Aix*Marseille Université
École Doctorale en Mathématiques et Informatique - ED 184

Laboratoire d'Informatique Fondamentale
UMR 7279 - CNRS

Thèse présentée pour obtenir le grade universitaire de docteur

Spécialité : Informatique

A Tensor Perspective on Weighted Automata, Low-Rank Regression and Algebraic Mixtures

Guillaume RABUSSEAU

Soutenue le 20 Octobre 2016 devant le jury :

Marc TOMMASI	Université Lille 3	Rapporteur
Massimiliano PONTIL	Istituto Italiano di Tecnologia	Rapporteur
Borja De BALLE PIGEM	Lancaster University	Examinateur
Daniel KRESSNER	École Polytechnique Fédérale de Lausanne	Examinateur
Liva RALAIVOLA	Aix-Marseille Université	Examinateur
François DENIS	Aix-Marseille Université	Directeur
Hachem KADRI	Aix-Marseille Université	Co-Directeur

À Lisa et Joséphine

Résumé

La communauté de l'apprentissage automatique a récemment porté un intérêt grandissant aux tenseurs. Les tenseurs généralisent les vecteurs et les matrices aux ordres supérieurs : un tenseur d'ordre 3 peut être vu comme un tableau tridimensionnel de scalaires. D'un point de vue algébrique, à l'instar des matrices qui représentent des applications linéaires, les tenseurs représentent des applications multilinéaires.

La connexion la plus évidente entre les tenseurs et l'apprentissage automatique apparaît lorsque les données ont une structure tensorielle : une image couleur peut être représentée par un tenseur d'ordre 3 (de dimensions largeur, hauteur et nombre de canaux), tout comme des données d'électroencéphalographie ou encore des séries temporelles multivariées. Plus fondamentalement, les tenseurs peuvent intervenir en tant que paramètres de modèles utilisés en apprentissage automatique, comme c'est le cas pour les séries reconnaissables d'arbres, ou encore comme outils algébriques servant à élaborer des algorithmes d'apprentissage.

Ce manuscrit regroupe différents travaux explorant ces interactions entre les tenseurs et l'apprentissage automatique. Le premier chapitre est consacré à l'extension des modèles de séries reconnaissables de chaînes et d'arbres aux graphes. Nous y montrons que les modèles d'automates pondérés de chaînes et d'arbres peuvent être interprétés d'une manière simple et unifiée à l'aide de réseaux de tenseurs, et que cette interprétation s'étend naturellement aux graphes ; nous étudions ensuite certaines propriétés fondamentales de ce modèle et présentons des résultats préliminaires sur leur apprentissage. Le second chapitre porte sur la minimisation approximée d'automates pondérés d'arbres et propose une approche théoriquement fondée à la problématique suivante : étant donné un automate pondéré d'arbres à n états, comment trouver un automate à $m < n$ états calculant une fonction proche de l'originale. Ces travaux généralisent des résultats récents sur la minimisation approximée d'automates pondérés de chaînes. Le troisième chapitre traite de la régression de faible rang pour sorties à structure tensorielle. Nous y proposons un algorithme d'apprentissage rapide et efficace pour traiter un problème de régression dans lequel les entrées sont des vecteurs et les sorties des tenseurs. Nous montrons que l'algorithme proposé est un algorithme d'approximation pour ce problème NP-difficile et nous donnons une analyse théorique des propriétés statistiques et de généralisation de cet algorithme. Enfin, le quatrième chapitre introduit le modèle de mélanges algébriques de distributions. Ce modèle considère des combinaisons affines de distributions (où les coefficients somment à un mais ne sont pas nécessairement positifs). Ces mélanges apparaissent naturellement dans le contexte des séries reconnaissables de chaînes et permettent de modéliser des phénomènes de bruitage d'observation (e.g. biais d'échantillonnage). Nous proposons une approche pour l'apprentissage de mélanges algébriques qui étend la méthode tensorielle des moments introduite récemment. Cette approche nous permet en particulier d'obtenir un algorithme d'apprentissage pour les mélanges algébriques de Gaussiennes sphériques.

Abstract

Recently, there has been an increasing interest from the machine learning community towards *tensors*. Tensors are a generalization of vectors and matrices to higher orders: a 3rd order tensor can be seen as a 3-dimensional array of scalars. From an algebraic perspective, matrices represent linear maps while tensors represent *multilinear maps*.

The more obvious connection between machine learning and tensors arises when the data is naturally structured as a tensor: color images can for example be seen as 3rd order tensors (of dimensions width, height and number of channels), which is also the case for multivariate time series (e.g. electroencephalogram signals). Tensors can also appear in a more fundamental way as parameters of machine learning models, which is the case for weighted tree automata, or as efficient tools used to design learning schemes.

This thesis tackles several problems exploring these connections between tensors and machine learning. In the first chapter, we propose an extension of the classical notion of recognizable function on strings and trees to graphs. We first show that the computations of weighted automata on strings and trees can be interpreted in a natural and unifying way using tensor networks, which naturally leads us to define a computational model on graphs: graph weighted models; we then study fundamental properties of this model and we present preliminary results on learning graph weighted models on the simple family of circular strings. The second chapter tackles a model reduction problem for weighted tree automata. We extend recent results on approximate minimization of weighted string automata and we propose a principled approach to the following problem: given a weighted tree automaton with n states, how can we find an automaton with $m < n$ states that is a good approximation of the original automaton? The motivation behind this problem is to reduce the complexity of algorithms working with weighted tree automata at the price of incurring a controlled amount of error in the output of such algorithms. In the third chapter, we consider a problem of low rank regression for tensor structured outputs. We design a fast and efficient algorithm to address a regression task where the inputs are vectors and the outputs are tensors. We show that this algorithm generalizes the reduced rank regression method and that it offers good approximation, statistical and generalization guarantees. Lastly in the fourth chapter, we introduce the algebraic mixture model. This model considers affine combinations of probability distributions (where the weights sum to one but may be negative). We show that algebraic mixtures naturally appear in a fundamental relation between probabilistic and weighted automata, and that they could prove useful to model missing data scenario such as selection bias. We extend the recently proposed tensor method of moments to algebraic mixtures, which allows us in particular to design a learning algorithm for algebraic mixtures of spherical Gaussian distributions.

Acknowledgments

I would like to thank all the people that supported and encouraged me during my studies and allowed me to arrive to this day where I defend my PhD thesis.

First and foremost I would like to thank my two PhD advisors, François Denis and Hachem Kadri, for their guidance, precious advices, support and kindness. Thanks to them, these three years of PhD studies have been a really wonderful journey. I also want to thank my coauthors Raphaël Bailly, Borja Balle and Shay B. Cohen. I am especially grateful to Borja for all the fruitful discussions we had and the help and advices he gave me. I also thank the members of my PhD committee who kindly accepted to review the works presented in this manuscript.

I would like to thank Liva whose machine learning courses in M2IF definitely were a cornerstone of my interest for machine learning. More generally I am grateful to all the teachers of the M2IF for transmitting their passion for research, and to the CTES for making it possible for me to go back to university with distance learning. I also would like to thank all the members of the Qarma team, it has been a pleasure working, discussing research and life, playing *pétanque* and partying with all of you during these years. A special thought goes to my old desk neighbors: Sokol with whom I had so many enjoyable discussions and Julien who was always available to answer my math questions and to share his enthusiasm for research. I also thank the signal processing team & co. (Denis, Dominique, Alex, ...) for making the CMI's corridor such a nice work place, and the LIF's angels (Sylvie, Nadine and Martine), Manu and Kai, and the *École Doctorale* 184 for always being so helpful.

Last but not least I am forever grateful to Lisa for always being so supportive and comprehensive, from the beginning of this strange idea of going back to school in 2008, until these three years of PhD studies with their ups and downs and stressful deadlines; and to my daughter Joséphine who took long enough naps for me to write parts of this manuscript while rocking her baby chair...

Contents

List of Figures

List of Tables

Introduction

Over the past decades *machine learning* has become a prominent subfield of computer sciences that is developing at an impressive rate. Quoting Arthur Samuel[a], machine learning is a "field of study that gives computers the ability to learn without being explicitly programmed". As an illustration, suppose that we want to write an algorithm able to distinguish images of the digit 8 from images of the digit 9. One way to proceed would be to first use image processing tools to count the number of loops in the image, and then classify the image as a 9 if there is only one loop and as an 8 if there are two. This approach directly encodes prior knowledge into the classification algorithm. Machine learning takes a different road and aims at designing algorithms capable of inferring discriminant features from a set of labeled examples: given a set of images of 8's and 9's the *learner* (i.e. the learning algorithm) will try to find rules to distinguish images from the two classes and will then be able to use these rules to classify unseen images of digits as representing an 8 or a 9. Machine learning is now a very rich and diverse discipline that is rapidly taking a considerable importance in our everyday life, from automatically detecting spam messages in our mail in-boxes to cars driving by themselves, from targeting advertisements according to user's profiles to predicting heart failures, and from suggesting which movie we should watch this evening to helping reduce wait times in emergency rooms.

Roughly speaking, machine learning tasks can be divided into two main categories: *supervised and unsupervised learning* tasks. In a supervised learning setting the learner is given a set of input/output pairs $\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$ (where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces respectively) from which it tries to infer a function $f : \mathcal{X} \to \mathcal{Y}$ mapping inputs to outputs. One example of such a task is the classification example described above where $\mathcal{X}$ is the set of images of the digits 8 and 9 and $\mathcal{Y} = \{-1, +1\}$ with $f(x) = -1$ if x is an image of the digit 8 and $f(x) = +1$ otherwise. In an unsupervised setting the learner is given a sample of unlabeled points $\{x_1, \cdots, x_N\} \subset \mathcal{X}$ from which it tries to reveal some hidden structure. One example of an unsupervised task is clustering where the learner has to identify groups of points that are similar to each other. Going back to the digits examples, even if the learner does not know which images represent an 8 and which images represent a 9, it could still be able to determine that there are two main groups in the training sample (e.g. images with one loop and the ones with two loops).

The input examples of a machine learning algorithm are often given as a set of points in a vector space $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^d$. Consequently, a wide range of machine learning algorithms deeply relies on *linear algebra* tools. Linear algebra is the branch of math-

[a] Arthur Samuel (December 5, 1901 – July 29, 1990) was an American pioneer in artificial intelligence and machine learning. Its checkers playing program (Samuel, 1959) appears to be the first self-learning program.

ematics that studies vector spaces and linear mappings between these spaces (Lang, 1987). For example, in the digit *classification* example the images can be represented as points in a d-dimensional vector space. The learner can then try to find a vector $\mathbf{w} \in \mathbb{R}^d$ such that the inner product $\langle \mathbf{w}, \mathbf{x}_n \rangle$ is negative if $\mathbf{x}_n$ is an image of an 8 and positive otherwise. A related task is the one of *regression*, where the outputs of the function to be learned are real values instead of classes (i.e. the output space $\mathcal{Y}$ is continuous instead of discrete). One example of such a task would be to predict the selling price of a house given its characteristics (e.g. number of rooms, surface, neighborhood, etc.). For a regression task, where the learner has to infer a real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ from a training sample $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^d \times \mathbb{R}$, one strategy consists in finding a regression vector $\mathbf{w} \in \mathbb{R}^d$ such that $\langle \mathbf{w}, \mathbf{x}_n \rangle$ is close to y_n for all n. If by close we mean that $(\langle \mathbf{w}, \mathbf{x}_n \rangle - y_n)^2$ is as small as possible throughout the whole training sample, finding the vector $\mathbf{w}$ boils down to solving a linear system of equations which can be done using simple linear algebra operations such as matrix products and inversions. More elaborate tools from linear algebra are also commonly used in machine learning. There is for example a great variety of machine learning methods gathered under the term *spectral methods*. These methods have in common the fact that they leverage spectral properties of some matrices (i.e. properties of eigenvectors and eigenvalues of the matrix). Principal Component Analysis (PCA) is a famous example where one wants to project a set of points onto a low-dimensional subspace (see e.g. Jolliffe, 2002). PCA consists in projecting the set of points onto the axes of largest possible variance and boils down to a singular value decomposition of the covariance matrix.

In the recent years, the machine learning community has shown an increasing interest towards *tensors and multilinear algebra*. Tensors are a generalization of vectors and matrices to higher orders: while a vector can be seen as a 1-dimensional array and a matrix as a 2-dimensional array, higher order tensors can be seen as multi-dimensional arrays. A vector is a tensor of order 1, a matrix is a tensor of order 2, and a 3rd order tensor can simply be seen as a 3-dimensional array of numbers. While linear algebra focuses on the study of linear maps — mappings between vectors spaces satisfying $f(\alpha\mathbf{x} + \beta\mathbf{x}') = \alpha f(\mathbf{x}) + \beta f(\mathbf{x}')$ for any scalar α, β and vectors $\mathbf{x}, \mathbf{x}'$ — multilinear algebra studies multilinear maps, i.e. mappings going from a cartesian product of vector spaces $V_1 \times V_2 \times \cdots \times V_p$ to an output vector space W that are linear separately in each variable:

$$f(\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \alpha\mathbf{x}_i + \beta\mathbf{x}'_i, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_p) = \alpha f(\mathbf{x}_1, \cdots, \mathbf{x}_p) + \beta f(\mathbf{x}_1, \cdots, \mathbf{x}'_i, \cdots, \mathbf{x}_p)$$

for any scalars α, β, vectors $\mathbf{x}_1 \in V_1, \cdots, \mathbf{x}_p \in V_p, \mathbf{x}'_i \in V_i$ and any $1 \leq i \leq p$. A fundamental result in linear algebra states that any linear map f between two real vector spaces of dimensions d and p respectively is uniquely determined by a matrix $\mathbf{M} \in \mathbb{R}^{d \times p}$ once bases have been chosen for the input and output spaces. Similarly, a multilinear map $f : V_1 \times \cdots \times V_p \to W$ is uniquely determined by a tensor $\boldsymbol{\mathcal{T}}$ of order $p + 1$ once bases have been chosen for the vector spaces $V_1, \cdots, V_p$ and W.

Tensors and multilinear algebra are powerful tools that have been successfully used in machine learning in the last decade. The works presented in this thesis are part of this emerging effort to **bring the power of multilinear algebra and tensors to machine learning**. The relevance of tensors to machine learning can be seen from different perspectives.

