNTF-APG) [ZZZ+16]

L'algorithme FastNTF-APG[6] [ZZZ+16] est une version modifiée de APG [XY13b] dédiée à la décomposition tensorielle non-négative. Dans [ZZZ+16], il est mentionné que, contrairement aux algorithmes classiques de factorisation tensorielle non-négative (NTF), qui souffrent d'une convergence lente en particulier dans les applications pratiques [JMC18], FastNTF-APG accélère la NTF et surmonte ce goulot d'étranglement en combinant APG avec l'approximation de rang faible.

Comme dans APG, il est également nécessaire dans FastNTF-APG de calculer la constante de Lipschitz du gradient de $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(1)} - \boldsymbol{A}(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T\|_F^2$ par rapport à $\boldsymbol{A}$ (selon (B.17)). Il est mentionné dans [ZZZ+16] que la constante de Lipschitz est $\|(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})^T(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})\|_F$, mais comme nous le prouvons dans l'annexe A, la constante de Lipschitz est en fait la *norme spectrale*[7] de $\{(\boldsymbol{A}_k^{(3)}) \odot \boldsymbol{A}_k^{(2)})^T(\boldsymbol{A}_k^{(3)} \odot \boldsymbol{A}_k^{(2)})\}$.

B.4.3.4 Métrique Variable Avant-Arrière par Bloc de Coordonnées (BC-VMFB) [CPR16, VCTMM17, VCTM+17]

L'algorithme BC-VMFB[8] consiste en deux étapes principales : une étape de gradient liée à la fidélité des données, qui est supposée être différentiable et avoir un gradient β-Lipschitz, et une étape proximale liée au terme de régularisation, pour laquelle un nouvel opérateur de proximité doit être calculé. On observe empiriquement dans [CMLZ18] que l'utilisation de cette nouvelle définition de l'opérateur de proximité accélère la convergence de la Minimisation Linéarisée Alternée Proximale (PALM) [BST14].

[6] Fast Non-negative Tensor Factorization-APG

[7] La norme spectrale d'une matrice est définie comme étant sa valeur singulière maximale [Ber05].

[8] Block Coordinate Variable Metric Forward-Backward

B.4.3.5 Méthode de la puissance itérée tensorielle robuste [AGH+14]

Dans [AGH+14], les auteurs proposent d'utiliser les deux moments ($\boldsymbol{P}$ et $\boldsymbol{\mathcal{T}}$) définis dans (B.1) et (B.2). La matrice $\boldsymbol{P}$ est théoriquement semi-définie positive, puisque $\boldsymbol{P}$ est une matrice de covariance (notez que φ_k sont des nombres non-négatifs dans (3.5)). Par conséquent, de manière similaire à ce qui a été fait pour la séparation aveugle de sources [CJ10, Com94], il existe une matrice de "blanchiment" $\boldsymbol{W}$ telle que $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{W} = \boldsymbol{I}$, où $\boldsymbol{I}$ est la matrice d'identité.

Ensuite, cette matrice de blanchiment est appliquée au tenseur $\boldsymbol{\mathcal{T}}$ pour donner :

$$\widetilde{\boldsymbol{\mathcal{T}}} \overset{\text{def}}{=} \boldsymbol{\mathcal{T}} \underset{1}{\bullet} \boldsymbol{W} \underset{2}{\bullet} \boldsymbol{W} \underset{3}{\bullet} \boldsymbol{W},$$

En d'autres termes, les composantes du tenseur blanchi sont, d'après (2.2) [Com14]:

$$\widetilde{\boldsymbol{\mathcal{T}}}(r,t,s) = \sum_{r',t',s'} \boldsymbol{\mathcal{T}}(r',t',s')\boldsymbol{W}(r,r')\boldsymbol{W}(t,t')\boldsymbol{W}(s,s').$$

Ce nouveau tenseur satisfait

$$\widetilde{\boldsymbol{\mathcal{T}}} = \sum_{k=1}^{K} \varphi_k^{-1/2}\ \tilde{\boldsymbol{a}}_k \otimes \tilde{\boldsymbol{a}}_k \otimes \tilde{\boldsymbol{a}}_k.$$

La conclusion est que $\widetilde{\boldsymbol{\mathcal{T}}}$ admet idéalement une décomposition CP *orthogonale*; voir *e.g.* [Com14, Com94] pour une introduction.

Les auteurs de [AGH+14] ont utilisé la méthode de la puissance itérée tensorielle [DLC+95] pour extraire le "vecteur propre" dominant[9], $\hat{\boldsymbol{a}}$, et la "valeur propre" dominante, $\widehat{\varphi}$, de $\widetilde{\boldsymbol{\mathcal{T}}}$ et ensuite procédé par déflation, *i.e.* $\widetilde{\boldsymbol{\mathcal{T}}} \leftarrow \widetilde{\boldsymbol{\mathcal{T}}} - \widehat{\varphi}\ \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}}$, pour obtenir les autres.

[9] Rappellez-vous qu'il existe plusieurs définitions des vecteurs propres du tenseur [Lim05, QL17]. La définition utilisée dans [AGH+14] – et donc ici – est $\boldsymbol{\mathcal{T}} \bullet \boldsymbol{v} \bullet \boldsymbol{v} = \lambda \boldsymbol{v}$, qui a la propriété indésirable que λ dépend de la norme de $\boldsymbol{v}$. En fait, si $(\lambda, \boldsymbol{v})$ est un couple propre, alors $(\alpha\lambda, \alpha\boldsymbol{v})$ l'est aussi pour tout α non nul. Ceci est analysé dans *e.g.* [QL17].

B.4.4 Algorithme proposé: Séparation Simple Avant-Arrière (SFBS)

Dans cette section, nous décrivons SFBS, la méthode que nous proposons pour la décomposition CP contrainte (*e.g.* non-négativité). Considérons le tenseur d'ordre N, $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ de rang R. Supposons que $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, où $\boldsymbol{\lambda} \in \mathbb{R}_+^R$ et $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$. Un problème général de décomposition CP contrainte de $\boldsymbol{\mathcal{T}}$ peut être formulé comme suit :

$$\min_{\boldsymbol{\lambda}, \boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2 \tag{B.20}$$
$$\text{s.t. } ; \mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)}), \ ; 1 \leq n \leq N,$$

où $\mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)})$ sont, respectivement, les contraintes sur le vecteur $\boldsymbol{\lambda}$ (y compris la contrainte mentionnée ci-dessus, *i.e.* $\boldsymbol{\lambda} \in \mathbb{R}_+^R$, comme l'appartenance à un simplex) et la matrice $\boldsymbol{A}^{(n)}$.

Comme mentionné dans la section B.4.3.1, une stratégie commune consiste à résoudre (B.20) via ALS. De plus, une optimisation avec contraintes peut être transformée en une optimisation sans contraintes en ajoutant la fonction indicatrice de l'ensemble des contraintes à la fonction de coût. Pour être plus précis, à la $n^{\text{ième}}$ étape de l'ALS pour résoudre (B.20), nous avons :

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \big\| \boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(n)}, \ldots, \boldsymbol{A}^{(N)}]\!] \big\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}), \tag{B.21}$$

où $i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)})$ est défini comme suit :

$$i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}) = \begin{cases} 0 & if \quad \boldsymbol{A}^{(n)} \in \mathcal{C}_{\boldsymbol{A}^{(n)}} \\ \infty & if \quad \boldsymbol{A}^{(n)} \notin \mathcal{C}_{\boldsymbol{A}^{(n)}} \end{cases}.$$

Définissons $\boldsymbol{W} \triangleq (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$. Alors en développant en mode-n l'expression (5.11), on a :

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)} \boldsymbol{W}\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}). \tag{B.22}$$

Toutes les hypothèses requises du Théorème 3 sont satisfaites pour (B.22), et selon ce théorème, le minimiseur de (B.22) est le point de convergence de

Algorithm 9 L'algorithme de SFBS

Entrée: $\boldsymbol{\mathcal{T}}$, $\mathcal{C}_{\boldsymbol{A}^{(n)}}$, initiale $\boldsymbol{A}_0^{(n)},\ n \in [1,\ldots,N]$, e

Sortie: Estimation de $\boldsymbol{A}^{(n)},\ n \in [1,\ldots,N]$

repeat
  for $n = 1, 2, \ldots, N$ **do**
    $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$
    $\beta = \{\max(\text{valeur singulière}(\boldsymbol{W}))\}^2$
    définir $\gamma = \frac{e}{\beta}$ et choisir α_g.
    for $g = 0, 1, 2, \ldots$ **do**
      $\boldsymbol{Y} = \boldsymbol{A}_g^{(n)} - \gamma(\boldsymbol{A}_g^{(n)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{\mathcal{T}}^{(n)}\boldsymbol{W}^T)$
      $\boldsymbol{A}_{g+1}^{(n)} = \boldsymbol{A}_g^{(n)} + \alpha_g(\text{proj}_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{Y}) - \boldsymbol{A}_g^{(n)})$
    end for
  end for
until Un certain critère de fin

l'équation à point fixe suivante :

$$\boldsymbol{A}^{(n)} = \text{prox}_{\gamma i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}}\{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T})\}.$$

La SFBS est décrite dans l'Algorithme 9. Comme elle est exprimée dans l'Algorithme 9, nous ignorons ϵ et fixons la valeur de $\gamma_k = \frac{e}{\beta}$, où $\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$ désigne la norme spectrale de la matrice $\boldsymbol{W}\boldsymbol{W}^T$, dans toutes les itérations sur k (nous avons observé expérimentalement que $e = 1.4$ ou $e = 1.9$ sont presque toujours des valeurs correctes).

SFBS est également assez facile à comprendre et à mettre en œuvre, c'est pourquoi nous l'appelons *Simple* Forward-Backward Splitting (SFBS). Bien que des descriptions détaillées des avantages de SFBS par rapport aux autres méthodes (discutées dans cette thèse) soient fournies dans la Section 5.7, nous les énumérons brièvement ci-dessous pour compléter la présente section:

- Contrairement à d'autres méthodes sous contraintes, il est expliqué comment la SFBS peut gérer une variété de contraintes telles que l'appartenance à un simplex de toutes les matrices facteurs ainsi que

du vecteur $\boldsymbol{\lambda}$,

- Contrairement à la méthode de la puissance tensorielle robuste, la SFBS est capable de gérer les cas "sur-complets" (lorsque le nombre de variables cachées, K, est supérieur au nombre de variables multi-vues, D) dans l'estimation des probabilités (cf. Section 3.3.1),

- Par rapport à APG, qui est l'une des méthodes les plus efficaces basées sur le concept proximal, SFBS est plus performante dans un scénario bruité en termes d'erreur de reconstruction relative et d'estimation des matrices facteurs,

- Contrairement à AO-ADMM, une analyse de convergence complète est fournie dans la section 5.5.4.

Garantie de convergence La convergence globale d'AO lorsque la minimisation à chaque étape est effectuée au moyen du concept proximal est prouvée dans [XY13b,BST14]. L'analyse de convergence fournie dans [BST14] convient à l'algorithme basé sur Proximal Forward-Backward; en conséquence nous renvoyons le lecteur, pour la convergence globale de SFBS (qui est également basée sur Proximal Forward-Backward), à l'analyse effectuée dans [BST14], à savoir, Proximal Alternating Linearized Minimization (PALM). Dans la section 5.5.4, nous avons discuté de la façon dont les hypothèses de convergence requises de PALM sont satisfaites pour SFBS.

La garantie de convergence de chaque étape de SFBS résulte d'un théorème établi dans [ABS13], qui traite de la convergence du fractionnement avant-arrière, et les hypothèses de ce théorème sont complètement satisfaites pour chaque étape de SFBS.

B.4.5 Résultats de la simulation

B.4.5.1 Contrainte de non-négativité

Dans toutes les simulations de cette section, $\epsilon_1 = 10^{-20}$, $e = 1.9$ et nous répétons cinq fois la boucle de la ligne 6 pour chaque mode dans l'Algorithme 9 (algorithme SFBS).

La figure B.7 montre l'erreur relative de reconstruction du tenseur sans bruit de taille $10 \times 10 \times 10$, de rang $R = 6$, avec un nombre d'initialisations, un nombre de moyennes, et un nombre d'itérations (max-number), égaux à 10, 10 et 5000, respectivement (cf. voir page 101 pour la définition de ces paramètres). Comme on peut le voir à première vue, BC-VMFB, malgré nos efforts pour ajuster ses paramètres correctement, résulte en une erreur relative autour de 1, ce qui signifie 100% (rappelons que l'erreur relative n'est pas indiquée sous forme de pourcentage). Bien que nous ayons consulté l'auteur correspondant de BC-VMFB au sujet des paramètres de leur algorithme, nous avons trouvé difficile d'ajuster plusieurs paramètres de BC-VMFB. En fait, non seulement ces paramètres dépendent des données, mais ils ont également un effet critique sur le résultat final de la méthode. Par conséquent, il semble inutile d'inclure l'algorithme BC-VMFB dans le reste des comparaisons de cette simulation. De plus, puisque FastNTF-APG essaie de décomposer une approximation de rang faible du tenseur désiré, on peut s'attendre à ce que la performance de FastNTF-APG soit pire que celle de APG. Par conséquent, nous supprimons FastNTF-APG dans certaines comparaisons de ce chapitre. Bien que la plupart des méthodes atteignent des performances raisonnables, APG et SFBS convergent plutôt plus rapidement.

Dans la Fig. B.8, l'écart entre $\boldsymbol{X}$ et $\widehat{\boldsymbol{X}}$ via CorrIndex et l'algorithme hongrois (qui est une erreur exacte) est indiqué pour la même simulation que la Fig. B.7. Comme on peut le voir sur la Fig. B.8, la même interprétation que celle de la Fig. B.7 peut être conclue à partir de "CorrIndex" et "Hungarian".

La figure B.9 indique l'erreur relative de reconstruction d'un tenseur

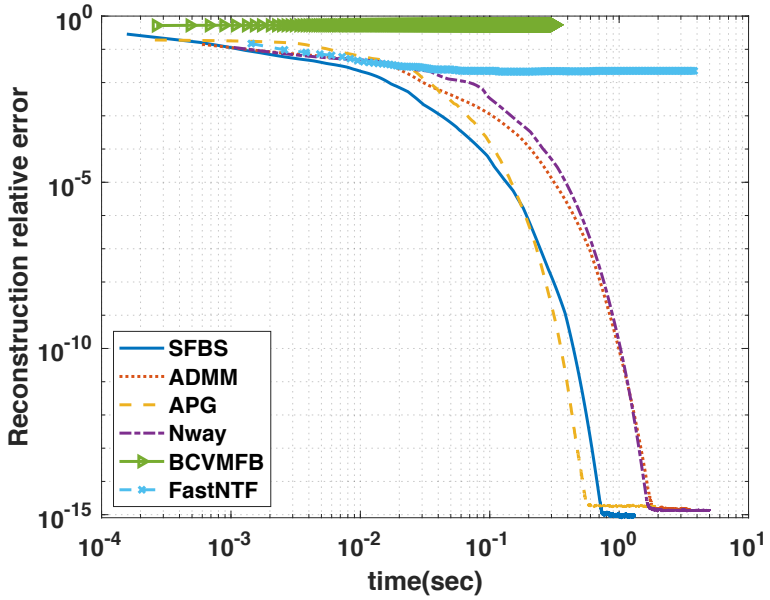

Figure B.7: Erreur relative de reconstruction lors de la décomposition d'un tenseur de dimensions $10 \times 10 \times 10$, de rang $R = 6$ sous la contrainte de non-négativité sur tous les matrices facteurs, dans le cas sans bruit avec les paramètres suivants: $\epsilon_1 = 10^{-20}$, nombre moyen= 10, nombre d'initialisation= 10, itérations max-number= 5000, $e = 1.9$.