(1) Tensor structured data. First, the underlying object on which machine learning reasons is data and it may happen that the data has a natural tensor structure. For example a color image encoded with three color channels can naturally be interpreted as a 3rd order tensor of size $width \times height \times 3$, similarly a video can be seen as a 4th order tensor where the 4th mode corresponds to the time dimension. Another example of data having a natural tensor structure arises in the task of spatio-temporal forecasting where a set of variables are observed in different locations across several time steps: the learning data could consist in meteorological variables (such as rain level, temperatures and rate of sunshine) that are observed in several cities across a country and for which we have access to monthly measurements. This data can naturally be organized as a 3rd order tensor where the different modes correspond to variables, cities and timestamps respectively (see Chapter 4).

(2) Tensors as parameters of a model. Second, tensors may arise as parameters or components of machine learning models. For example the factorization machines introduced in (Rendle, 2010) consider a non-linear model to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ that takes into account polynomial features of the input $\mathbf{x} \in \mathbb{R}^d$, e.g. $f(\mathbf{x}) = \sum_i \alpha_i \mathbf{x}_i + \sum_{i,j} \beta_{i,j} \mathbf{x}_i \mathbf{x}_j + \sum_{i,j,k} \gamma_{i,j,k} \mathbf{x}_i \mathbf{x}_j \mathbf{x}_k$. The parameters of the model (the coefficients α_i, $\beta_{i,j}$, $\gamma_{i,j,k}$ in the previous example) can be seen as components of a tensor, and tensor factorization techniques are used to reduce the number of parameters of the model and to introduce some regularization. The same idea was successfully applied in the context of multi-view learning where the learner has access to several representations of the input examples. If the inputs are images the views could consist in different descriptors such as SIFT, histograms, etc. In (Cao et al., 2016) the authors propose to take into account multiplicative interactions between the different views using a model closely related to factorization machines.

Another example of models parameterized by tensors are weighted tree automata. Classical automata on words and trees are finite state machines that run on some input (a word or a tree) in discrete time steps and can either accept or reject the input. Rather than simply accept or reject an input, a weighted automaton will take as input a string or a tree and will output a value in some semi-ring, e.g. a real number. Thus weighted automata are computational devices that modelize functions whose domain is a set of structured objects (e.g. strings or trees). In particular, weighted automata can compute probability distributions on strings or trees and therefore are relevant to machine learning. Weighted tree automata are traditionally defined by means of multilinear maps (Berstel and Reutenauer, 1982), henceforth the parameters of a weighted tree automaton can be thought of as tensors (see Chapters 2 and 3).

(3) Tensors as tools. Third, tensors may simply arise as efficient tools to design learning schemes. This is the case for the tensor method of moments proposed in (Anandkumar et al., 2014). This method is a special case of the general method of moments used to infer the parameters of a parametric distribution from a sample of

training data drawn from this distribution. The connection with tensors comes from the fact that the pth order moment of a vector random variable is a pth order tensor. The tensor method of moments relies on deriving polynomial equations relating the higher-order moments of a random variable with the parameters of the distribution, and solving the resulting system of equations using tensor decomposition techniques (see Chapter 5). Reducing a learning problem to finding a decomposition of an observable tensor has also been investigated for blind source decomposition and independent component analysis in the signal processing community (see e.g. Cardoso, 1990; Cichocki et al., 2009b; Cichocki et al., 2015). Tensor decomposition techniques are also recently getting attention from the neural network community where it has been shown that several tensor decomposition techniques can be used to compress the weight matrices of hidden layers, and lead to a considerable speedup of computations at inference time (see e.g. Novikov et al., 2015; Lebedev et al., 2014). Similarly, using tensor decomposition techniques to speedup parsing with a probabilistic context-free grammar has been explored in (Cohen et al., 2013; Collins and Cohen, 2012).

In this thesis we will tackle several problems illustrating these connections between machine learning and tensors. The first two are related to weighted automata and involve models that are parameterized by a set of tensors: we will propose an extension of the classical notion of recognizable string and tree series to graphs, and a principled approach for approximate minimization of weighted tree automata. The third problem concerns a regression task where the output data has a natural tensor structure and will be addressed using tensor decomposition techniques. Lastly, we will study the problem of learning algebraic mixtures (i.e. affine combinations of probability distributions) and derive an extension of the tensor method of moments to this setting.

Outline of the thesis.

- Chapter 1 is devoted to the introduction of the fundamental objects that are at the core of this thesis: *tensors*. We will introduce our notations, common notions and operations on tensors, and tensor decomposition techniques. We will also introduce *tensor network diagrams* that allow one to represent computation on tensors in a graphical and intuitive way: a tensor network is a graph where the nodes are tensors and where edges represent contractions between the tensors in the graph.

- In Chapter 2, we first show how computations of a weighted automaton on strings or trees can be interpreted as a simple mapping between the input (a string or a tree) and a tensor network reflecting its structure. This interpretation suggests a natural extension of weighted automata to labeled graphs and leads us to define the novel notion of *Graph Weighted Models* (GWM). A major part of this chapter is dedicated to the study of theoretical properties of GWMs. More precisely, we study on which conditions properties that are desirable for a model extending weighted automata on strings and trees are satisfied: is the set of functions that can be computed by a GWM closed under sum, Hadamard (pointwise) product and scalar multiplication? Is a GWM able to compute any finite support functions (i.e. a function taking non-zero values only on a finite set of graphs)?

This work is motivated by our aspiration to extend existing spectral learning methods for weighted automata on strings and trees to functions defined over graphs. The last part of this chapter presents preliminary results in this direction by considering a learning problem for GWMs defined over a very simple family of graphs: circular strings. Roughly speaking, circular strings are strings that are closed onto themselves and they are in some sense the simplest family of graphs with loops. After showing that a function computed by a GWM on circular strings can be learned using learning algorithms for traditional weighted automata on strings, we propose an alternative approach that relies on tensor decomposition techniques and seems more promising to be extended to the problem of learning GWMs defined over larger families of graphs.

- Chapter 3 tackles a model reduction problem for weighted tree automata (WTA) and it extends the results presented in (Balle, Panangaden, and Precup, 2015) for weighted automata on strings to the tree case. Without diving into the details, a WTA can be parameterized by a set of tensors whose common dimension is called the number of states. For example a WTA with n states computing a function on (unlabeled) binary trees consists of two vectors $\boldsymbol{\alpha}, \boldsymbol{\omega} \in \mathbb{R}^n$ and a 3rd order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times n \times n}$. The problem we consider is the following: given a WTA with n states and a target number of states $\hat{n} < n$, how can we find a WTA with $\hat{n}$ states that is a good approximation of the original automaton? The motivation behind this problem is to reduce the complexity of algorithms working with WTA at the price of incurring a small, controlled amount of error in the output of such algorithms. We first introduce a canonical form for weighted tree automata where each state of the automaton corresponds to a singular value of the *infinite* Hankel matrix associated with the WTA. Once a WTA has been transformed into this canonical form, the states associated with the smallest singular values are removed to obtain the minimized model. The transformation of a WTA into this canonical form relies on finding the singular value decomposition of the infinite Hankel matrix. Our main result is an efficient algorithm for computing this singular value decomposition by operating directly on the WTA representation of the Hankel matrix; that is, without the need to explicitly represent this infinite matrix at any point. The idea of speeding up parsing with probabilistic context free grammar by approximating the original model with a smaller one was recently studied in (Collins and Cohen, 2012; Cohen, Satta, and Collins, 2013), where a tensor decomposition technique was used in order to obtain the minimized model. We compare that approach to ours on experiments where both techniques are used to compute approximations to a grammar learned from a corpus of real linguistic data.

- In Chapter 4 we consider a regression problem where the outputs are tensors. One example of such a problem is the spatio-temporal forecasting task described above where the outputs of the function f we want to learn are the values of different meteorological variables in different cities (f would be a matrix-valued function in this case). The method we propose is a generalization of the *reduced rank regression* method where a linear function $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$ is learned from a training sample of input/output pairs by minimizing a least-square criterion subject to a low-rank constraint; that is $f : \mathbf{x} \mapsto \mathbf{W}^\top \mathbf{x}$ where $\mathbf{W} \in \mathbb{R}^{d_0 \times d_1}$ is of low rank.

In the case where the outputs are higher order tensors we propose to replace the rank constraint on the regression matrix $\mathbf{W}$ by a multilinear rank constraint on a regression tensor $\mathcal{W}$. The resulting optimization problem is NP-hard but we manage to design an efficient approximation algorithm (HOLRR) for this problem. After showing that HOLRR can easily be extended to the non-linear setting using the so called kernel trick, we provide a theoretical analysis of HOLRR: we show that HOLRR is computationally efficient and has good approximation guarantees, we show that it generalizes reduced rank regression and that it is consistent, and we provide a generalization bound for the class of tensor-valued regression functions with low multilinear rank. Experiments on both synthetic and real world data show that HOLRR outperforms multivariate and multilinear regression methods and can be considerably faster than existing tensor methods.

- Finally, Chapter 5 proposes an extension of the tensor method of moments proposed in (Anandkumar et al., 2014) to *algebraic mixture models*. Traditional mixture models are probability distributions that can be expressed as a mixture of distributions: the probability density function of a mixture is given by $f = w_1 f_1 + w_2 f_2 + \cdots + w_k f_k$ where each f_i is a probability density function and the weights w_i satisfy $0 \leq w_i \leq 1$ and $w_1 + w_2 + \cdots + w_k = 1$. It may happen that the function f remains positive even if some of the weights w_i are negative, in which case f still defines a probability density function. We call such distributions *algebraic mixtures*. In this chapter, we show how algebraic mixtures naturally appear in a fundamental relation between weighted and probabilistic automata. We also provide some intuition on algebraic mixtures by showing that generating an example from such a distribution can be interpreted as sampling an example from the positive components of the mixture and accepting or rejecting it with a probability that depends on the negative components of the mixture. Thus a sample drawn from an algebraic mixture can be seen as a sample drawn from the positive components of the mixture where some of the data is missing. Using algebraic mixtures of Gaussian distributions as an illustration, we propose an extension of the tensor method of moments to this setting that we evaluate in a simulation study.

List of Publications

- In peer-reviewed international conferences:
  - Guillaume Rabusseau and Hachem Kadri. *Low-Rank Regression with Tensor Responses*. In Advances in Neural Information Processing Systems, NIPS 2016.
  - Guillaume Rabusseau, Borja Balle, and Shay B. Cohen. *Low-Rank Approximation of Weighted Tree Automata*. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, pages 839-847, 2016.
  - Raphaël Bailly, François Denis, and Guillaume Rabusseau. *Recognizable Series on Hypergraphs*. In Proceedings of the 9th International Conference on Language and Automata Theory and Applications, LATA 2015, Nice, France, March 2-6, 2015, pages 639-651, 2015. **Journal version under review for a special issue of the Journal of Computer and System Sciences.**
  - Guillaume Rabusseau and François Denis. *Maximizing a Tree Series in the Representation Space*. In Proceedings of the 12th International Conference on Grammatical Inference, ICGI 2014, Kyoto, Japan, September 17-19, 2014, pages 124-138, 2014.

- In peer-reviewed french conferences:
  - Guillaume Rabusseau. *Régression de faible rang pour réponses tensorielles*. Conférence sur l'Apprentissage Automatique, CAP 2016, Marseille, France, July 5-7, 10 pages, 2016.
  - Guillaume Rabusseau, Borja Balle, and Shay B. Cohen. *Minimisation approximée d'automates pondérés d'arbres*. Conférence sur l'Apprentissage Automatique, CAP 2016, Marseille, France, July 5-7, 10 pages, 2016.
  - Guillaume Rabusseau, Hachem Kadri, and François Denis. *Régression de faible rang non-paramétrique pour réponses tensorielles*. Colloque International Francophone de Traitement du Signal et de l'Image, GRETSI 2015, Lyon, France, September 8-11, 4 pages, 2015.
  - Guillaume Rabusseau and François Denis. *Décompositions tensorielles pour l'apprentissage de modèles de mélanges négatifs*. Conférence sur l'Apprentissage Automatique, CAP 2014, Saint-Etienne, France, July 4-7, 10 pages, 2014. **Best student paper award.**
- Workshop contributions:
  - Guillaume Rabusseau and François Denis. *Learning Negative Mixture Models by Tensor Decompositions*. Workshop on Method of Moments and Spectral Learning (ICML 2014), Beijing, China, June 25, 2014.

Notations

$[p]$	set of integers from 1 to p
$\lvert S\rvert,\ \lvert\alpha\rvert$	cardinal of the set S, absolute value/modulus of the scalar α
δ_{ij}	Kronecker symbol: equals 1 if $i = j$ and 0 otherwise
$\alpha, \mathbf{v}, \mathbf{M}, \boldsymbol{\mathcal{T}}$	scalar, vector, matrix, tensor
$\mathbf{I}_n$	$n \times n$ identity matrix
$\mathbf{M}^+$	Moore-Penrose pseudoinverse of the matrix $\mathbf{M}$
$det(\mathbf{M})$	determinant of the matrix $\mathbf{M}$
$\mathrm{Tr}(\mathbf{M})$	trace of the matrix $\mathbf{M}$
$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle$	inner product between vectors, matrices or tensors
$\lVert \cdot \rVert_F$	Frobenius norm
$span(\{\mathbf{v}_1, \cdots, \mathbf{v}_k\})$	linear span of a set of vectors (i.e. $\{\sum_{i=1}^{k} \alpha_i \mathbf{v}_i\}$)
$V^{\otimes k}$	kth order tensor product of the vector space V
$\mathrm{vec}(\mathbf{M}), \mathrm{vec}(\boldsymbol{\mathcal{T}})$	vectorization of a matrix/tensor
$\mathbf{M}_{i,:},\ \mathbf{M}_{:,j},\ \boldsymbol{\mathcal{T}}_{k,:,:}$	ith row of $\mathbf{M}$, jth column of $\mathbf{M}$, kth mode-1 slice of $\boldsymbol{\mathcal{T}}$
$\mathbf{T}_{(k)}$	mode-k matricization of the tensor $\boldsymbol{\mathcal{T}}$
$\mathbf{x} \circ \mathbf{y}$	outer product between vectors, matrices or tensors
$\mathbf{x}^{\circ k}$	kth tensor power $\mathbf{x}^{\circ k} = \mathbf{x} \circ \mathbf{x} \circ \cdots \circ \mathbf{x}$ (k times)
$\mathbf{x} \otimes \mathbf{y}$	Kronecker product (for vectors, matrices and higher-order tensors)
$\mathbf{x}^{\otimes k}$	kth Kronecker power $\mathbf{x}^{\otimes k} = \mathbf{x} \otimes \mathbf{x} \otimes \cdots \otimes \mathbf{x}$ (k times)
$\boldsymbol{\mathcal{X}} \oplus \boldsymbol{\mathcal{Y}}$	direct sum of two vectors/matrices/tensors
$\boldsymbol{\mathcal{T}} \times_k \mathbf{M}$	mode-k matrix product
$\boldsymbol{\mathcal{T}} \bullet_k \mathbf{v}$	mode-k vector product
$\boldsymbol{\mathcal{T}}(\mathbf{X}_1, \cdots, \mathbf{X}_p)$	tensor as a multilinear map (equals $\boldsymbol{\mathcal{T}} \times_1 \mathbf{X}_1^\top \times_2 \cdots \times_p \mathbf{X}_p^\top$)
$\langle\!\langle \boldsymbol{\mathcal{T}} \rangle\!\rangle_{(i,j)}$	contraction of the ith and jth mode of the tensor $\boldsymbol{\mathcal{T}}$
$\mathbb{P}[\,\cdot\,]$	probability of an event
$\mathbb{E}[\,\cdot\,]$	expectation of a random variable
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	PDF of a multivariate normal with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\Sigma, \Sigma^*, \Sigma^+$	finite alphabet/set of strings on Σ/set of non-empty strings on Σ
$\mathcal{F} = (\Sigma, \sharp)$	ranked alphabet (with arity function $\sharp$)
$\mathfrak{T}_\mathcal{F}, \mathfrak{C}_\mathcal{F}$	set of trees/contexts on the ranked alphabet $\mathcal{F}$
$\mathfrak{G}_\mathcal{F},\ \mathfrak{G}_\mathcal{F}^k$	set of closed graphs/graphs with k free ports on $\mathcal{F}$
$G_1 \cup G_2$	disjoint union of two graphs
$\langle\!\langle G \rangle\!\rangle_{(p,p')}$	graph obtained by connecting the free ports p and p' in the graph G

1 Preliminaries

Contents

In this chapter, we introduce the objects that are at the core of this thesis: *tensors*.