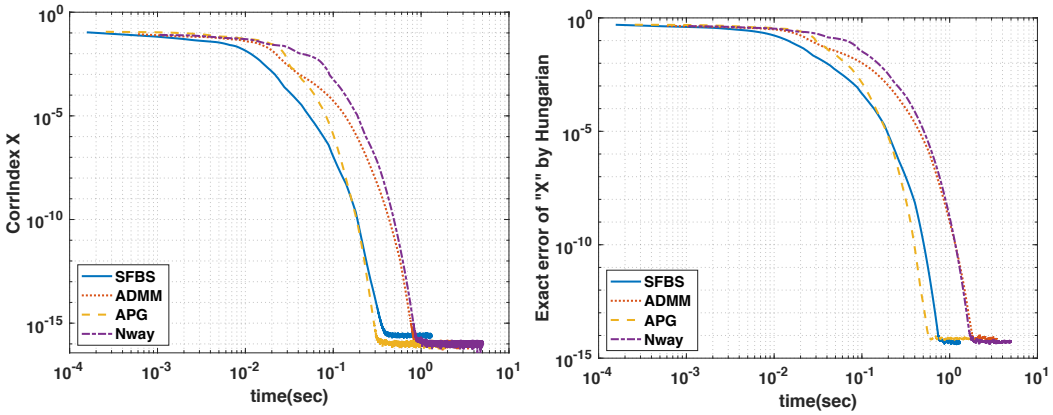

Figure B.8: Comparez l'estimation de la matrice $\boldsymbol{X}$ via CorrIndex (à gauche) et Hungarian (à droite) dans la même expérience que Fig. B.7.

bruité de taille $10 \times 10 \times 10$, de rang $R = 6$ avec SNR $= 10$, nombre d'initialisations $= 20$, nombre de moyennes $= 200$ et nombre maximal d'itérations $= 1000$. La figure B.9 (Gauche) compare toutes les méthodes ensemble, tandis que la figure B.9 (Droite) exclut BC-VMFB afin de montrer plus précisément les résultats des autres méthodes.

Comme dans la Fig. B.7, BC-VMFB ne fonctionne pas correctement dans l'expérience de la Fig. B.9. De plus, bien que Nway s'en sorte bien dans un cas sans bruit, sa performance n'est pas acceptable dans la situation bruitée (cf. Fig. B.9). Comme mentionné précédemment, FastNTF-APG remplace le tenseur bruité par son approximation de rang faible, ce qui permet de filtrer le bruit. Comme on peut le voir sur la Fig. B.9, le résultat de FastNTF-APG est meilleur que APG dans un scénario bruité, alors que APG est l'une des meilleures méthodes pour la décomposition d'un tenseur sans bruit (cf. Fig. B.7). De plus, SFBS et AO-ADMM surpassent les autres méthodes dans l'expérience de la Fig. B.9. Par conséquent, le seul algorithme qui fonctionne correctement et mieux que les autres dans les situations bruitées et non bruitées est SFBS.

Dans la Fig. B.10, l'écart entre $\boldsymbol{X}$ et $\widehat{\boldsymbol{X}}$ via CorrIndex et l'algorithme hongrois (qui est l'erreur exacte) est indiqué pour la même expérience que la Fig. B.9. Bien que selon l'erreur de reconstruction relative de la Fig. B.9, AO-ADMM et SFBS surpassent les autres, dans l'estimation des matrices facteurs (cf. Fig. B.10), SFBS est plus performant que AO-ADMM. La performance sur l'estimation des matrices facteurs est critique dans de nombreuses applications telles que l'exploration de données et l'exploration de textes décrites dans le chapitre 3. De plus, les performances de Nway étant insuffisantes dans le cas bruité, cela révèle l'importance de prendre en compte les contraintes dans la décomposition des tenseurs (rappelons que Nway exécute simplement une décomposition CP non-contrainte).

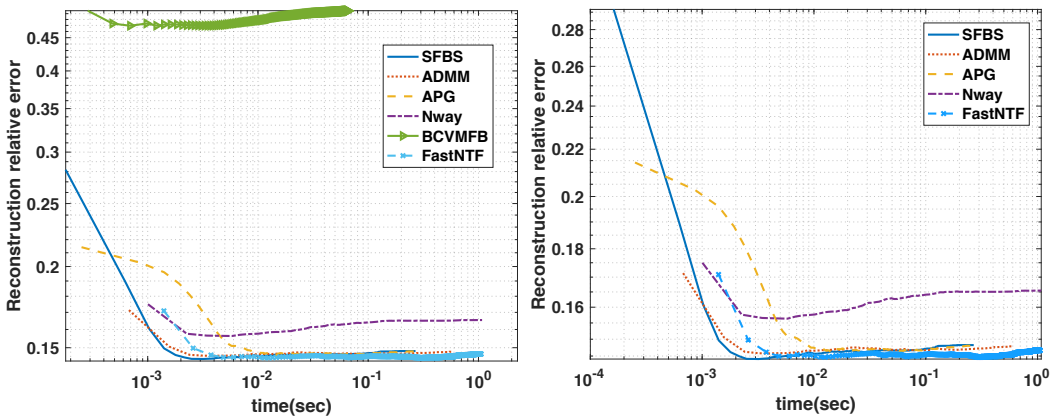

Figure B.9: Erreur relative de reconstruction incluant BC-VMFB (gauche) et excluant BC-VMFB (droite) dans la décomposition d'un tenseur de dimensions $10 \times 10 \times 10$, de rang $R = 6$ sous la contrainte de non-négativité sur tous les matrices facteurs, dans un cas bruité avec SNR $= 10$ et avec les paramètres suivants: $\epsilon_1 = 10^{-20}$, nombre moyennes $= 200$, nombre d'initialisations $= 20$, nombre maximal d'itérations $= 1000$, $e = 1.9$.

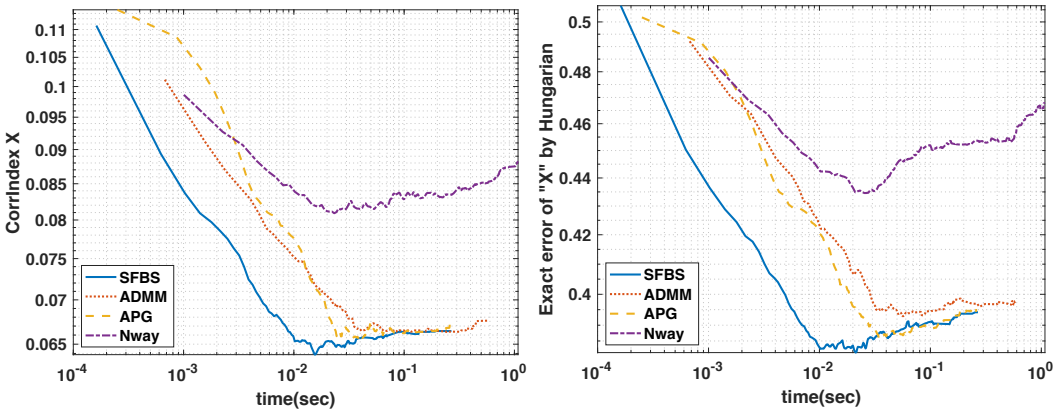

Figure B.10: Comparaison entre l'estimation de la matrice $\boldsymbol{X}$ via CorrIndex (à gauche) et Hungarian (à droite) dans la même expérience que la Fig. B.9.

B.4.5.2 Contrainte du simplex

Dans cette section, nous étudions de manière pratique les performances des algorithmes de décomposition tensorielle sous la contrainte d'appartenance au simplex des probabilités, sur toutes les colonnes de toutes les matrices facteurs et sur $\boldsymbol{\lambda}$. Dans les simulations de cette section, nous fixons $\epsilon_1 = 10^{-20}$ et répétons cinq fois la boucle de la ligne 6 pour chaque mode de l'Algorithme 9. (algorithme SFBS).

Dans cette section, nous comparons SFBS avec AO-ADMM, puisque, premièrement, la contrainte du simplexe est directement considérée et étudiée par les auteurs dans [HSL16] ainsi que dans leur implantation. Deuxièmement, selon les résultats sous la contrainte de non-négativité dans la section 5.6.1.1, AO-ADMM est l'algorithme qui a les performances les plus proches de celles de SFBS dans les cas bruités et non bruités.

La méthodologie d'application de la contrainte du simplex sur toutes les colonnes de toutes les matrices facteurs en plus du vecteur de coefficients ($\boldsymbol{\lambda}$) est expliquée dans la section 5.5.2.2, et nous avons utilisé cette méthodologie dans les simulations de la présente la section.

Cas sans bruit. La figure B.11 montre l'erreur relative de reconstruction du tenseur sans bruit de taille $10 \times 10 \times 10$, de rang $R = 3$ avec un nombre d'initialisations, un nombre de moyennes, e et un nombre maximal d'itérations, égaux à 10, 10, 1.5 et 20000, respectivement, sous la contrainte du simplex (des probabilités) sur toutes les colonnes de toutess les matrices facteurs en plus de $\boldsymbol{\lambda}$[10]. En comparant la Fig. B.11 et la Fig. B.7, on peut en déduire que le simplex est une contrainte plus difficile à atteindre que la non-négativité seule, puisque le maximum d'itérations requises pour la contrainte du simplex est de 20000, et c'est plus que ce qui est fixé pour la contrainte de non-négativité (*i.e.* 5000) pour la convergence des algorithmes. Comme on peut le voir sur la Fig. B.11, SFBS et AO-ADMM atteignent tous deux le même niveau d'erreur relative, mais SFBS converge légèrement plus

[10]Voir page 101 et Algorithme 5 pour la définition de ces paramètres.

vite.

Dans la Fig. B.12, l'écart entre $\boldsymbol{X}$ et $\widehat{\boldsymbol{X}}$ via CorrIndex et l'algorithme hongrois (qui est l'erreur exacte) est indiqué pour la même expérience que la Fig. B.11. La même conclusion que précédemment peut être tirée de l'estimation des matrices facteurs, c'est-à-dire que SFBS et AO-ADMM sont tous deux performants, cependant, SFBS converge un peu plus rapidement.

Cas bruités. La figure B.13 montre l'erreur relative de reconstruction d'un tenseur de taille $10 \times 10 \times 10$, de rang $R = 3$ avec SNR, nombre d'initialisations, nombre de moyennes, e et nombre d'itérations max-number, égaux à 10, 20, 200, 1.9 et 1000, respectivement, sous les contraintes du simplex sur toutes les colonnes de tous les matrices facteurs en plus de $\boldsymbol{\lambda}$. La figure B.14 représente l'écart entre $\boldsymbol{X}$ et $\widehat{\boldsymbol{X}}$ via CorrIndex et l'algorithme hongrois (qui est l'erreur exacte) pour la même expérience de la Fig. B.13. Les Fig. B.13 et Fig. B.14 montrent que SFBS est plus performant que AO-ADMM et converge légèrement plus rapidement.

La figure. B.15 montre l'écart entre $\boldsymbol{\lambda}$ et $\widehat{\boldsymbol{\lambda}}$ via CorrIndex et l'algorithme hongrois (qui est l'erreur exacte) pour la même expérience que la figure. B.13. On peut tirer la même conclusion que précédemment, c'est-à-dire que SFBS fonctionne et converge mieux et plus rapidement que AO-ADMM. On peut voir que l'erreur relative dans la Fig. B.15 augmente légèrement. On note que la diminution monotone de l'erreur de reconstruction relative (ou fonction objectif) est essentielle, et non pas pour l'estimation des matrices facteurs ou $\boldsymbol{\lambda}$, ce qui est le cas dans toutes les expériences telles que celle exprimée dans la Fig. B.13.

Sensibilité au bruit additif. Dans la suite, nous montrerons expérimentalement que la méthode de la puissance tensorielle robuste n'est pas performante en présence de bruit additif. Pour ce faire, nous avons généré des matrices arbitraires $\boldsymbol{A}^{10\times 3}$ et des vecteurs $\boldsymbol{\varphi}^{3\times 1}$. Ensuite, nous avons calculé leurs moments d'ordre 3 et 2 originaux correspondants, *i.e.* $\boldsymbol{\mathcal{T}}$ et $\boldsymbol{P}$ (cf. (B.3) et (B.4)). Enfin, nous avons appliqué plusieurs algorithmes de décomposition

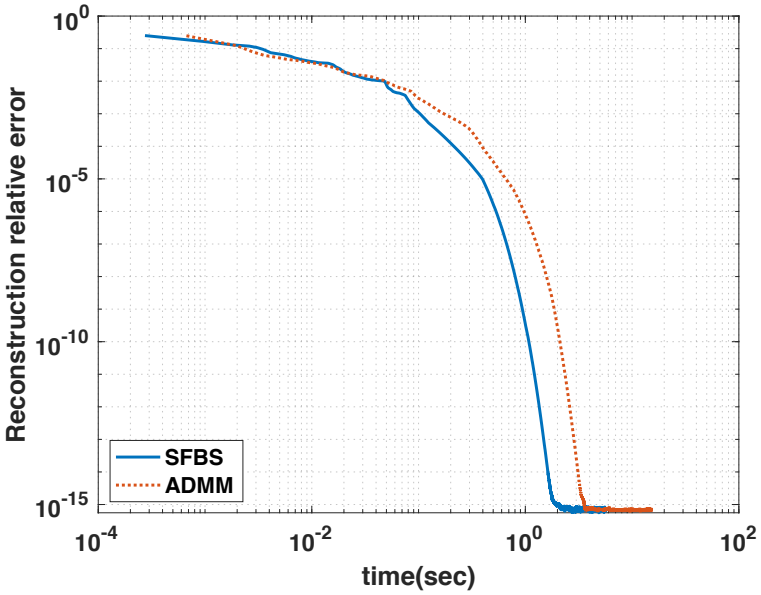

Figure B.11: Erreur relative de reconstruction lors de la décomposition d'un tenseur de dimensions $10 \times 10 \times 10$, de rang $R = 3$ sous les contraintes du Simplex sur toutes les colonnes de toutes les matrices facteurs et sur $\boldsymbol{\lambda}$, dans un cas sans bruit avec les paramètres suivants : $\epsilon_1 = 10^{-20}$, nombre de moyennes $= 10$, nombre d'initialisations $= 10$, nombre maximal d'itérations max-number $= 20000$, $e = 1.5$.

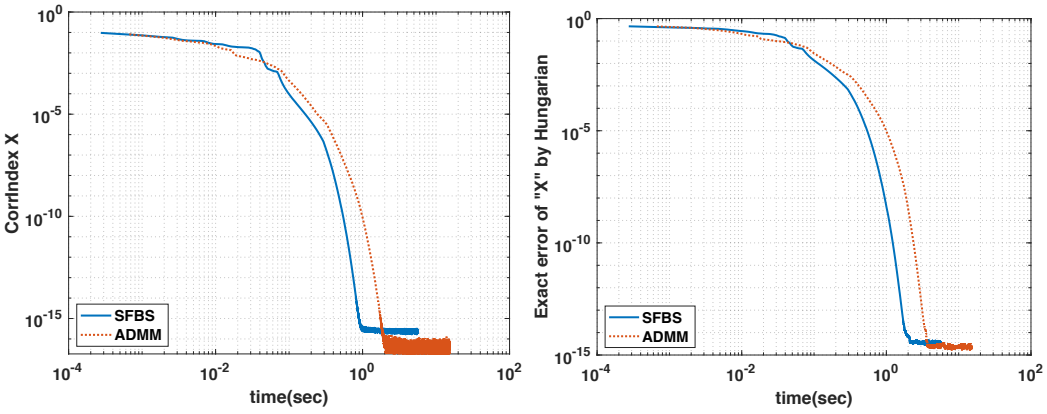

Figure B.12: Comparaison entre l'estimation de la matrice $\boldsymbol{X}$ via CorrIndex (à gauche) et Hungarian (à droite) dans la même expérience que la Fig.B.11.

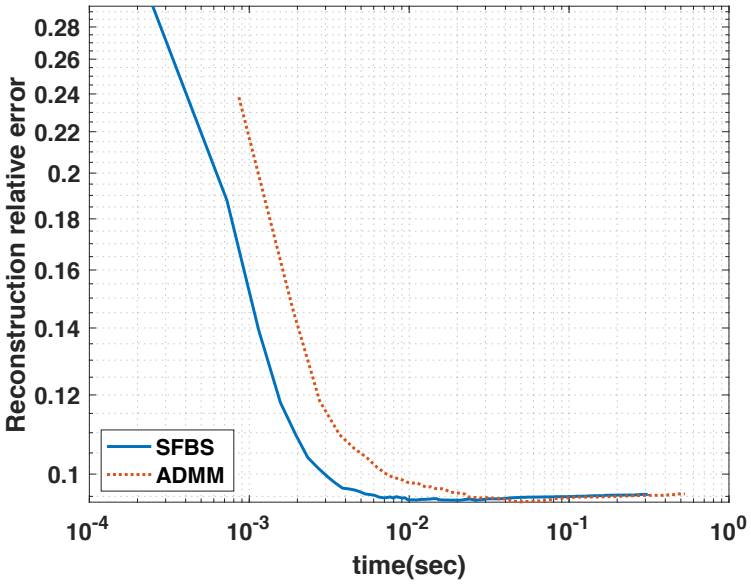

Figure B.13: Erreur relative de reconstruction lors de la décomposition d'un tenseur de dimensions $10 \times 10 \times 10$, de rang $R = 3$ sous les contraintes du simplex sur toutes les colonnes de toutes les matrices facteurs et sur $\boldsymbol{\lambda}$, SNR $= 10$ avec le paramétrage suivant : $\epsilon_1 = 10^{-20}$, nombre moyennes $= 200$, nombre d'initialisation$= 20$, itérations max-number$= 1000$, $e = 1.9$.