1.1 Notations

We first introduce some notations that we will use throughout the thesis. For convenience, a table summarizing the notations that we frequently use is available on page 8 and an index is given on page 160.

For any set S we denote by $|S|$ its cardinality. For any integer n, the set of integer from 1 to n is denoted by $[n] = \{1, 2, \cdots, n\}$. We use lower case bold letters (or symbols) for vectors (e.g. $\mathbf{v} \in \mathbb{R}^{d_1}$), upper case bold letters for matrices (e.g. $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$) and bold calligraphic letters for higher order tensors (e.g. $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$). The transpose of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^\top$. Given $i_1 \in [d_1], i_2 \in [d_2], i_3 \in [d_3]$ we use $\mathbf{v}_{i_1}$, $\mathbf{M}_{i_1,i_2}$ and $\boldsymbol{\mathcal{T}}_{i_1,i_2,i_3}$ to denote the corresponding entries (we will sometimes use the notation $\mathbf{M}(i_1, i_2) = \mathbf{M}_{i_1,i_2}$ for convenience). The ith row (resp. column) of a matrix $\mathbf{M}$ will be denoted by $\mathbf{M}_{i,:}$ (resp. $\mathbf{M}_{:,i}$) and this notation will be extended to tensors in the straightforward way. The $n \times n$ identity matrix will be written $\mathbf{I}_n$ (or simply $\mathbf{I}$ if the dimension is clear from context). Given a matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ we use $\mathrm{vec}(\mathbf{M}) \in \mathbb{R}^{d_1 d_2}$ to denote the column vector obtained by concatenating the columns of $\mathbf{M}$.

The trace of a square matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ is the sum of its diagonal elements: $\mathrm{Tr}(\mathbf{M}) = \sum_{i=1}^{d} \mathbf{M}_{i,i}$. The inner product between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ is defined by $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{d} \mathbf{u}_i \mathbf{v}_i$. Similarly the inner product between two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d_1 \times d_2}$ is defined by $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}^\top \mathbf{B}) = \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \mathbf{A}_{i,j} \mathbf{B}_{i,j}$. The Frobenius norm of a vector (resp. of a matrix) is denoted by $\|\mathbf{v}\|_F = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$ (resp. $\|\mathbf{A}\|_F = \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle}$). The rank of a matrix $\mathbf{M}$ will be denoted by $\mathrm{rank}(\mathbf{M})$, its inverse by $\mathbf{M}^{-1}$, and its Moore-Penrose pseudo-inverse by $\mathbf{M}^{+}$.

1.2 Tensors, Tensor Algebra and Tensor Networks

We now introduce basic notions on tensors and multilinear algebra. After formally defining what is a tensor, we introduce the notion of tensor networks that will help us visualize the various tensor operations and decompositions presented in this section. Most of the notations we use have been introduced in (Kolda and Bader, 2009) where more details can be found. We also refer the reader to the surveys (Grasedyck, Kressner, and Tobler, 2013) and (Sidiropoulos et al., 2016) and to the monograph (Hackbusch, 2012).

1.2.1 Tensors

Let $\mathbb{F}$ be a field, in this thesis we only consider the case where $\mathbb{F}$ is the field of real numbers $\mathbb{R}$ or the field of complex numbers $\mathbb{C}$. For any integers $p, d_1, \cdots, d_p$, a *pth order tensor* $\boldsymbol{\mathcal{T}} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ is a p-dimensional array of scalars

$$(\boldsymbol{\mathcal{T}}_{i_1,\cdots,i_p} \in \mathbb{F} \ : \ i_n \in [d_n], n \in [p]).$$

A fundamental operation used to construct tensors is the *outer product* (or *tensor product*) between vectors $\mathbf{v}_1 \in \mathbb{F}^{d_1}, \cdots, \mathbf{v}_p \in \mathbb{F}^{d_p}$. The outer product $\mathbf{v}_1 \circ \mathbf{v}_2 \circ \cdots \circ \mathbf{v}_p \in \mathbb{F}^{d_1 \times d_2 \times \cdots \times d_p}$ is the pth order tensor defined by

$$(\mathbf{v}_1 \circ \mathbf{v}_2 \circ \cdots \circ \mathbf{v}_p)_{i_1,i_2,\cdots,i_p} = (\mathbf{v}_1)_{i_1}(\mathbf{v}_2)_{i_2} \cdots (\mathbf{v}_p)_{i_p}$$

for all indices $i_1 \in [d_1], i_2 \in [d_2], \cdots, i_p \in [d_p]$. In particular when $p = 2$, we have $\mathbf{x} \circ \mathbf{y} = \mathbf{x}\mathbf{y}^\top$ for any vectors $\mathbf{x}$ and $\mathbf{y}$. A tensor can be defined in a more abstract way as an element of the tensor product space $\mathbb{F}^{d_1} \otimes \mathbb{F}^{d_2} \otimes \cdots \otimes \mathbb{F}^{d_p}$, which is the vector space consisting of all linear combinations $\sum_i \alpha_i (\mathbf{v}_1^i \circ \mathbf{v}_2^i \circ \cdots \circ \mathbf{v}_p^i)$ where $\mathbf{v}_n^i \in \mathbb{F}^{d_n}$ for all $n \in [p]$. However it will be sufficient in this thesis to simply consider tensors as multidimensional arrays.

We say that a tensor $\boldsymbol{\mathcal{T}}$ is *(hyper)cubic* if all its dimensions $d_1, d_2, \cdots, d_p$ are equal, in which case we will write $\boldsymbol{\mathcal{T}} \in (\mathbb{F}^d)^{\otimes p}$ where $d = d_1 = \cdots = d_p$. A cubic tensor is *symmetric* if it is invariant under permutation of its indices, that is $\boldsymbol{\mathcal{T}}_{i_1,\cdots,i_p} = \boldsymbol{\mathcal{T}}_{i_{\sigma(1)},\cdots,i_{\sigma(p)}}$ for any permutation $\sigma : [p] \to [p]$ and all indices $i_1, \cdots, i_p \in [d]$.

The different axes (i.e. indices or dimensions) of a tensor will be referred to as *modes*. The *mode-n* fibers of a tensor $\boldsymbol{\mathcal{T}}$ are the vectors obtained by fixing all indices except for the nth one. For example $\boldsymbol{\mathcal{T}}_{:,i_2,\cdots,i_p}$ is a vector of dimension d_1, and for a 2nd order tensor $\mathbf{M}$ (i.e. a matrix) the mode-1 fibers correspond to columns and the mode-2 fibers to rows. The *nth mode matricization* of $\boldsymbol{\mathcal{T}}$ is the matrix having the mode-n fibers of $\boldsymbol{\mathcal{T}}$ for columns and is denoted by $\mathbf{T}_{(n)} \in \mathbb{R}^{d_n \times d_1 \cdots d_{n-1} d_{n+1} \cdots d_p}$. Observe that we use the same letter to denote that $\mathbf{T}_{(n)}$ is the matricization of the tensor $\boldsymbol{\mathcal{T}}$; we will sometimes also write $(\boldsymbol{\mathcal{T}})_{(n)}$. In a matricization the mode-n fibers are organized using the inverse lexicographical order: for example if $\boldsymbol{\mathcal{X}} \in \mathbb{F}^{2\times 2\times 2}$ we have

$$\mathbf{X}_{(2)} = \begin{bmatrix} \boldsymbol{\mathcal{X}}_{1,:,1} & \boldsymbol{\mathcal{X}}_{2,:,1} & \boldsymbol{\mathcal{X}}_{1,:,2} & \boldsymbol{\mathcal{X}}_{2,:,2} \end{bmatrix}.$$

Once the mode of the matricization is chosen, there is a one-to-one correspondence between tensors and their matricizations: for example a matrix $\mathbf{M} \in \mathbb{F}^{d_1 \times d_2 d_3}$ uniquely determines a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{F}^{d_1 \times d_2 \times d_3}$ with the relation $\mathbf{M} = \mathbf{T}_{(1)}$.

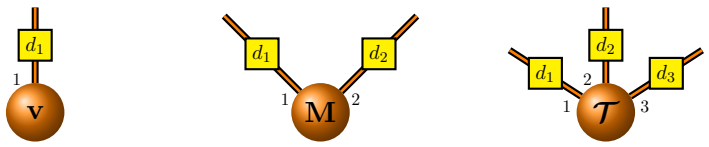


Figure 1.1: Tensor network representation of a vector $\mathbf{v} \in \mathbb{R}^{d_1}$, a matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ and a 3rd-order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$.

The vectorization of a tensor is defined by $\text{vec}(\mathcal{T}) = \text{vec}(\mathbf{T}_{(1)})$. The *inner product* between two tensors $\mathcal{S}$ and $\mathcal{T}$ (of the same size) is defined by $\langle \mathcal{S}, \mathcal{T} \rangle = \langle \text{vec}(\mathcal{S}), \text{vec}(\mathcal{T}) \rangle$ and the Frobenius norm is defined by $\|\mathcal{T}\|_F^2 = \langle \mathcal{T}, \mathcal{T} \rangle$. Observe that it follows from this last definition that the Frobenius norm of a tensor is equal the one of its vectorization and to the one of any of its matricizations.

1.2.2 Tensor Networks

Tensor network diagrams, which have been introduced in (Penrose, 1971), are a very useful tool to visualize tensor decompositions and multilinear operations of tensor contractions (see e.g. Kressner and Tobler (2012), Huckle, Waldherr, and Schulte-Herbrüggen (2013), and Holtz, Rohwedder, and Schneider (2012b) for more details). They have been traditionally used in the quantum physic community to capture the relevant entanglement properties of a system. Tensor networks are also closely related to tensor decompositions as they allow one to express a tensor as a set of smaller tensors of lower order connected to each other, and thus to reduce the number of parameters needed to represent the original tensor. In doing so, they make it possible to manipulate extremely large tensors by simply operating on the parameterization induced by their tensor network representation, which is currently leading to an increasing interest from the machine learning and data mining communities (see e.g. Stoudenmire and Schwab (2016), Cichocki (2014), Novikov et al. (2015), and Zhao et al. (2016))

In a tensor network, tensors are represented as nodes in a graph and each outgoing edge from a node represents a mode of the tensor. In contrast with classical graphs, an edge in a tensor network does not need to connect two nodes but can remain *open*. Such open (or free) edges (or legs) correspond to modes of the tensor represented by the tensor network. For example a vector will be represented by a graph with only one node with one free leg, and a matrix by a node with two free legs (see Figure 1.1).

An edge connecting two nodes in a tensor network represents a *contraction* between the two corresponding modes, that is a summation over the corresponding indices of the tensors. For example, the summation in the classical matrix product $(\mathbf{AB})_{i,j} = \sum_k \mathbf{A}_{i,k}\mathbf{B}_{k,j}$ is a contraction between the second mode of $\mathbf{A}$ and the first mode of $\mathbf{B}$. Examples of common matrix and vector operations represented as tensor networks are shown in Figure 1.2. Observe in this figure that the number of free legs corresponds to the order of the tensor: the two tensor networks on the left represent scalars, the one in the top-right corner a vector and the one in the bottom-right a matrix.

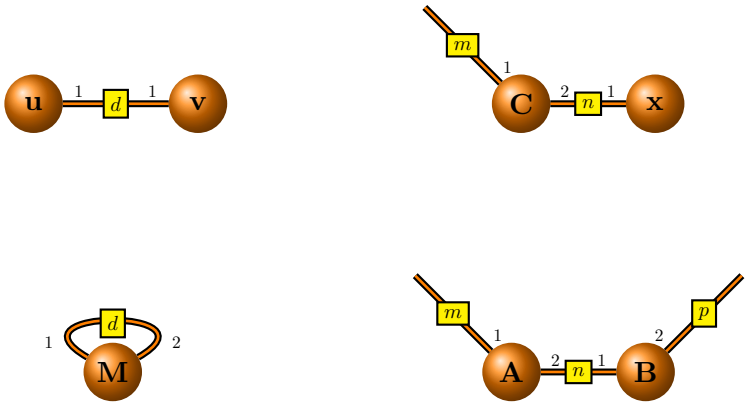


Figure 1.2: Tensor networks for common operations on vectors and matrices. From top-left to bottom-right: the scalar product between two vectors in $\mathbb{R}^d$, the product $\mathbf{Cx}$ of an $m \times n$ matrix with an n-dimensional vector, the trace of a $d \times d$ square matrix, and the matrix product between two matrices of size $m \times n$ and $n \times p$.

1.2.3 Basic Operations on Tensors

We now define several operations on tensors that we will use throughout this thesis.