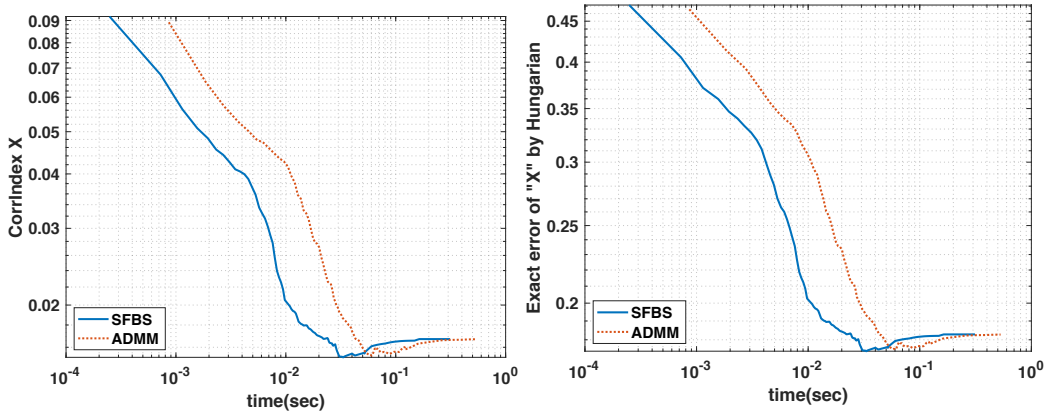

Figure B.14: Comparaisonentre l'estimation de la matrice $\boldsymbol{X}$ via CorrIndex (à gauche) et Hungarian (à droite) dans la même expérience que la Fig. B.13.

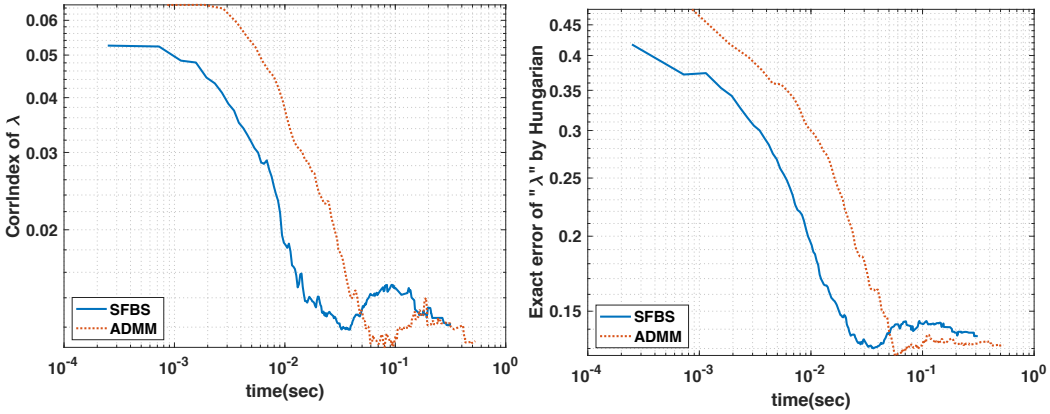

Figure B.15: Comparaison entre l'estimation du vecteur $\boldsymbol{\lambda}$ via CorrIndex (à gauche) et Hungarian (à droite) dans la même expérience que la Fig. B.13.

tensorielle, dont la méthode de puissance tensorielle robuste, sur les versions bruitées et non bruitées (Fig. B.16 et Fig. B.17) de $\boldsymbol{\mathcal{T}}$ et $\boldsymbol{P}$.

Bien que la méthode de la puissance et la méthode de la puissance projetée soient très performantes dans le cas sans bruit, les figures B.16 et B.17 montrent que ces algorithmes sont très sensibles au bruit additif dans $\boldsymbol{\mathcal{T}}$ et $\boldsymbol{P}$.

La figure B.16 (Gauche) montre l'erreur de reconstruction relative des algorithmes en fonction d'une gamme de valeurs de SNR, *i.e.* $[10, 20, 30, 40]$. Comme on peut le constater, la méthode "Power" est très sensible au bruit additif, et son erreur de reconstruction relative peut atteindre des valeurs très élevées (comme 500). Dans la Fig. B.16 (droite), afin de distinguer les performances des autres algorithmes, nous supprimons la méthode Power. La figure B.16 (Droite) exprime que même en projetant le résultat de la méthode Power sur le simplex (méthode Projected Power), la performance de la méthode Power ne peut pas devenir acceptable.

B.4.5.3 Comparaison entre les performances pour les données synthétiques avec des relations cachées

L'effet de la contrainte sur la performance. Nous présentons dans cette section les résultats de comparaison obtenus avec une seule taille de dictionnaire, $D = 8$, avec un nombre fixe de sujets, $K = 4$, et pour différentes tailles de corpus jusqu'à $N_c = 2^{17}$. De plus, les résultats sont obtenus pour une réalisation particulière de la matrice $\boldsymbol{A}$ et du vecteur $\boldsymbol{\varphi}$ (des résultats similaires sont toutefois obtenus avec d'autres choix de $\boldsymbol{A}$ et de $\boldsymbol{\varphi}$).

Les résultats sont présentés dans la Fig. B.18 ; chaque cercle noir (resp. croix rouge) correspond à l'erreur de la méthode de puissance tensorielle robuste (resp. non-négative AO-ADMM) dans l'estimation de $\boldsymbol{\varphi}$ ou $\boldsymbol{A}$ pour un corpus spécifique de documents avec une taille de corpus particulière, N_c. Pour faciliter la comparaison, la médiane (resp. écart-type) parmi les corpus de documents est également représentée par une ligne pleine (resp. pointillés) pour chaque taille de corpus.

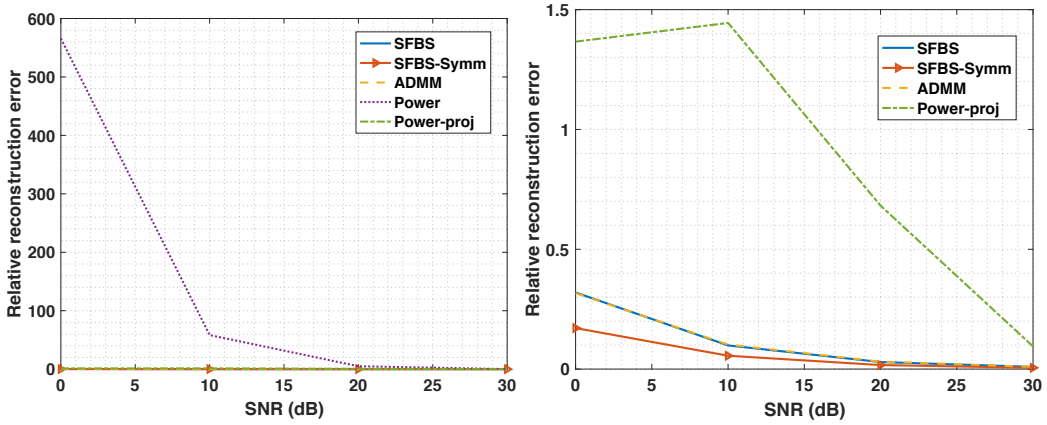

Figure B.16: Gauche: erreur de reconstruction relative de la décomposition des moments d'ordre 3 incluant la méthode Power. A droite: zoom sur la gauche, sans la méthode Power. La moyenne des résultats de la décomposition des *noisy* $\boldsymbol{\mathcal{T}}$ selon une plage de valeurs SNR, *i.e.* $[10, 20, 30, 40]$ sur 200 de réalisations de $\boldsymbol{A}$ et $\boldsymbol{\varphi}$, révèle que les algorithmes contraints tels que SFBS sont bien plus performants que la méthode Power et ses variantes.