Outer product. We first straightforwardly extend the definition of outer product given above for vectors to tensors. Given two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{F}^{m_1 \times \cdots \times m_p}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{F}^{n_1 \times \cdots \times n_q}$ of order p and q respectively, their outer product $\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}} \in \mathbb{F}^{m_1 \times \cdots \times m_p \times n_1 \times \cdots \times n_q}$ is the tensor of order $p+q$ defined by

$$(\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}})_{i_1,\cdots,i_p,j_1,\cdots,j_q} = \boldsymbol{\mathcal{A}}_{i_1,\cdots,i_p} \boldsymbol{\mathcal{B}}_{j_1,\cdots,j_q}$$

for all indices $i_1 \in [m_1], \cdots, i_p \in [m_p], j_1 \in [n_1], \cdots, j_q \in [n_q]$. It is easy to check that the outer product is associative. In a tensor network the outer product of two tensors simply consists in juxtaposing the tensor networks representing each tensor without adding any edge.

Kronecker product. The Kronecker product is an operation closely related to the outer product that is usually defined for matrices. Given a matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$ with entries $(a_{i,j})_{1 \leq i \leq m, 1 \leq j \leq n}$ and a matrix $\mathbf{B} \in \mathbb{F}^{p \times q}$, their Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{F}^{mp \times nq}$ is the block matrix defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}.$$

Observe that the matrix $\mathbf{A} \otimes \mathbf{B} \in \mathbb{F}^{mp \times nq}$ can be obtained by rearranging the entries of the 4th order tensor $\mathbf{A} \circ \mathbf{B} \in \mathbb{F}^{m \times n \times p \times q}$. In particular, if $n = 1$ and $q = 1$ then

the matrices $\mathbf{A}$ and $\mathbf{B}$ can be seen as vectors which we denote by $\mathbf{a}$ and $\mathbf{b}$, and we have the relation $\mathbf{a} \otimes \mathbf{b} = \mathrm{vec}(\mathbf{a} \circ \mathbf{b}) = \mathrm{vec}(\mathbf{a}\mathbf{b}^\top)$. More generally we have the relation $\mathrm{vec}(\mathbf{u}^1 \circ \mathbf{u}^2 \circ \cdots \circ \mathbf{u}^p) = \mathbf{u}^1 \otimes \mathbf{u}^2 \otimes \cdots \otimes \mathbf{u}^p$. Useful properties of the Kronecker product that we will use in this thesis are (Laub, 2004, Theorem 13.3, 13.4 and 13.6)

$$(\mathbf{A} \otimes \mathbf{B})^\top = \mathbf{A}^\top \otimes \mathbf{B}^\top$$
$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$
$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}.$$

We will refer to the last equality as the *mixed-product* property of the Kronecker product.

We extend the Kronecker product to tensors in the following way. Given two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{F}^{m_1 \times \cdots \times m_p}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{F}^{n_1 \times \cdots \times n_p}$ *of the same order* p, their Kronecker product is the pth order tensor $\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}} \in \mathbb{F}^{m_1 n_1 \times \cdots \times m_p n_p}$ defined by the relation

$$(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}})_{(1)} = \mathbf{A}_{(1)} \otimes \mathbf{B}_{(1)}$$

where $(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}})_{(1)}$ is the mode-1 matricization of $\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$. It is easy to check that this definition implies that $(\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}})_{(i)} = \mathbf{A}_{(i)} \otimes \mathbf{B}_{(i)}$ for any $i \in [p]$ and that the Kronecker product is associative.

mode-n product. We now introduce a fundamental product operation between tensors and matrices. The *mode-n matrix product* of a pth order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ (with $1 \leq n \leq p$) with a matrix $\mathbf{A} \in \mathbb{F}^{m \times d_n}$ is the pth order tensor $\boldsymbol{\mathcal{T}} \times_n \mathbf{A}$ of size $d_1 \times \cdots \times d_{n-1} \times m \times d_{n+1} \times \cdots \times d_p$ obtained by contracting the nth mode of $\boldsymbol{\mathcal{T}}$ with the second mode of $\mathbf{A}$. Formally,

$$(\boldsymbol{\mathcal{T}} \times_n \mathbf{A})_{i_1, \cdots, i_p} = \sum_{j=1}^{d_n} \boldsymbol{\mathcal{T}}_{i_1, \cdots, i_{n-1}, j, i_{n+1}, \cdots, i_p} \mathbf{A}_{i_n, j}$$

for all indices $i_1 \in [d_1], \cdots, i_{n-1} \in [d_{n-1}], i_n \in [m], i_{n+1} \in [d_{n+1}], \cdots, i_p \in [d_p]$. Given matrices $\mathbf{B}_i \in \mathbb{R}^{d_i \times m_i}$ for $i \in [p]$ we will sometimes use the notation

$$\boldsymbol{\mathcal{T}}(\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_p) = \boldsymbol{\mathcal{T}} \times_1 \mathbf{B}_1^\top \times_2 \cdots \times_p \mathbf{B}_p^\top \in \mathbb{R}^{m_1 \times \cdots \times m_p}$$

which has been used in several papers (see e.g. Anandkumar et al., 2014). It is easy to check that the n-mode product satisfies the following identities:

$$(\boldsymbol{\mathcal{T}} \times_n \mathbf{A})_{(n)} = \mathbf{A}\mathbf{T}_{(n)}$$
$$(\boldsymbol{\mathcal{T}} \times_n \mathbf{A}) \times_n \mathbf{B} = \boldsymbol{\mathcal{T}} \times_n \mathbf{BA}$$
$$\langle \boldsymbol{\mathcal{T}} \times_n \mathbf{A}, \boldsymbol{\mathcal{S}} \rangle = \langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{S}} \times_n \mathbf{A}^\top \rangle$$

where we assume compatible dimensions of the tensors $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$. Moreover, if $m, n \in [p]$ with $m \neq n$, the n-mode and m-mode products commute:

$$\boldsymbol{\mathcal{T}} \times_m \mathbf{A} \times_n \mathbf{B} = \boldsymbol{\mathcal{T}} \times_n \mathbf{B} \times_m \mathbf{A}.$$

Similarly, the *mode-n vector product* of the tensor $\boldsymbol{\mathcal{T}}$ with a vector $\mathbf{v} \in \mathbb{F}^{d_n}$ is the

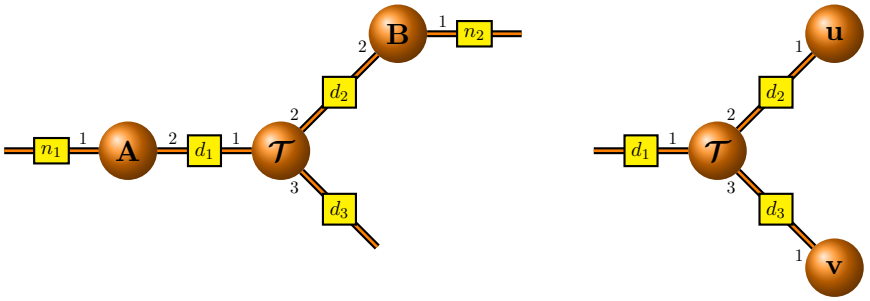


Figure 1.3: Tensor networks of the mode-n matrix and vector products with a tensor $\mathcal{T}$ of size $d_1 \times d_2 \times d_3$: (left) $\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B}$ is a tensor of size $n_1 \times n_2 \times d_3$ and (right) $\mathcal{T} \bullet_2 \mathbf{u} \bullet_3 \mathbf{v}$ is a d_1-dimensional vector.

$(p-1)$th order tensor $\mathcal{T} \bullet_n \mathbf{v} \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times d_{n+1} \times \cdots \times d_p}$ obtained by contracting the nth mode of $\mathcal{T}$ with the unique mode of $\mathbf{v}$. Formally,

$$(\mathcal{T} \bullet_1 \mathbf{v})_{i_1,\cdots,i_{n-1},i_{n+1},\cdots,i_p} = \sum_{i_n=1}^{d_n} \mathcal{T}_{i_1,\cdots,i_p} \mathbf{v}_{i_n}$$

for all indices $i_1 \in [d_1], \cdots, i_{n-1} \in [d_{n-1}], i_{n+1} \in [d_{n+1}], \cdots, i_p \in [d_p]$.

Examples of mode-n matrix and vector products represented as tensor networks are shown in Figure 1.3.

Direct sum of tensors. For any pth order tensors $\mathcal{A} \in \mathbb{F}^{m_1 \times \cdots \times m_p}$ and $\mathcal{B} \in \mathbb{F}^{n_1 \times \cdots \times n_p}$, we define their direct sum $\mathcal{A} \oplus \mathcal{B} \in \mathbb{F}^{(m_1+n_1) \times \cdots \times (m_p+n_p)}$ by

$$(\mathcal{A} \oplus \mathcal{B})_{i_1,\cdots,i_p} = \begin{cases} \mathcal{A}_{i_1,\cdots,i_p} & \text{if } 1 \le i_k \le m_k \text{ for all } k \in [p] \\ \mathcal{B}_{i_1-m_1,\cdots,i_p-m_p} & \text{if } m_k < i_k \le m_k + n_k \text{ for all } k \in [p] \\ 0 & \text{otherwise.} \end{cases}$$

This definition generalizes the direct sum of two vectors or two matrices. Indeed, recall that if $\mathbf{a}$ and $\mathbf{b}$ are vectors we have $\mathbf{a} \oplus \mathbf{b} = (\mathbf{a}^\top \ \mathbf{b}^\top)^\top$. Similarly, if $\mathbf{A}$ and $\mathbf{B}$ are matrices, their direct sum is the block diagonal matrix

$$\mathbf{A} \oplus \mathbf{B} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}.$$

Tensor contraction. Lastly, we define a general contraction operator that acts on cubic tensors by contracting two modes. In terms of tensor networks, applying the contraction operator corresponds to connecting two free legs in a tensor network. Consequently, applying the contraction operator to a pth order tensor will result in a tensor of order $p-2$.

Formally, for any $1 \le m < n \le p$, we define the *contraction operator* (or generalized trace operator)

$$\langle\!\langle \cdot \rangle\!\rangle_{(m,n)} : (\mathbb{F}^d)^{\otimes p} \to (\mathbb{F}^d)^{\otimes (p-2)}$$

that contracts the mth and nth mode of a pth order tensor $\boldsymbol{\mathcal{T}} \in (\mathbb{F}^d)^{\otimes p}$ by

$$\left(\langle\!\langle \boldsymbol{\mathcal{T}} \rangle\!\rangle_{(m,n)}\right)_{i_1,\cdots,i_{m-1},i_{m+1},\cdots,i_{n-1},i_{n+1},\cdots,i_p} = \sum_{k=1}^{d} \boldsymbol{\mathcal{T}}_{i_1,\cdots,i_{m-1},k,i_{m+1},\cdots,i_{n-1},k,i_{n+1},\cdots,i_p}$$

for all indices $i_1, \cdots, i_{m-1}, i_{m+1}, \cdots, i_{n-1}, i_{n+1}, \cdots, i_p \in [d]$. We will extend this notation to the composition of multiple contractions in the straightforward way. For example, if $\boldsymbol{\mathcal{T}}$ is a 5th order tensor we have

$$(\langle\!\langle \boldsymbol{\mathcal{T}} \rangle\!\rangle_{(1,5),(2,3)})_i = \sum_{jk} \boldsymbol{\mathcal{T}}_{j,k,k,i,j}.$$

As an illustration, we can express the trace, the matrix product, the n-mode product and the inner product using the outer product and the contraction operator:

$$\begin{aligned}
\mathrm{Tr}(\mathbf{A}) &= \langle\!\langle \mathbf{A} \rangle\!\rangle_{(1,2)}, \\
\mathbf{AB} &= \langle\!\langle \mathbf{A} \circ \mathbf{B} \rangle\!\rangle_{(2,3)}, \\
\boldsymbol{\mathcal{T}} \times_n \mathbf{A} &= \langle\!\langle \mathbf{A} \circ \boldsymbol{\mathcal{T}} \rangle\!\rangle_{(2,n+2)}, \text{ and} \\
\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle &= \langle\!\langle \boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}} \rangle\!\rangle_{(1,p+1),(2,p+2),\cdots,(p,2p)}.
\end{aligned}$$

1.2.4 Tensor Decompositions and Rank of a Tensor

We now describe several *tensor decomposition models*. One of the motivations behind the development of tensor decomposition techniques is that it is often not possible to store all the entries of a high order tensor explicitly. Tensor decompositions can be seen as compression schemes that help reduce storage requirements.

We will mainly focus on the CP (CANDECOMP/PARAFAC) and Tucker decompositions. Each of these decompositions gives rise to a different notion of *rank* of a tensor: the *CP-rank* and the *multilinear rank*. In the case of tensors of order 2, these two notions of rank coincide but correspond to two different interpretations of the rank of a matrix:

(1) The rank of a matrix $\mathbf{M}$ is the minimum integer R such that $\mathbf{M}$ can be written as a sum of R rank one matrices.

(2) The rank of a matrix is the dimension of the vector space spanned by its columns (or equivalently by its rows).

As we will see below, the first interpretation corresponds to the CP-rank of a tensor while the second one corresponds to the multilinear rank.

CP decomposition and CP-rank. A rank one tensor $\boldsymbol{\mathcal{U}} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ is a tensor that can be written as an outer product of p vectors:

$$\boldsymbol{\mathcal{U}} = \mathbf{u}_1 \circ \mathbf{u}_2 \circ \cdots \circ \mathbf{u}_p$$

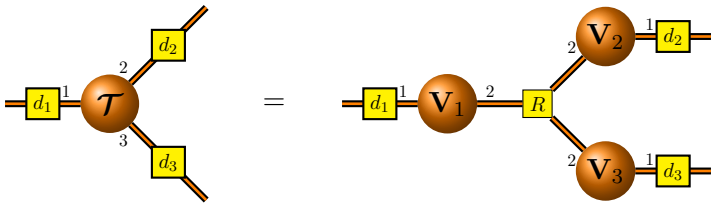


Figure 1.4: CP decomposition of a 3rd-order tensor $\mathcal{T} = [\![\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3]\!]$ of rank R as a tensor network.

where $\mathbf{u}_1 \in \mathbb{F}^{d_1}, \cdots, \mathbf{u}_p \in \mathbb{F}^{d_p}$. A *CP decomposition* of a tensor $\mathcal{T} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ consists in expressing $\mathcal{T}$ as a sum of rank one tensors:

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{v}_1^r \circ \cdots \circ \mathbf{v}_p^r \tag{1.1}$$

where $\mathbf{v}_i^r \in \mathbb{R}^{d_i}$ for all $i \in [p]$ and all $r \in [R]$. Stacking the vectors $\mathbf{v}_i^1, \cdots, \mathbf{v}_i^R$ in a matrix $\mathbf{V}_i \in \mathbb{F}^{d_i \times R}$ for each $i \in [p]$, the notation $\mathcal{T} = [\![\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_p]\!]$ is sometimes used to denote Equation (1.1). The CP decomposition requires the storage of $R(d_1 + d_2 + \cdots + d_p)$ entries (instead of $d_1 d_2 \cdots d_p$ for the original tensor) which makes it attractive for small values of R. A tensor network representation of a CP decomposition is given in Figure 1.4 where an *hyperedge* is used to represent the common contraction corresponding to the summation over r in Eq. (1.1).