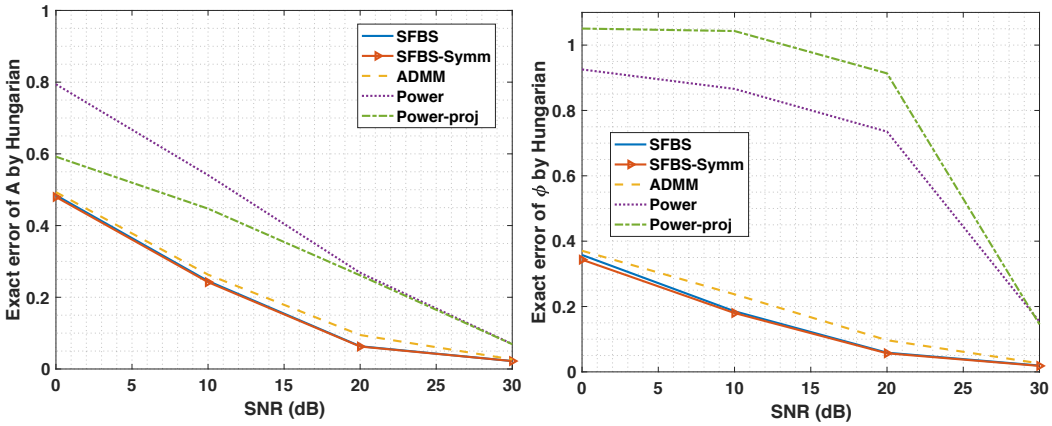

Figure B.17: Gauche: erreur relative de l'estimation de $\boldsymbol{A}$ mesurée par l'algorithme hongrois dans la même expérience que la Fig. B.16. Droite: erreur relative de l'estimation de $\boldsymbol{\varphi}$ mesurée par l'algorithme hongrois dans la même expérience que la Fig. B.16. La moyenne des résultats de la décomposition de *noisy* $\boldsymbol{\mathcal{T}}$ selon une plage de valeurs SNR, *i.e.* $[10, 20, 30, 40]$ sur 200 réalisations de $\boldsymbol{A}$ et $\boldsymbol{\varphi}$, révèle que les algorithmes contraints tels que SFBS sont beaucoup plus performants que la méthode Power et ses variantes. De plus, SFBS Simplex Symétrique et SFBS Simplex sont également un peu plus performants que AO-ADMM.

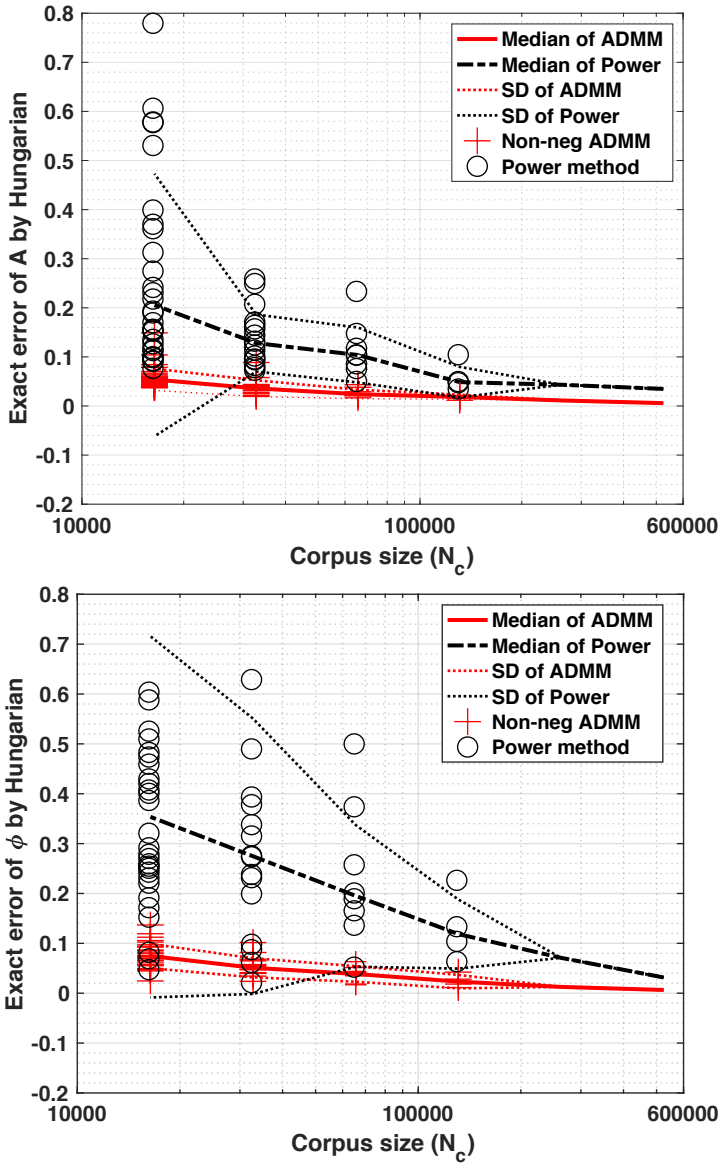

Figure B.18: Comparaison des performances de la méthode de puissance tensorielle robuste et de la méthode AO-ADMM non-négative dans l'estimation des probabilités, *i.e.* $\boldsymbol{A}$ (en haut) et $\boldsymbol{\varphi}$ (en bas), en fonction de la taille du corpus jusqu'à $N_c = 2^{17}$, lorsque l'estimateur de moment est une simple moyenne. Le nombre de mots et de sujets est fixé à $D = 8$ et $K = 4$, respectivement. Chaque cercle noir (resp. croix rouge) correspond à l'erreur de la méthode de puissance tensorielle robuste (resp. AO-ADMM non-négative) dans l'estimation de $\boldsymbol{\varphi}$ ou $\boldsymbol{A}$ pour un corpus spécifique de documents avec une taille de corpus particulière, N_c. La médiane (resp. écart-type) parmi les corpus de documents est également tracée en ligne pleine (resp. pointillés) pour chaque taille de corpus.

Comme le montre la Fig. B.18, l'erreur de la méthode contrainte telle que l'AO-ADMM non-négative converge plus rapidement vers zéro lorsque la taille du corpus augmente, et contrairement à la méthode de puissance tensorielle robuste, l'AO-ADMM non-négative semble être beaucoup plus robuste en raison de son plus petit écart-type de l'erreur relative. De plus, la Fig.B.18 (en bas) révèle que la différence de performance et d'écart type de la méthode de puissance tensorielle robuste est beaucoup plus importante dans l'estimation de $\boldsymbol{\varphi}$. Par conséquent, la prise en compte d'une contrainte appropriée (telle que la non-négativité) dans la décomposition est très efficace, notamment pour l'estimation de $\boldsymbol{\varphi}$.

B.4.6 Expériences sur des données réelles

Dans cette section, nous décrivons les expérimentations réalisées sur une partie d'un ensemble de données textuelles bien connu, à savoir 20 Newsgroups, qui consiste en 11314 messages sur 20 sujets disponibles en ligne [Lan95a, Lan95b, Nig00].

En effectuant quelques étapes de prétraitement sur l'ensemble des données et en conservant les mots ayant une fréquence supérieure à 20% de la fréquence moyenne du corpus (term-document), nous obtenons un dictionnaire de taille 14×14 ($D = 14$ mots). Étant donné que la méthode Power et ses variantes ne peuvent pas traiter les cas surcomplets ($K > D$), nous sommes contraints de choisir $K = 14$ sujets plutôt que $K = 20$, bien que $K = 20$ soit plus approprié étant donné que 20 sujets existent dans l'ensemble de données considéré ($N_c = 11314$ messages sur 20 sujets). Par conséquent, le sac de mots résultant (cf. (B.7)) est une matrice de dimensions 14 par 11314.

Bien que dans ce corpus, le nombre de documents pour tous les sujets soit presque le même (ce qui signifie que $\boldsymbol{\varphi}$ devrait avoir une distribution proche de la distribution uniforme), le vecteur $\boldsymbol{\varphi}$ estimé par la méthode de puissance robuste, *i.e.* $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$, n'est pas seulement non uniforme, mais n'est pas non plus une estimation de distribution de probabilité, puisque ses valeurs ne se

trouvent pas dans le simplex de probabilités:

$$\widehat{\boldsymbol{\varphi}}_{\text{Power}} = [0.00, 0.02, 0.14, 0.03, 0.08, 0.26, 0.39, \\ 0.35, 0.65, \mathbf{302.89}, 1.01, 0.87, 8.30, 0.95]^T.$$

Même si on projette $\widehat{\boldsymbol{\varphi}}_{\text{Power}}$ sur le simplexe comme mentionné précédemment, et que l'on obtient $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$, cela n'aurait pas de sens; en effet "$\boldsymbol{\varphi}$ est censé avoir une distribution proche de la distribution uniforme", puisque nous avons :

$$\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}} = \Big[0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{1}, 0, 0, 0, 0\Big]^T.$$

D'autre part, le $\boldsymbol{\varphi}$ estimé par SFBS, *i.e.* $\widehat{\boldsymbol{\varphi}}_{\text{SFBS}}$, est le suivant :

$$\widehat{\boldsymbol{\varphi}}_{\text{SFBS}} = [0.10, 0.08, 0.09, 0.09, 0.10, 0.04, 0.02, \\ 0.05, 0.06, 0.01, 0.06, 0.09, 0.08, 0.12]^T,$$

qui est plus proche de la distribution uniforme que $\widehat{\boldsymbol{\varphi}}_{\text{Power-proj}}$. Cela prouve à nouveau que l'application de la décomposition tensorielle contrainte est plus fiable que la décomposition non contrainte.