The *tensor rank* or *CP rank* of $\mathcal{T}$ is the smallest integer R for which $\mathcal{T}$ admits a CP decomposition with R terms, which will be denoted by $\mathrm{rank}_{CP}(\mathcal{T}) = R$. Unlike for matrices, the set of tensors with tensor rank at most R is not closed in general, which makes the problem of finding a best low rank approximation of a tensor ill-posed (De Silva and Lim, 2008). There exist several algorithms to find a CP decomposition of a tensor. The simplest one relies on the Alternating Least Squares (ALS) method and consists in minimizing the objective $\|\mathcal{T} - [\![\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3]\!]\|_F^2$ with respect to one of the matrices $\mathbf{V}_i$ while keeping the other ones fixed; this process is repeated in turn for each matrices $\mathbf{V}_i$ until convergence. Even though this method is not guaranteed to converge to a global minimizer, it often performs well in practice. We refer the reader to (Kolda and Bader, 2009) for properties of the CP rank and details on computing the CP decomposition; pointers to further developments since 2009 can also be found in (Grasedyck, Kressner, and Tobler, 2013) and a brief review on bounds for the CP rank can be found in (Sidiropoulos et al., 2016).

Tucker decomposition and multilinear rank. We now define the Tucker decomposition and the related notion of multilinear rank. Since these notions are at the core of the algorithm we propose in Chapter 4 for a problem of regression with tensor structured outputs, we will give more technical details than we did for the CP-decomposition and the CP rank.

A Tucker decomposition consists in expressing a tensor $\mathcal{T} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ as a core tensor

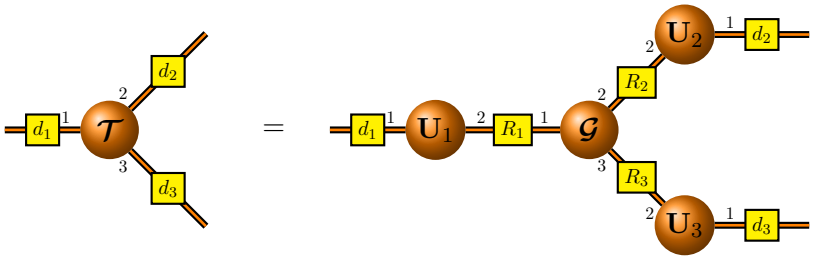


Figure 1.5: Tucker decomposition with a core tensor of size $R_1 \times R_2 \times R_3$ of a 3rd-order tensor $\mathcal{T} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$ as a tensor network.

$\mathcal{G}$ transformed by a column-wise orthogonal matrix along each mode:

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_p \mathbf{U}_p, \tag{1.2}$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_p}$, $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ for $i \in [p]$ and $\mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}$ for all $i \in [p]$. Even though the column-wise orthogonality of the matrices $\mathbf{U}_i$ is not required, it is often assumed and does not lead to any loss of generality[a]. The Tucker decomposition requires the storage of $R_1 R_2 \cdots R_p + d_1 R_1 + d_2 R_2 + \cdots + d_p R_p$ entries. A tensor network representation of the Tucker decomposition is given in Figure 1.5.

The matricizations and the vectorization of the Tucker decomposition in Eq. (1.2) lead to the following useful identities:

$$\mathbf{T}_{(n)} = \mathbf{U}_n \mathbf{G}_{(n)} (\mathbf{U}_p \otimes \cdots \otimes \mathbf{U}_{n+1} \otimes \mathbf{U}_{n-1} \otimes \cdots \otimes \mathbf{U}_1)^\top \tag{1.3}$$

$$\mathrm{vec}(\mathcal{T}) = (\mathbf{U}_p \otimes \mathbf{U}_{p-1} \otimes \cdots \otimes \mathbf{U}_1)\mathrm{vec}(\mathcal{G}). \tag{1.4}$$

Equation (1.3) implies that for any $n \in [p]$ the rank of the matricization $\mathbf{T}_{(n)}$ is bounded by R_n. This naturally leads to the notion of *multilinear rank* of a tensor which is denoted by $\mathrm{rank}_{ml}(\mathcal{T})$ and defined by

$$\mathrm{rank}_{ml}(\mathcal{T}) = (R_1, \cdots, R_p) \quad \text{where } R_n = \mathrm{rank}(\mathbf{T}_{(n)}) \text{ for each } n \in [p].$$

Observe that each R_n corresponds to the dimension of the vector space spanned by the mode-n fibers of $\mathcal{T}$, which is similar to the interpretation of the matrix rank as the dimension of the space spanned by its columns. We will write $\mathrm{rank}_{ml}(\mathcal{T}) \leq (S_1, \cdots, S_p)$ whenever $S_1 \leq R_1$, $S_2 \leq R_2$, $\cdots$, and $S_p \leq R_p$. Contrary to the CP rank, the set of tensors with multilinear rank at most $(R_1, \cdots, R_p)$ is closed.

It is easy to check that the multilinear rank of a tensor $\mathcal{T}$ exactly coincides with the smallest tuple $(R_1, \cdots, R_p)$ such that $\mathcal{T}$ admits a Tucker decomposition with a core tensor of size $R_1 \times \cdots \times R_p$. Indeed, as we mentioned earlier if $\mathcal{T}$ admits a Tucker decomposition with a core of size $R_1 \times \cdots \times R_p$ then $\mathrm{rank}_{ml}(\mathcal{T}) \leq (R_1, \cdots, R_p)$. Conversely, if $\mathrm{rank}_{ml}(\mathcal{T}) = (R_1, \cdots, R_p)$ we can obtain a Tucker decomposition in the following way: for each $n \in [p]$ let $\mathbf{U}_n \in \mathbb{R}^{d_n \times R_n}$ be the matrix having for columns the R_n left singular

[a] Indeed, if the matrices $\mathbf{U}_i$ are not column-wise orthogonal it suffices to perform QR decompositions $\mathbf{U}_i = \mathbf{Q}_i \mathbf{R}_i$ where $\mathbf{Q}_i \in \mathbb{F}^{d_i \times R_i}$ is column-wise orthogonal and $\mathbf{R}_i \in \mathbb{F}^{R_i \times R_i}$ to obtain the decomposition $(\mathcal{G} \times_1 \mathbf{R}_1 \times_2 \cdots \times_p \mathbf{R}_p) \times_1 \mathbf{Q}_1 \times_2 \cdots \times_p \mathbf{Q}_p$.

vectors of $\mathbf{T}_{(n)}$ and let $\mathcal{G} = \mathcal{T} \times_1 \mathbf{U}_1^\top \times_2 \cdots \times_p \mathbf{U}_p^\top \in \mathbb{F}^{R_1 \times \cdots \times R_p}$. We then have

$$\mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_p \mathbf{U}_p = \mathcal{T} \times_1 \mathbf{U}_1\mathbf{U}_1^\top \times_2 \cdots \times_p \mathbf{U}_p\mathbf{U}_p^\top = \mathcal{T}$$

where we used the simple fact that $\mathbf{U}_n\mathbf{U}_n^\top$ is the matrix of the orthogonal projection onto the left singular vectors of $\mathbf{T}_{(n)}$ which implies $\mathcal{T} \times_n \mathbf{U}_n\mathbf{U}_n^\top = \mathcal{T}$ for any $n \in [p]$.

Higher-order SVD. Even though finding the best low multilinear rank approximation of a tensor $\mathcal{T}$ is an NP-hard problem (Hillar and Lim, 2013), the higher-order SVD (HOSVD) introduced in (De Lathauwer, De Moor, and Vandewalle, 2000; De Lathauwer, 1997) for approximating a tensor with a Tucker decomposition of lower multilinear rank offers good approximation guarantees. In HOSVD, each factor matrices $\mathbf{U}_n$ is obtained by taking the top R_n left singular vectors of $\mathbf{T}_{(n)}$ and the core tensor is given by $\mathcal{G} = \mathcal{T} \times_1 \mathbf{U}_1^\top \times_2 \cdots \times_p \mathbf{U}_p^\top \in \mathbb{F}^{R_1 \times \cdots \times R_p}$. If $\mathcal{T}^*$ denotes the best approximation of $\mathcal{T}$ with multilinear rank $(R_1, \cdots, R_p)$, that is $\mathcal{T}^*$ is a minimizer of the optimization problem

$$\min_{\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_p}} \|\mathcal{T} - \mathcal{X}\|_F^2 \quad \text{subject to } \operatorname{rank}_{ml}(\mathcal{X}) \leq (R_1, \cdots, R_p), \tag{1.5}$$

then the low multilinear rank tensor $\hat{\mathcal{T}}$ returned by HOSVD satisfies

$$\|\mathcal{T} - \hat{\mathcal{T}}\|_F^2 \leq p\|\mathcal{T} - \mathcal{T}^*\|_F^2,$$

i.e. HOSVD is a p-approximation algorithm for the minimization problem (1.5). For sake of completeness we give here the short and elegant proof of this result provided in (Grasedyck, 2010). The proof relies on the following lemma.

Lemma 1. *Let $\mathcal{T} \in \mathbb{F}^{d_1 \times \cdots \times d_p}$ be a pth order tensor, let $m, n \in [p]$, and let $\mathbf{P} \in \mathbb{F}^{d_m \times d_m}$ and $\mathbf{Q} \in \mathbb{F}^{d_n \times d_n}$ be two orthogonal projection matrices. Then*

$$\|\mathcal{T} - \mathcal{T} \times_m \mathbf{P} \times_n \mathbf{Q}\|_F^2 \leq \|\mathcal{T} - \mathcal{T} \times_m \mathbf{P}\|_F^2 + \|\mathcal{T} - \mathcal{T} \times_n \mathbf{Q}\|_F^2.$$

Proof. First observe that for any orthogonal projection matrix $\mathbf{\Pi}$ and any tensors $\mathcal{A}, \mathcal{B}$ we have

$$\|\mathcal{A} \times_n \mathbf{\Pi}\|_F^2 \leq \|\mathcal{A}\|_F^2 \quad \text{and} \quad \|\mathcal{A} \times_n (\mathbf{I} - \mathbf{\Pi}) + \mathcal{B} \times_n \mathbf{\Pi}\|_F^2 = \|\mathcal{A} \times_n (\mathbf{I} - \mathbf{\Pi})\|_F^2 + \|\mathcal{B} \times_n \mathbf{\Pi}\|_F^2.$$

Both equations follow from the fact that the Frobenius norm of a tensor is equal to the one of any of its matricization. Thus

$$\|\mathcal{A} \times_n \mathbf{\Pi}\|_F^2 = \|\mathbf{\Pi}\mathbf{A}_{(n)}\|_F^2 \leq \|\mathbf{A}_{(n)}\|_F^2 = \|\mathcal{A}\|_F^2$$

since $\mathbf{\Pi}$ is a projection. The second equality is proved similarly using the orthogonality of $\mathbf{\Pi}$ and $\mathbf{I} - \mathbf{\Pi}$.

Then, under the hypothesis of the lemma, we have

$$\begin{aligned}\|\mathcal{T} - \mathcal{T} \times_m \mathbf{P} \times_n \mathbf{Q}\|_F^2 &= \|\mathcal{T} \times_m (\mathbf{I} - \mathbf{P}) + (\mathcal{T} - \mathcal{T} \times_n \mathbf{Q}) \times_m \mathbf{P}\|_F^2 \\ &= \|\mathcal{T} \times_m (\mathbf{I} - \mathbf{P})\|_F^2 + \|(\mathcal{T} - \mathcal{T} \times_n \mathbf{Q}) \times_m \mathbf{P}\|_F^2 \\ &\leq \|\mathcal{T} - \mathcal{T} \times_m \mathbf{P}\|_F^2 + \|\mathcal{T} - \mathcal{T} \times_n \mathbf{Q}\|_F^2. \qquad \square\end{aligned}$$

Proving the approximation guarantees of HOSVD stated above is then easy: the low multilinear rank tensor computed by HOSVD is defined by

$$\hat{\mathcal{T}} = \mathcal{T} \times_1 \mathbf{U}_1\mathbf{U}_1^\top \times_2 \cdots \times_p \mathbf{U}_p\mathbf{U}_p^\top$$

where $\mathbf{U}_i\mathbf{U}_i^\top$ is the matrix of the orthogonal projection onto the space spanned by the top R_i left singular vectors of $\mathbf{T}_{(i)}$ for each $i \in [p]$. Thus $\mathbf{U}_i\mathbf{U}_i^\top\mathbf{T}_{(i)}$ is the best rank R_i approximation of $\mathbf{T}_{(i)}$ which implies $\|\mathcal{T} - \mathcal{T} \times_i \mathbf{U}_i\mathbf{U}_i^\top\|_F^2 \leq \|\mathcal{T} - \mathcal{T}^*\|_F^2$ for all $i \in [p]$, where $\mathcal{T}^*$ is a minimizer of problem (1.5). Using successive applications of the previous lemma it follows that

$$\begin{aligned}\|\mathcal{T} - \hat{\mathcal{T}}\|_F^2 &= \|\mathcal{T} - \mathcal{T} \times_1 \mathbf{U}_1\mathbf{U}_1^\top \times_2 \cdots \times_p \mathbf{U}_p\mathbf{U}_p\|_F^2 \\ &\leq \sum_{i=1}^{p} \|\mathcal{T} - \mathcal{T} \times_i \mathbf{U}_i\mathbf{U}_i^\top\|_F^2 \leq p\|\mathcal{T} - \mathcal{T}^*\|_F^2.\end{aligned}$$

Even though HOSVD does not return an optimal solution, the approximation guarantees it provides are often sufficient in practice. Various alternatives have been proposed to improve the HOSVD approximate solution, again we refer the reader to (Kolda and Bader, 2009) and (Grasedyck, Kressner, and Tobler, 2013) for more details.