B.5 Conclusion et perspectives

Dans cette thèse, nous nous sommes concentrés sur un modèle de données particulier, appelé *single topic* (cf. Fig. B.1), constitué de quelques variables multi-vues qui sont reliées entre elles par une variable cachée/latente. Un document peut être décrit correctement par ce modèle lorsque ses mots jouent le rôle de variables multi-vues, qui sont liées les unes aux autres en fonction du sujet de ce document en tant que variable cachée.

Nous avons étudié les applications de la décomposition Canonique Polyadique (CP) pour l'estimation des probabilités des variables cachées et des probabilités conditionnelles des variables multi-vues. Nous avons expliqué qu'il y a deux étapes principales dans l'estimation des probabilités via une approche

tensorielle: premièrement, estimer le tenseur des moments du troisième ordre en observant les données qui obéissent au modèle à sujet unique; deuxièmement, décomposer le tenseur des moments du troisième ordre estimé au moyen d'algorithmes de décomposition tensorielle appropriés.

Nous avons étudié chaque étape nécessaire à l'estimation de probabilité via une approche tensorielle. Nous avons étudié théoriquement et expérimentalement les défis relevés dans l'état de l'art sur les estimations de moments et sur les décompositions tensorielles. En outre, nous avons proposé un estimateur de moments (moyenne standard), un nouvel indice de performance (CorrIndex) et un algorithme de décomposition tensorielle (SFBS), qui présentent certains avantages par rapport aux méthodes et indices existants.

Le principal inconvénient des algorithmes de décomposition tensorielle sous contrainte pour les grandes dimensions (typiquement de l'ordre de $D > 50$) est qu'ils prennent du temps en raison du calcul du produit de Khatri-Rao des matrices facteurs. Par conséquent, une future extension de SFBS serait de suivre l'idée de CPRAND (décomposition CP randomisée) efficace pour des tenseurs de rang faible, dans chaque itération de SFBS.

Bien que nous nous soyons concentrés sur le modèle à sujet unique dans cette thèse, le contenu peut être étendu à d'autres modèles de données tels que LDA [BNJ03] et les modèles de Markov [AGH$^+$14].

Bibliography

[ABRS10] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of operations research*, 35(2):438–457, 2010.

[ABS13] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[ACY+96] Shun-Ichi Amari, Andrzej Cichocki, Howard Hua Yang, et al. A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, pages 757–763. Morgan Kaufmann Publishers, 1996.

[AFH+12] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. Liu. A spectral algorithm for latent Dirichlet allocation. In *NIPS*, volume 25, pages 1–9, 2012.

[AGH+14] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *J. Machine Learning Research*, 15:2773–2832, August 2014.

[AHESK10] Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379, 2010.

[AHJK15] A. Anandkumar, D. Hsu, M. Janzamin, and S. Kakade. When are overcomplete topic models identifiable? uniqueness of tensor Tucker decompositions with structured sparsity. *Jour. Machine Learning Research*, 16:2643–2694, December 2015.

[AHK12a] A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. In *25th Conf. Learning Theory*, volume 23, pages 33.1–33.34, 2012.

[AHK12b] Animashree Anandkumar, Daniel Hsu, and Sham M Kakade. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1. JMLR Workshop and Conference Proceedings, 2012.

[BAC+14] Hanna Becker, Laurent Albera, Pierre Comon, Martin Haardt, Gwénaël Birot, Fabrice Wendling, Martine Gavaret, Christian G Bénar, and Isabelle Merlet. Eeg extended source localization: tensor-based vs. conventional methods. *NeuroImage*, 96:143–157, 2014.

[BBK18] Casey Battaglino, Grey Ballard, and Tamara G Kolda. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

[Ber05] D. S. Bernstein. *Matrix Mathematics*. Princeton Univ. Press, 2005.

[BK03] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(5):274–286, 2003.

[BK+21] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 3.2. Available online, April 2021.

[Ble12] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BPC+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[Bro98] Rasmus Bro. Multi-way analysis in the food industry. *Models, Algorithms, and Applications. Academish proefschrift. Dinamarca*, 1998.

[BST14] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[BT97] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[BT09a] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434, 2009.

[BT09b] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[CBDC09] P. Comon, J. M. F. Ten Berge, L. DeLathauwer, and J. Castaing. Generic and typical ranks of multi-way arrays. *Linear Algebra Appl.*, 430(11–12):2997–3007, June 2009. hal-00410058.

[CGLM08] P. Comon, G. Golub, L-H. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis Appl.*, 30(3):1254–1279, September 2008. hal-00327599.

[CJ10] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[CKAC14] Julie Coloigner, Ahmad Karfoul, Laurent Albera, and Pierre Comon. Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors. *Linear Algebra and its applications*, 450:334–374, 2014.

[CLDA09] Pierre Comon, Xavier Luciani, and André LF De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(7-8):393–405, 2009.

[CLS21] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

[CMDL+15] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh

Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163, 2015.

[CMLZ18] Huibin Chang, Stefano Marchesini, Yifei Lou, and Tieyong Zeng. Variational phase retrieval with globally convergent preconditioned proximal algorithm. *SIAM Journal on Imaging Sciences*, 11(1):56–93, 2018.

[Com93] Patrick L Combettes. The foundations of set theoretic estimation. *Proceedings of the IEEE*, 81(2):182–208, 1993.

[Com94] Pierre Comon. Independent component analysis, a new concept? *Signal Proc., Elsevier*, 36(3):287–314, 1994.

[Com00] P Comon. Tensor decompositions—state of the art and applications, keynote address in ima conf. *Mathematics in Signal Processing, Warwick, UK*, 375, 2000.

[Com14] Pierre Comon. Tensors: a brief introduction. *IEEE Sig. Proc. Magazine*, 31(3):44–53, 2014.

[Com21] P. Comon. Tenseurs en sciences des données. In *Encyclopédie des Techniques de l'Ingénieur*, volume Mathématiques pour l'Ingénieur, chapter AF114. Techniques de l'Ingénieur, Paris, October 2021.

[Con16] Laurent Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

[COV14] Luca Chiantini, Giorgio Ottaviani, and Nick Vannieuwenhoven. An algorithm for generic and low-rank specific identifiability of complex tensors. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1265–1287, 2014.

[CP11] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011. Ch.10.

[CPR16] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. A block coordinate variable metric forward–backward algorithm. *Journal of Global Optimization*, 66(3):457–485, 2016.

[CW05] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[CYM13] Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, 2013.

[CZ+97] Yair Censor, Stavros Andrea Zenios, et al. *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press on Demand, 1997.

[CZPA09] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[D+59] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[DDF+90] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[DK69] EA Dinic and MA Kronrod. An algorithm for the solution of the assignment problem. In *Soviet Math. Dokl*, volume 10, pages 1324–1326, 1969.

[DLC+95] L De Lathauwer, P. Comon, et al. Higher-order power method, application in Independent Component Analysis. In *NOLTA*, pages 91–96, Las Vegas, December 1995.

[DLDMV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[Don06] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[DP14] Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[DSL08] Vin De Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[Duc07] John Duchi. Properties of the trace and matrix derivatives. *Available electronically at https://web. stanford. edu/~ jduchi/projects/matrix_ prop. pdf*, 2007.

[FCC16] Rodrigo Cabral Farias, Jeremy Emile Cohen, and Pierre Comon. Exploring multimodal data fusion through joint de-

compositions with flexible couplings. *IEEE Transactions on Signal Processing*, 64(18):4830–4844, 2016.

[Fis25] R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(5):700–725, 1925.

[FIW+20] Xiao Fu, Shahana Ibrahim, Hoi-To Wai, Cheng Gao, and Kejun Huang. Block-randomized stochastic proximal gradient for low-rank tensor factorization. *IEEE Transactions on Signal Processing*, 68:2170–2185, 2020.

[Gal83] Zvi Galil. Efficient algorithms for finding maximal matching in graphs. In *Colloquium on Trees in Algebra and Programming*, pages 90–113. Springer, 1983.