Other tensor decomposition models. We presented two tensor decomposition models. On the one hand a CP decomposition of a pth order tensor with R terms requires to store $R(d_1 + d_2 + \cdots + d_p)$ entries which makes it very attractive, but the set of tensors with CP rank at most R is not closed and algorithms for computing a low CP rank approximation of a tensor lack strong theoretical guarantees. On the other hand the closedness of the set of tensors with low multilinear rank and the existence of quasi-optimal SVD-based compression algorithms (such as HOSVD) make the Tucker decomposition very appealing. However a Tucker decomposition requires the storage of the core tensor $\mathcal{G}$ of size $R_1 \times \cdots \times R_p$ which may become very large as p increases. This has motivated the search for other decomposition models that avoid this exponential growth of the size of the model while preserving the main advantages of the Tucker decomposition: closedness and quasi-optimal compression algorithms. To conclude this chapter, we mention some of these other decomposition models.

In the *tensor train (TT) format* a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_p}$ is factorized into p core tensors $\mathcal{G}_1 \in \mathbb{R}^{d_1 \times R_1}, \mathcal{G}_2 \in \mathbb{R}^{R_1 \times d_2 \times R_2}, \cdots, \mathcal{G}_{p-1} \in \mathbb{R}^{R_{p-2} \times d_{p-1} \times R_{p-1}}, \mathcal{G}_p \in \mathbb{R}^{R_{p-1} \times d_p}$ as

$$\mathcal{T}_{i_1,\cdots,i_p} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{p-1}=1}^{R_{p-1}} (\mathcal{G}_1)_{i_1,r_1} (\mathcal{G}_2)_{r_1,i_2,r_2} (\mathcal{G}_3)_{r_2,i_3,r_3} \cdots (\mathcal{G}_{p-1})_{r_{p-2},i_{p-1},r_{p-1}} (\mathcal{G}_p)_{r_{p-1},i_p}$$

for all indices $i_1 \in [d_1], \cdots, i_p \in [d_p]$. The smallest tuple $(R_1, \cdots, R_{p-1})$ for which such

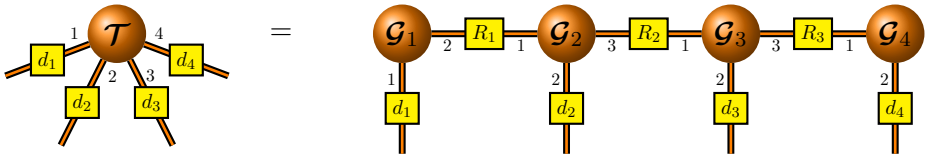


Figure 1.6: Tensor train decomposition of a 4th order tensor.

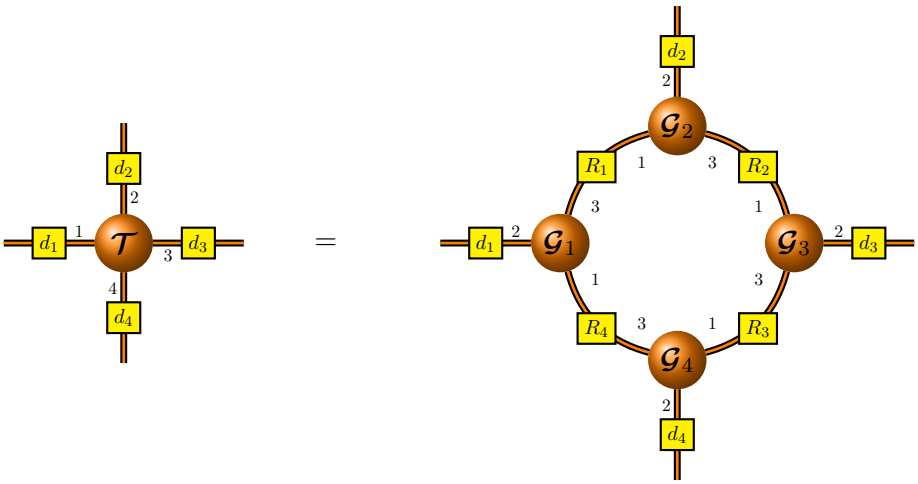


Figure 1.7: Tensor ring decomposition of a 4th order tensor.

a decomposition exists is called the *TT rank* of the tensor $\mathcal{T}$. Using the contraction operator introduced previously we can rewrite the previous equation as

$$\mathcal{T} = \langle\langle \mathcal{G}_1 \circ \mathcal{G}_2 \circ \cdots \circ \mathcal{G}_p \rangle\rangle_{(2,3),(5,6),(8,9),\cdots,(3p-4,3p-3)}.$$

An example of such a decomposition is shown in Figure 1.6 (where we see that the core tensors are somehow arranged as a *train* hence the name of the decomposition). The tensor train decomposition has been introduced in (Oseledets, 2011) in the context of numerical analysis, but it had been considered earlier in the density renormalization group method for simulating quantum systems (White, 1992). In this area the term *matrix product state* representation is used for this decomposition. Observe that a tensor in the TT format requires the storage of $d_1R_1 + R_1d_2R_2 + \cdots + R_{p-2}d_{p-1}R_{p-1} + R_{p-1}d_p$ entries which does not exhibit the dependency on the product of the ranks appearing in the Tucker decomposition. Moreover, a quasi-optimal SVD based compression algorithm has been proposed in (Oseledets, 2011) and the set of tensors of fixed TT rank is closed and forms a smooth manifold (Holtz, Rohwedder, and Schneider, 2012a).

Other decomposition models having these desirable properties have been proposed. For example, the *hierarchical Tucker (HT) decomposition*, which consists in factorizing a tensor into a binary tree tensor network, has been proposed in (Grasedyck, 2010) and (Hackbusch and Kühn, 2009). Lastly we would like to mention the *tensor ring de-*

composition model that has been very recently proposed in (Zhao et al., 2016). This model is very close to the TT format as it consists in decomposing a tensor into a ring of p low-dimensional 3rd order core tensors (instead of the string of core tensors from the TT format), see Figure 1.7. The authors show that tensor ring decompositions are equivalent to linear combinations of TT decompositions, and are invariant under cyclic permutation of the rank. It turns out that the TT format is closely related to the computations of a weighted automaton on strings, and that the tensor ring decomposition model is itself related to the computations of Graph Weighted Models on circular strings that we will introduce in the next chapter.

2 Recognizable Series on Graphs

Contents

2.1 Introduction

Real-valued functions whose domains are composed of syntactical structures, such as strings, trees or graphs, are widely used in computer science. One way to handle such functions is by means of devices computing them. *Weighted automata*, which are able to jointly analyze the structure of a syntactical input and to compute an associated output value, are such a computing device. The first two chapters of this thesis will be related to weighted automata. Weighted automata are finite state machines that define uniform computations on a set of structured objects (e.g. strings or trees). For example, a string weighted automaton (WA) computes a function that takes as input a string on a finite alphabet, e.g. $abba$, and outputs a value in a semi-ring, e.g. a real value. Roughly speaking, a WA is composed of states and weighted transitions between these states. A string can be parsed by a weighted automaton by following a path of states using

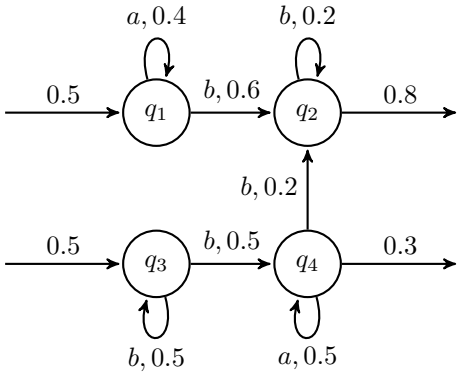


Figure 2.1: An example of weighted automaton that computes a probability distribution on the set of strings on the alphabet $\Sigma = \{a, b\}$. There exist two paths to parse the string bb in this automaton: $q_3q_3q_4$ and $q_3q_4q_2$. The value it computes on the string bb is thus equal to the sum of the weights of these paths: $0.5 * 0.5 * 0.5 * 0.3 + 0.5 * 0.5 * 0.2 * 0.8$.

weighted transitions labeled by the letters in the string, and a weight can naturally be associated with each path in the automaton. A WA then computes a value by summing the weights over all possible parsing paths. In particular, weighted automata can define probability distributions on the set of strings (or trees), which makes them relevant to machine learning. An example of a WA, along with an example of computation, is shown in Figure 2.1. Weighted automata on trees on a ranked alphabet can be defined in a similar fashion in terms of states and weighted transitions between states.

The notion of weighted automata gives rise to the notion of *recognizable series*: a series is a function from a set of structured objects (e.g. strings, trees, graphs) to a semi-ring, and it is recognizable if it can be computed by a weighted automaton. Weighted automata have been defined for strings and trees, but their extension to graphs is challenging. On the other hand, recognizable series defined on strings and trees have equivalent algebraic characterizations by mean of *linear representations*. A linear representation is an algebraic representation of a weighted automaton that associates each symbol in a (ranked) alphabet to a linear or multilinear map acting on some vector space. The dimension of this vector space corresponds to the number of states in the weighted automaton and this vector space is henceforth called the *state space*. The computation of the automaton on a string (or a tree) is then achieved by successive compositions of these maps. In (Bailly, Denis, and Rabusseau, 2015) we showed that this algebraic formalism can naturally be extended to hypergraphs recognizable series. For simplicity of presentation, we will focus on graph recognizable series in this chapter.

In Section 2.1.1, we formally define weighted automata on strings and trees using this algebraic representation, and we show how computations of a weighted automaton on strings or trees can be interpreted as a simple mapping between the input (a string or a tree) and a tensor network reflecting its structure. This interpretation suggests a natural extension of weighted automata to labeled graphs and leads us to define the novel notion of *Graph Weighted Models* (GWM): a computational model that generically associates a tensor network to a graph and that computes a value by successive tensor

contractions directed by its edges (Section 2.2). We say that a series f defined on a family of graphs is *GWM-recognizable* if there exists a GWM M that computes it: we then denote f by f_M.

In the first part of this chapter we study theoretical properties of this computational model. We first show that the notion of GWM extends several models defined on particular families of graphs: this is the case for the classical notion of recognizable series on strings and trees, and also for the more recent model of recognizable picture series (Section 2.3.1). We then investigate to what extent GWMs inherit fundamental properties that are satisfied by the classical notions of recognizable series on strings and trees. We will see that some of these properties are satisfied by GWMs in general, while others are not. When faced which such a property that is not satisfied in general, we will try to identify smaller families of graphs for which this property holds. For example, we show that GWMs satisfy two important closure properties: if f_1 and f_2 are two recognizable series defined on a family $\mathcal{S}$ of connected graphs, then $r+s$ and $r \cdot s$, respectively defined for all graphs $G \in \mathcal{S}$ by $(f_1 + f_2)(G) = f_1(G) + f_2(G)$ and $(f_1 \cdot f_2)(G) = f_1(G)f_2(G)$ (the Hadamard product), are GWM-recognizable. However, GWM-recognizable series defined on general classes of graphs are not closed under scalar multiplication. Nonetheless, we show that GWM-recognizable series are closed under scalar multiplication for any family of rooted graphs (Section 2.3.2). Another fundamental property that we consider is the recognizability of finite support series. Finite support series are series that takes a non zero value on only a finite number of graphs. Finite support series on strings and trees are recognizable but this is not always the case for recognizable series defined on more general families of graphs. For example, we show that finite support series are not recognizable on the family of circular strings. The main reason is that if a recognizable series is not null on some graph G, it must also be different from zero on *coverings* of G, i.e. connected graphs made of copies of G. We show that if a graph family is *covering-free*, then finite support series are recognizable (Section 2.4). Strings and trees, as any family of rooted graphs, are tiling-free. We also show that the fact that GWM can deal with complex syntactical structures may entail compressed representations of computational models: a GWM with n states can compute a function on strings that cannot be computed by a string weighted automaton with less than n^2 states (Section 2.3.3).

The second part of this chapter is devoted to give some insight on the issues that have motivated the present study and to describe some of our perspectives. In machine learning, a classical problem consists in inferring an unknown recognizable series f from examples $(x, f(x))$ of this function. The systematical use of algebraic representations for recognizable series on strings and trees has entailed the development of several successful methods, such as the so-called spectral methods, which try first to estimate several algebraic characteristics (spectrum, eigenspaces, singular vectors, etc) of the underlying operators from learning data, from which the target can be reconstructed. We expect that similar learning schemes for GWM-recognizable graph series can be developed. As an illustration of this general research program, we present a learning algorithm for GWMs defined on the family of circular strings that relies on tensor decomposition techniques (Section 2.5).

Related work. String recognizable series and weighted automata have their roots in automata theory (Eilenberg and Tilson, 1976; Schützenberger, 1961) and their study

can be found in (Berstel and Reutenauer, 1988; Droste, Kuich, and Vogler, 2009; Kuich and Salomaa, 2012; Sakarovitch, 2009; Salomaa and Soittola, 2012). The extension of rational/recognizable series and weighted automaton to trees is presented in (Berstel and Reutenauer, 1982; Droste, Kuich, and Vogler, 2009). A model of recognizable series on 2-dimensional words has been introduced in (Bozapalidis and Grammatikopoulou, 2005). Computational models used to parse and generate graphs have been proposed using the formalism of grammars (see e.g. Rozenberg, 1997). The extension of weighted automata to graphs by mean of weighted logic has been considered in (Droste and Dück, 2015) where the authors propose a quantitative version of Thomas' unweighted model of graph acceptors (Thomas, 1991), and show that this model is expressively equivalent to some suited monadic second order logic. The definability of graph functions in monadic second order logic has also been investigated in the work of Johann A. Makovsky (see e.g. Makowsky and Kotek, 2014). Spectral methods for inference of stochastic languages of strings/trees have been developed upon the notion of linear representation of a rational series (see Bailly, Denis, and Ralaivola (2009), Hsu, Kakade, and T. Zhang (2008), Bailly, Habrard, and Denis (2010), and Balle et al. (2014) for example).

Summary of the contributions. We show how the computations of weighted automata on strings and trees can *be interpreted in terms of tensor networks*, which naturally leads us to define a novel computational model that *naturally extends the notion of recognizable series to labeled graphs: Graph Weighted Models*. We provide a theoretical study of several properties of this model such as *closure properties and the question of the recognizability of finite support series*. While some of these properties are not satisfied by GWMs defined on arbitrary families of graphs, *we identify smaller families of graphs on which these properties hold*. Finally, we study the problem of *learning GWMs defined on the family of circular strings* for which we propose an algorithm relying on tensor decomposition techniques.

The works presented in this chapter have been realized in collaboration with Raphaël Bailly (*Maître de Conférences* at Paris 1 University). The general model of recognizable series on hypergraphs has been presented in the *Journées Montoises d'Informatique Théorique* in September 2014 (25mn talk) and in the international conference LATA (Bailly, Denis, and Rabusseau, 2015) in March 2016 (25mn talk). A journal version is currently under review for a special issue of the *Journal of Computer and System Sciences*.

2.1.1 Recognizable Series over Strings and Trees

In this chapter, unless explicitly stated otherwise, $\mathbb{F}$ is either the field of real numbers $\mathbb{R}$ or the field of complex numbers $\mathbb{C}$.

We first recall the standard notions of recognizable series on strings and trees. Recognizable series are often described in terms of weighted automata but we will use their alternative definition in terms of linear representations. Linear representations allow one to study weighted automata from the perspective of linear and multilinear algebra. In particular, spectral learning methods for weighted automata on strings or trees heavily rely on the algebraic structure of weighted automata. In the following, we formally introduce recognizable series on strings and trees and we present a simple and unifying view of weighted automata by interpreting their computations in terms of tensor net-

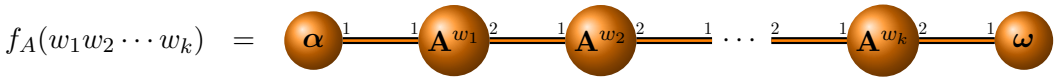


Figure 2.2: Computation of a weighted automaton as a tensor network.

works. While defining automata over graphs is delicate, this analogy between weighted automata on strings/trees and tensor networks can straightforwardly be extended to labeled graphs. This will lead us to define *graph weighted models* in the next section, a computational model that directly generalizes the notion of linear representation to graphs.

Recognizable series on strings. We denote by Σ^* the set of all strings on a finite alphabet Σ and by Σ^+ the set of all non-empty strings on Σ. The empty word is denoted by ε and we write $|x|$ to denote the length of a string $x \in \Sigma^\star$.

A *weighted (string) automaton* (WA) is a tuple $A = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ where n is the number of states (or dimension) of the automata, $\boldsymbol{\alpha}, \boldsymbol{\omega} \in \mathbb{F}^n$ are the initial and terminal weight vectors, and $\mathbf{A}^\sigma$ is an $n \times n$ transition matrix for each $\sigma \in \Sigma$. A weighted automaton computes a function $f_A : \Sigma^* \to \mathbb{F}$ defined by

$$f_A(w) = \boldsymbol{\alpha}^\top \mathbf{A}^{w_1} \mathbf{A}^{w_2} \cdots \mathbf{A}^{w_k} \boldsymbol{\omega}$$

for any word $w = w_1 w_2 \cdots w_k \in \Sigma^*$. The *rank* of a function $f : \Sigma^* \to \mathbb{F}$ is the minimal number of states of a WA computing it, if f is not recognizable (i.e. it cannot be computed by a WA) we set $\mathrm{rank}(f) = \infty$. An automaton is *minimal* if its number of states is equal to the rank of the function its compute. We refer the reader to the books of Berstel and Reutenauer (1988), Sakarovitch (2009) and Droste, Kuich, and Vogler (2009) for more details.

The tensor network shown in Figure 2.2 represents the computation of a WA and shows how the linear structure of a string is naturally translated in the WA computation.

Hidden Markov Models (HMM) are widely used statistical tools to modelize discrete time series (Baum and Eagon, 1967; Rabiner, 1989) and it is worth mentioning that weighted automata on strings encompass HMMs: any probability distribution on Σ^* defined by an HMM can be computed by a weighted automaton.

Recognizable series on trees. We first define (ranked) trees on a ranked alphabet. A *ranked alphabet* is a tuple $\mathcal{F} = (\Sigma, \sharp)$ where Σ is a *finite* alphabet and $\sharp : \Sigma \to \mathbb{N}$ is an arity function. We denote by $\mathcal{F}_p = \{g \in \Sigma : \sharp g = p\}$ the set of symbols with arity p. Similarly, we will denote by $\mathcal{F}_{\leq p}$ (resp. $\mathcal{F}_{\geq p}$) the set of symbols with arity at most p (resp. at least p). We will sometimes use parenthesis and commas for a short declaration of symbols with arity, e.g. $g(\cdot, \cdot)$ is a symbol of arity 2.

The set of trees $\mathfrak{T}_\mathcal{F}$ on a ranked alphabet $\mathcal{F}$ is the smallest set such that

- $\sigma \in \mathfrak{T}_\mathcal{F}$ for any $\sigma \in \mathcal{F}_0$,
- $g(t_1, \cdots, t_k) \in \mathfrak{T}_\mathcal{F}$ for any $k \geq 1$, $g \in \mathcal{F}_k$ and $t_1, \cdots, t_k \in \mathfrak{T}_\mathcal{F}$.

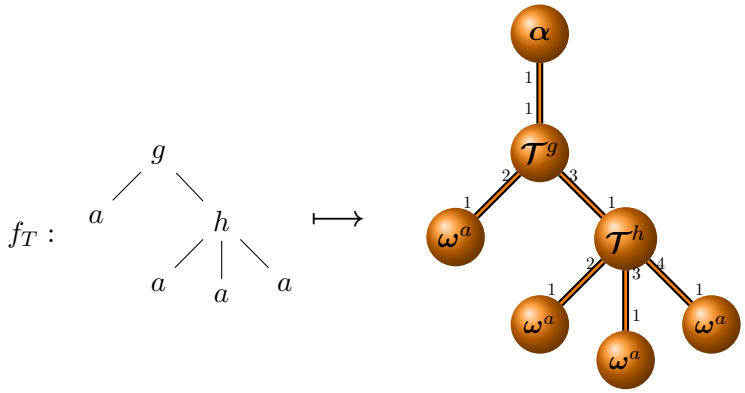


Figure 2.3: Computation of a weighted tree automaton on the tree $g(a, h(a, a, a))$ as a tensor network.

Here $g(t_1, \cdots, t_k)$ denotes the tree with root labeled by the symbol g and having k sons $t_1, \cdots, t_k$. We will call symbols in $\mathcal{F}_0$ *leaf symbols* and symbols in $\mathcal{F}_{\geq 1}$ *internal symbols*.

A *weighted tree automaton* (WTA) is a tuple $T = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{T}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$ where n is the number of states (or dimension), $\boldsymbol{\alpha} \in \mathbb{F}^n$ is the initial weight vector, $\boldsymbol{\omega}^\sigma \in \mathbb{F}^n$ is the final weight vector associated with the leaf symbol σ for each $\sigma \in \mathcal{F}_0$, and for any symbol g of arity $k \geq 1$, $\boldsymbol{\mathcal{T}}^g \in (\mathbb{F}^n)^{\otimes k+1}$ is the transition tensor associated with the internal symbol g.

A WTA computes a function $f_T : \mathfrak{T}_\mathcal{F} \to \mathbb{F}$ defined by $f_T(t) = \boldsymbol{\alpha}^\top \boldsymbol{\omega}_T(t)$ where the mapping $\boldsymbol{\omega}_T : \mathfrak{T}_\mathcal{F} \to \mathbb{F}^n$ is inductively defined by

- $\boldsymbol{\omega}_T(\sigma) = \boldsymbol{\omega}^\sigma$ for any $\sigma \in \mathcal{F}_0$,
- $\boldsymbol{\omega}_T(g(t_1, \cdots, t_k)) = \boldsymbol{\mathcal{T}}^g \bullet_2 \boldsymbol{\omega}_T(t_1) \bullet_3 \cdots \bullet_{p+1} \boldsymbol{\omega}_T(t_p)$ for any $p \geq 1$, $g \in \mathcal{F}_p$, and trees $t_1, \cdots, t_p \in \mathfrak{T}_\mathcal{F}$.

In many cases we will just write $\boldsymbol{\omega}(t)$ when the automaton T is clear from context. Similarly to the string case, the *rank* of a function $f : \mathfrak{T}_\mathcal{F} \to \mathbb{F}$ is the minimal number of states of a WTA computing it and we let $\mathrm{rank}(f) = \infty$ if f is not recognizable, and a WTA is *minimal* if its number of states is equal to the rank of the function its compute. We refer the reader to the books (Comon et al., 2007) and (Droste, Kuich, and Vogler, 2009) for more details on (weighted) tree automata.

The tensor network shown in Figure 2.3 represents the computation of a WTA on a tree on the ranked alphabet $\mathcal{F} = \{a, g(\cdot, \cdot), h(\cdot, \cdot, \cdot)\}$. It illustrates how the computation of a WTA on a given tree naturally reflects the tree's structure.

Recognizable series and tensor networks. Figures 2.2 and 2.3 show how the computation of a weighted automaton on a string or a tree can naturally be interpreted as a tensor network (this interpretation directly comes from the algebraic nature of linear representations of weighted automata). A weighted automaton can thus simply be seen as a mapping that associates each symbol in a (ranked) alphabet to a tensor, and its computation can be summarized as follows: given a syntactical structure (e.g. a string or a tree), construct the tensor product of all the tensors associated with the *atoms* of

the structure (letters for strings or symbols for trees), and apply successive contractions corresponding to links between the atoms (successive letters in a string or father and child nodes in a tree). Of course once this has been done, one is left with a matrix in the case of strings or a vector in the case of trees; the final step in the computation consists in *plugging* vectors in the remaining open edges of the tensor network using $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ for strings or solely $\boldsymbol{\alpha}$ for trees.

It should now be clear that this unifying view of weighted automata computations on strings and trees in terms of tensor networks can directly be extended to *labeled graphs*. This chapter is devoted to the introduction and the study of this computational model on graphs, which we call *Graph Weighted Models*.

As a side note, we mentioned in Chapter 1 that the computation of a weighted automaton on strings is closely related to the tensor train (TT) decomposition; we can now briefly explicit this relation. Let $A = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ be a weighted automaton on strings. For any integer k, let us define the kth order tensor $\boldsymbol{\mathcal{H}} \in (\mathbb{F}^\Sigma)^{\otimes k}$ by

$$\boldsymbol{\mathcal{H}}_{\sigma_1,\sigma_2,\cdots,\sigma_k} = f_A(\sigma_1\sigma_2\cdots\sigma_k) \quad \text{for all } \sigma_1,\sigma_2,\cdots,\sigma_k \in \Sigma.$$

The components of this tensor are all the values computed by A on strings of length k. By letting $\boldsymbol{\mathcal{G}} \in \mathbb{F}^{n\times\Sigma\times n}$ be the 3rd order tensor defined by $\boldsymbol{\mathcal{G}}_{:,\sigma,:} = \mathbf{A}^\sigma$ for any $\sigma \in \Sigma$, one can check that the tensor $\boldsymbol{\mathcal{H}}$ can be decomposed in the following form which is similar to a TT decomposition:

$$\boldsymbol{\mathcal{H}}_{\sigma_1,\sigma_2,\cdots,\sigma_k} = \sum_{i_0,i_1,\cdots,i_k} \boldsymbol{\alpha}_{i_0}\boldsymbol{\mathcal{G}}_{i_0,\sigma_1,i_1}\boldsymbol{\mathcal{G}}_{i_1,\sigma_2,i_2}\cdots\boldsymbol{\mathcal{G}}_{i_{k-1},\sigma_k,i_k}\boldsymbol{\omega}_{i_k}.$$

This relationship has been previously noticed in (Critch and Morton, 2014; Critch, 2013) where the authors show that hidden Markov models bear a close relationship to Matrix Product States (MPS) with open boundary conditions (which are similar to decompositions in the TT format). Similarly, GWMs defined over the family of circular strings are related to MPS with periodic boundary conditions (which are similar to tensor ring decompositions), while GWMs defined over the family of 2-dimensional words are related to another tensor network model used in quantum physics known as Projected Entangled Pair States (which are, roughly speaking, tensor networks having a 2-dimensional grid structure).

2.2 A Model of Recognizable Series over Graphs

2.2.1 Graphs

We start by introducing a definition of graphs on a ranked alphabet. This definition is very closely related to the notion of tensor networks that we introduced in Section 1.2.2. In particular, we introduce the notion of *ports* of a vertex that allows us to distinguish the different outgoing edges from a node in a graph. Moreover, each vertex of a graph is labeled by a symbol of the ranked alphabet, and its degree (i.e. the number of outgoing edges from the vertex) must coincide with the arity of the symbol it is labeled with. We also allow *open edges* which are outgoing edges from some vertex that are not connected to any other vertex in the graph. We will show in Section 2.2.2 how classical objects

such as strings and trees can be defined using this formalism.

Definition 1. *A* graph G *over a ranked alphabet* $\mathcal{F} = (\Sigma, \sharp)$ *is a tuple* (V, E, ℓ) *where*

- V *is a non empty* set of vertices,
- $\ell : V \to \Sigma$ *is a* labeling *of the vertices,*
- E *is a partition composed of sets of cardinality 1 or 2 of the set of ports*

$$P(G) = \{(v, j) : v \in V, j \in [\sharp v]\}$$

of G *(where* $\sharp v = \sharp \ell(v)$*).*

The partition E of the ports of a graph G is the *set of edges* of the graph. The elements of E with cardinality 2 correspond to the classical notion of edges of a graph (i.e. a connection between two ports of the graph), while elements with cardinality 1 correspond to *free ports* (or open edges) of the graph (i.e. ports that are not connected to any other port in the graph). Observe that the fact that the degree of each vertex must coincide with the arity of its label is a direct consequence of Definition 1 (since E is a partition of $P(G)$). Two examples of graphs are shown in Figure 2.4.

We will denote by $\mathfrak{G}_{\mathcal{F}}$ the *set of graphs on the ranked alphabet* $\mathcal{F}$. For any integer k we denote by $\mathfrak{G}_{\mathcal{F}}^k$ the *set of graphs with* k *free ports*:

$$\mathfrak{G}_{\mathcal{F}}^k = \{G = (V, E, \ell) \in \mathfrak{G}_{\mathcal{F}} : |\{e \in E : |e| = 1\}| = k\}.$$

We will say that two graphs $G = (V, E, \ell)$ and $G' = (V', E', \ell')$ on the same ranked alphabet $\mathcal{F}$ (i.e. $G, G' \in \mathfrak{G}_{\mathcal{F}}$) are *isomorphic* if there exists a bijection $\psi : V \to V'$ such that

- $\ell(v) = \ell'(\psi(v))$ for all $v \in V$ and
- for all $\{(u, i), (v, j)\} \in E$ we have $\{(\psi(u), i), (\psi(v), j)\} \in E'$.