[GT89] Harold N Gabow and Robert E Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[Hit27] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[HJ99] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1999.

[HL13] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[HLP14] Sung-Hsien Hsieh, Chun-Shien Lu, and Soo-Chang Pei. 2d sparse dictionary learning via tensor decomposition. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 492–496. IEEE, 2014.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[HSL16] Kejun Huang, Nicholas D Sidiropoulos, and Athanasios P Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065, 2016.

[Hua15] Kejun Huang. AO-ADMM-Code @ONLINE. `https://www.catalyzex.com/paper/arxiv:1506.04209#clicktoread/`, October 2015.

[JMC18] M. Jouni, M. Dalla Mura, and P. Comon. Some issues in computing the CP decomposition of nonnegative tensors. In *14th Int. Conf. on Latent Variable Analysis and Signal Separation (LVA-ICA)*, volume LNCS 10891, pages 57–66, Univ. of Surrey, UK, July 2-6 2018. Springer. hal-01784370.

[Joa98] Thorsten Joachims. Making large-scale svm learning practical. Technical report, Technical report, 1998.

[Jor04] Michael I Jordan. Graphical models. *Statistical science*, 19(1):140–155, 2004.

[KA21] Megha Khosla and Avishek Anand. Revisiting the auction algorithm for weighted bipartite perfect matchings. *arXiv preprint arXiv:2101.07155*, 2021.

[KBGP18] Nicolas Keriven, Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. Sketching for large-scale learning of mixture

models. *Information and Inference: A Journal of the IMA*, 7(3):447–508, 2018.

[KM11] Tamara G Kolda and Jackson R Mayo. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1095–1124, 2011.

[Kol01] Tamara G Kolda. Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255, 2001.

[Kow09] Matthieu Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[Lan95a] Ken Lang. 20 Newsgroups - Introduction @ONLINE. `http://qwone.com/~jason/20Newsgroups/`, 1995.

[Lan95b] Ken Lang. 20 Newsgroups - Raw data @ONLINE. `https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json`, 1995.

[LC09] L-H. Lim and P. Comon. Nonnegative approximations of nonnegative tensors. *Jour. Chemometrics*, 23:432–441, August 2009.

[LC14] L.-H. Lim and P. Comon. Blind multilinear identification. *IEEE Trans. Inf. Theory*, 60(2):1260–1280, February 2014. open access. hal-00763275.

[Lim05] L-H. Lim. Singular values and eigenvalues of tensors: a variational approach. In *IEEE SAM Workshop*, Puerto Vallarta, Mexico, 13-15 Dec. 2005.

[Lim13] L. H. Lim. Tensors and hypermatrices. In L. Hogben, editor, *Handbook of Linear Algebra*, chapter 15, pages 15.1–15.29. CRC Press, Boca Raton, FL, 2013. 2nd Ed.

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory*, pages 2140–2157. PMLR, 2019.

[Mar70] Bernard Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Recherche Opérationnelle*, 4:154–158, 1970.

[MBZJ08] Hosein Mohimani, Massoud Babaie-Zadeh, and Christian Jutten. A fast approach for overcomplete sparse decomposition based on smoothed ℓ_0 norm. *IEEE Transactions on Signal Processing*, 57(1):289–301, 2008.

[MM94] Eric Moreau and Odile Macchi. A one stage self-adaptive algorithm for source separation. In *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages III–49. IEEE, 1994.

[MOA79] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979.

[Mor62] Jean-Jacques Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus Acad. Sciences Paris Serie A Math.*, 255:2897–2899, 1962.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[Mur12] K. P. Murphy. *Machine Learning, A Probabilistic Perspective*. MIT press, London, 2012.

[Nig00] Kamal Nigam. 20 Newsgroups data sets @ONLINE. `http://www.cs.cmu.edu/~TextLearning/datasets.html`, February 2000.

[NMC20] Marouane Nazih, Khalid Minaoui, and Pierre Comon. Using the proximal gradient and the accelerated proximal gradient as a canonical polyadic tensor decomposition algorithms in difficult situations. *Signal Processing*, page 107472, 2020.

[NMSC21] Marouane Nazih, Khalid Minaoui, Elaheh Sobhani, and Pierre Comon. Computation of low-rank tensor approximation under existence constraint via a forward-backward algorithm. *Signal Processing*, page 108178, 2021.

[NMTM00] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.

[PB14] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[PC+19] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[Pea94] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

[Pra18] Selva Prabhakaran. 20 Newsgroups - Pre-processing steps @ONLINE. `https://www.machinelearningplus.`

`com/nlp/topic-modeling-gensim-python/#20topicdistributionacrossdocuments`, March 2018.

[PS11] Evangelos E Papalexakis and Nicholas D Sidiropoulos. Co-clustering as multilinear decomposition with sparse latent factors. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2064–2067. IEEE, 2011.

[QL17] L. Qi and Z. Luo. *Tensor Analysis, Spectral Theory and Special tensors*. SIAM, 2017.

[RCG18] Matteo Ruffini, Marta Casanellas, and Ricard Gavaldà. A new method of moments for latent variable models. *Machine Learning*, 107(8):1431–1455, 2018.

[RCM+16] Francesca Raimondi, Pierre Comon, Olivier Michel, Souleymen Sahnoun, and Agnes Helmstetter. Tensor decomposition exploiting diversity of propagation velocities: Application to localization of icequake events. *Signal Processing*, 118:75–88, 2016.

[RHL13] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[RMD11] Daniel Ramage, Christopher D Manning, and Susan Dumais. Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–465. ACM, 2011.

[RW09] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[SC08] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[SC15] S. Sahnoun and P. Comon. Joint source estimation and localization. *IEEE Trans. Sig. Proc.*, 63(10):2485–2495, May 2015.

[SCBZ19] E. Sobhani, P. Comon, and M. Babaie-Zadeh. Data mining with tensor decompositions. In *Gretsi*, Lille, August 26-29 2019.

[SCJBZ19] Elaheh Sobhani, Pierre Comon, Christian Jutten, and Massoud Babaie-Zadeh. Text mining with constrained tensor decomposition. In *International Conference on Machine Learning, Optimization, and Data Science (LOD)*, pages 219–231. Springer, 2019.

[SCJBZ22] Elaheh Sobhani, Pierre Comon, Christian Jutten, and Massoud Babaie-Zadeh. Corrindex: a permutation invariant performance index. *Signal Processing*, page 108457, 2022.

[SDLF+17] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[Ste09] Alwin Stegeman. Using the simultaneous generalized schur decomposition as a candecomp/parafac algorithm for ill-conditioned data. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(7-8):385–392, 2009.

[SWY75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[TK04] Petr Tichavsky and Zbynek Koldovsky. Optimal pairing of signal components separated by blind techniques. *IEEE Signal Processing Letters*, 11(2):119–122, 2004.

[Tse01] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[UC08] G. Upton and I. Cook. *A Dictionary of Statistics*. Oxford Paperback Reference. OUP Oxford, 2008.

[VCTM+17] Xuan Vu, Caroline Chaux, Nadège Thirion-Moreau, Sylvain Maire, and Elfrida Mihaela Carstea. A new penalized nonnegative third-order tensor decomposition using a block coordinate proximal gradient approach: Application to 3d fluorescence spectroscopy. *Journal of Chemometrics*, 31(4):e2859, 2017.

[VCTMM17] Xuan Vu, Caroline Chaux, Nadège Thirion-Moreau, and Sylvain Maire. A proximal approach for nonnegative tensor decomposition. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 201–210. Springer, 2017.

[Whi09] Joe Whittaker. *Graphical models in applied multivariate statistics*. Wiley Publishing, 2009.

[WYZ19] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

[XY13a] Yangyang Xu and Wotao Yin. APG-Code @ONLINE. `https://xu-yangyang.github.io/BCD/`, 2013.

[XY13b] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[YQ12] Wen-tau Yih and Vahed Qazvinian. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 616–620, 2012.

[YW82] Dan C Youla and Heywood Webb. Image restoration by the method of convex projections: Part 1-theory. *IEEE transactions on medical imaging*, 1(2):81–94, 1982.

[ZHPA13] James Y Zou, Daniel J Hsu, David C Parkes, and Ryan P Adams. Contrastive learning using spectral methods. *Advances in Neural Information Processing Systems*, 26:2238–2246, 2013.

[ZZZ+16] Yu Zhang, Guoxu Zhou, Qibin Zhao, Andrzej Cichocki, and Xingyu Wang. Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation. *Neurocomputing*, 198:148–154, 2016.