Note that we did not assume that $u \neq v$ and $i \neq j$ in the second condition, thus this condition must also be satisfied by open edges of the graphs. Loosely speaking, two graphs are isomorphic if one can be obtained from the other by simply renaming its vertices. In this chapter, we will mostly consider that *two isomorphic graphs are the same object*. If for example we say that a property holds for all graph except for some graph G, we mean that the property is true for any graph that is not isomorphic to G.

We will say that a graph G is *connected* if there exists a path between any two vertices of G, i.e. for all $u, v \in V$ there exists a sequence of edges $e_1, \cdots, e_m \in E$ such that $(u, i) \in e_1, (v, j) \in e_m$ for some $i \in [\sharp u], j \in [\sharp v]$ and $e_k \cap e_{k+1} \neq \emptyset$ for all $k \in [m-1]$.

We will call a graph with no free port $G \in \mathfrak{G}_{\mathcal{F}}^0$ a *closed graph*. For any symbol $x \in \Sigma$ of arity k, we denote by $G^x \in \mathfrak{G}_{\mathcal{F}}^k$ the *singleton graph* with one vertex labeled by x and k free ports: $G^x = (\{v\}, \{\{(v, i)\} : i \in [k]\}, \ell)$ with $\ell(v) = x$. We will call (v, i) the ith free port of G^x.

Example 1. *Let* $\mathcal{F} = (\{h, g, a\}, \sharp)$ *be a ranked alphabet with* $\sharp h = 2$, $\sharp g = 3$ *and* $\sharp a = 1$. *An example of a graph* $G = (V, E, \ell) \in \mathfrak{G}_{\mathcal{F}}^0$ *is shown in Figure 2.4 (left) where*

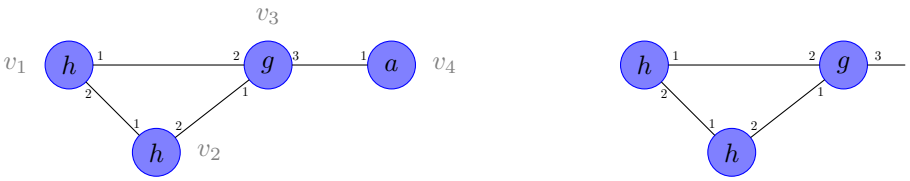


Figure 2.4: The graphs $G \in \mathfrak{G}^0_{\mathcal{F}}$ and $G' \in \mathfrak{G}^1_{\mathcal{F}}$ from Example 1.

- $V = \{v_1, v_2, v_3, v_4\}$,
- $\ell(v_1) = \ell(v_2) = h$, $\ell(v_3) = g$, $\ell(v_4) = a$, *and*
- $E = \{\{(v_1, 1), (v_3, 2)\}, \{(v_1, 2), (v_2, 1)\}, \{(v_2, 2), (v_3, 1)\}, \{(v_3, 3), (v_4, 1)\}\}$.

The graph with one free port $G' = (V \setminus \{v_4\}, (E \cup \{\{(v_3, 3)\}\}) \setminus \{\{(v_3, 3), (v_4, 1)\}\}, \ell) \in \mathfrak{G}^1_{\mathcal{F}}$ *obtained by removing the vertex* v_4 *from* G *is shown in Figure 2.4 (right).*

Let $\mathcal{F} = (\Sigma, \sharp)$ be a ranked alphabet. The set of graphs $\mathfrak{G}_{\mathcal{F}}$ can also be defined inductively using the following two simple operations on graphs:

- Let $G_1 = (V_1, E_1, \ell_1) \in \mathfrak{G}^{k_1}_{\mathcal{F}}$ and $G_2 = (V_2, E_2, \ell_2) \in \mathfrak{G}^{k_2}_{\mathcal{F}}$. We assume that G_1 and G_2 have distinct sets of vertices ($V_1 \cap V_2 = \emptyset$), if this is not the case it suffices to rename the elements of V_1. We denote by $G_1 \cup G_2 = (V, E, \ell) \in \mathfrak{G}^{k_1+k_2}_{\mathcal{F}}$ the *union* of G_1 and G_2, where $V = V_1 \cup V_2$, $E = E_1 \cup E_2$ and $\ell(v) = \ell_1(v)$ if $v \in V_1$ and $\ell_2(v)$ otherwise.
- Let $G = (V, E, \ell) \in \mathfrak{G}^k_{\mathcal{F}}$ for some $k \geq 2$ and let $p, p' \in P(G)$ be two distinct free ports of G (that is $p \neq p'$ and $\{p\}, \{p'\} \in E$). We denote by $\langle\!\langle G \rangle\!\rangle_{(p,p')} \in \mathfrak{G}^{k-2}_{\mathcal{F}}$ the graph obtained by connecting p and p' in G. Formally,

$$\langle\!\langle G \rangle\!\rangle_{(p,p')} = (V, (E \cup \{\{p, p'\}\}) \setminus \{\{p\}, \{p'\}\}, \ell).$$

It is then easy to check that $\mathfrak{G}_{\mathcal{F}}$ is the smallest set such that

- For all $x \in \Sigma$, the singleton graph G^x is in $\mathfrak{G}_{\mathcal{F}}$.
- $G_1 \cup G_2 \in \mathfrak{G}_{\mathcal{F}}$ for any $G_1, G_2 \in \mathfrak{G}_{\mathcal{F}}$.
- $\langle\!\langle G \rangle\!\rangle_{(p,p')} \in \mathfrak{G}_{\mathcal{F}}$ for any $G \in \mathfrak{G}_{\mathcal{F}}$ such that $p, p' \in P(G)$, $p \neq p'$ and $\{p\}, \{p'\} \in E$.

2.2.2 Some Families of Graphs

We now define several families of graphs. We first show how strings and trees can be naturally mapped to closed graphs (see Figure 2.5).

Strings. Let Σ be a finite alphabet and let $w = \sigma_1 \sigma_2 \cdots \sigma_k \in \Sigma^*$. We define the ranked alphabet $\mathcal{F} = (\Sigma \cup \{\alpha, \omega\}, \sharp)$ where α and ω are two new symbols of arity 1 and $\sharp\sigma = 2$ for all $\sigma \in \Sigma$. The graph $graph(w)$ is obtained by mapping each letter in w to a vertex whose first (resp. second) port is connected to the previous (resp. next)

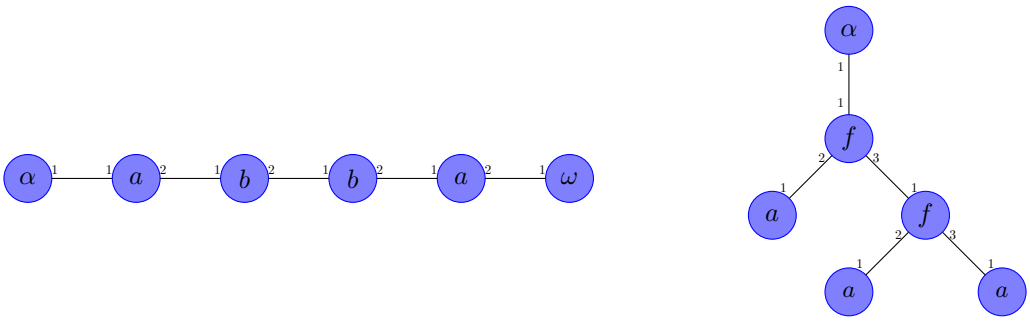


Figure 2.5: (left) Graph $graph(w)$ associated with the string $w = abba$. (right) Graph $graph(t)$ associated with the tree $t = f(a, f(a, a))$.

vertex and the special symbols α and ω are used to mark the start and end of the string. Formally, $graph(w) = (V, E, \ell)$ where $V = \{v_0, v_1, \cdots, v_{k+1}\}$, $\ell(v_0) = \alpha$, $\ell(v_{k+1}) = \omega$ and $\ell(v_i) = \sigma_i$ for $i \in [k]$, and $E = \{\{(v_i, 2), (v_{i+1}, 1)\} : i \in [k]\} \cup \{\{(v_0, 1), (v_1, 1)\}\}$.

Trees. We consider trees on a ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$. Let $\mathcal{F}' = (\Sigma \cup \{\alpha\}, \sharp')$ where α is a new symbol of arity 1 and $\sharp' g = \sharp g + 1$ for all $g \in \Sigma$. We first map any tree $t \in \mathfrak{T}_{\mathcal{F}}$ to a graph $\tilde{G}^t \in \mathfrak{G}^1_{\mathcal{F}'}$ where the free port in $\tilde{G}^t$ marks the root of the tree. This mapping is defined inductively:

- for any leaf symbol $\sigma \in \mathcal{F}_0$ we let $\tilde{G}^\sigma$ be the singleton graph G^σ
- for any tree $t = g(t_1, \cdots, t_k)$ in $\mathfrak{T}_{\mathcal{F}}$ where $g \in \mathcal{F}_k$ and $t_1, \cdots, t_k \in \mathfrak{T}_{\mathcal{F}}$ we let

$$\tilde{G}^t = \langle\!\langle G^g \cup \tilde{G}^{t_1} \cup \cdots \cup \tilde{G}^{t_k} \rangle\!\rangle_{(q_2,p_1),(q_3,p_2),\cdots,(q_{k+1},p_k)}$$

  where q_i is the ith free port of G^g and p_j is the (only) free port of $\tilde{G}^{t_j}$.

The graph $graph(t)$ is then obtained by connecting a root vertex to $\tilde{G}^t$: $graph(t) = \langle\!\langle G^\alpha \cup \tilde{G}^t \rangle\!\rangle_{(p,p')}$ where p (resp. p') is the free port of G^α (resp. $\tilde{G}^t$).

The set of graphs on a ranked alphabet is a very rich and complex family of structured objects. Consequently, we will see in the next sections that some fundamental properties of recognizable series over strings and trees are not verified by recognizable series over graphs. One objective of this chapter will be to identify families of graphs on which graph recognizable series satisfy these fundamental properties. Such a family of graph should be smaller than the whole set of graphs $\mathfrak{G}_{\mathcal{F}}$ while still being more general than the families of strings and trees. One way to proceed is by enforcing some structural constraint on the graphs belonging to such a family. We will consider two such structural constraints (that are satisfied by graphs obtained from strings and trees): graphs with a root, and graphs for which ports can be partitioned into input and output ports.

Rooted graphs. Let $\mathcal{F} = (\Sigma \cup \{\alpha_0\}, \sharp)$ be a ranked alphabet. We will say that a set of graphs $\mathcal{S} \subset \mathfrak{G}^0_{\mathcal{F}}$ is a family of *rooted graphs* if the special root symbol α_0 appears exactly once in each graph of $\mathcal{S}$:

$$\forall G = (V, E, \ell) \in \mathcal{S}, \quad |\{v \in V : \ell(v) = \alpha_0\}| = 1.$$

Note that the families of graphs obtained from strings and trees are families of rooted graphs.

In-out graphs. Let $\mathcal{F} = (\Sigma, \sharp)$ be a ranked alphabet. The notion of in-out graphs is related to the classical notion of directed graphs. Loosely speaking, we will say that a set of graphs $\mathcal{S} \subset \mathfrak{G}^0_{\mathcal{F}}$ is a family of *in-out graphs* if for each symbol in Σ of arity k, the set $[k]$ can be partitioned in *in-ports* and *out-ports* such that any graph in $\mathcal{S}$ does not contain any edge with two in-ports or two out-ports. Formally, $\mathcal{S}$ is a family of *in-out graphs* if and only if there exist a mapping $\zeta : \{(x, i) : x \in \Sigma, i \in [\sharp x]\} \to \{in, out\}$ such that

$$\forall G = (V, E, \ell) \in \mathcal{S} : \{(u, i), (v, j)\} \in E \text{ and } (u, i) \neq (v, j) \Rightarrow \zeta(\ell(u), i) \neq \zeta(\ell(v), j).$$

Again, the family of graphs obtained from strings and trees are families of in-out graphs, and so are the families of circular strings and 2d-words defined below.

Finally, we present two families of graphs that extend the traditional notion of string over a finite alphabet. Circular strings are strings that are closed onto themselves, and two-dimensional words naturally extend (one-dimensional) strings to arrays of symbols (see Figure 2.6).

Circular strings. Let Σ be an alphabet and let $\sharp$ be the arity function defined by $\sharp\sigma = 2$ for all $\sigma \in \Sigma$.

Let $w = w_1 w_2 \cdots w_k \in \Sigma^+$ be a word of length k. The *circular string* $circ(w)$ is the graph $G = (V, E, \ell) \in \mathfrak{G}_{(\Sigma,\sharp)}$ defined by:

- $V = \{v_1, \cdots, v_k\}$,
- $\ell(v_i) = w_i$ for $i \in [k]$ and
- $E = \{\{(v_1, 2), (v_2, 1)\}, \{(v_2, 2), (v_3, 1)\}, \cdots, \{(v_{k-1}, 2), (v_k, 1)\}, \{(v_k, 2), (v_1, 1)\}\}$.

We denote by $\mathfrak{G}^{circ}_{\Sigma}$ the set of all circular strings over Σ. The family of circular strings will be of particular interest in this chapter since it is in some sense the smallest family of graphs with loops.

Two-dimensional words. A *2d-word* (or picture) $p \in \Sigma^{++}$ over a finite alphabet Σ is defined as a non-empty rectangular array of elements of Σ, formally $\Sigma^{++} = \cup_{m,n\geq 1}\Sigma^{m\times n}$. We write $p_{i,j}$ for the component of p at position (i, j). With each 2d-word $p \in \Sigma^{++}$ of size $m \times n$ we associate a closed graph $graph(p) = (V, E, \ell)$ on the ranked alphabet $\mathcal{F} = (\Sigma \cup \{\mathtt{w}, \mathtt{n}, \mathtt{e}, \mathtt{s}\}, \sharp)$ where $\sharp\sigma = 4$ for all $\sigma \in \Sigma$ and $\sharp\mathtt{w} = \sharp\mathtt{e} = \sharp\mathtt{n} = \sharp\mathtt{s} = 1$. The graph $graph(p)$ is constructed in a straightforward way by translating p into a graph and adding nodes to the border of the picture: nodes on the west border will be labeled by the symbol `w`, on the east border by `e`, etc. Formally, we have

- $V = \{v_{i,j}, \bar{w}_i, \bar{e}_i, \bar{n}_j, \bar{s}_j : i \in [m], j \in [n]\}$
- $\ell(v_{i,j}) = p_{i,j}$, $\ell(\bar{w}_i) = \mathtt{w}$, $\ell(\bar{e}_i) = \mathtt{e}$, $\ell(\bar{n}_i) = \mathtt{n}$, $\ell(\bar{s}_i) = \mathtt{s}$ for $i \in [m], j \in [n]$
- E contains the following edges:

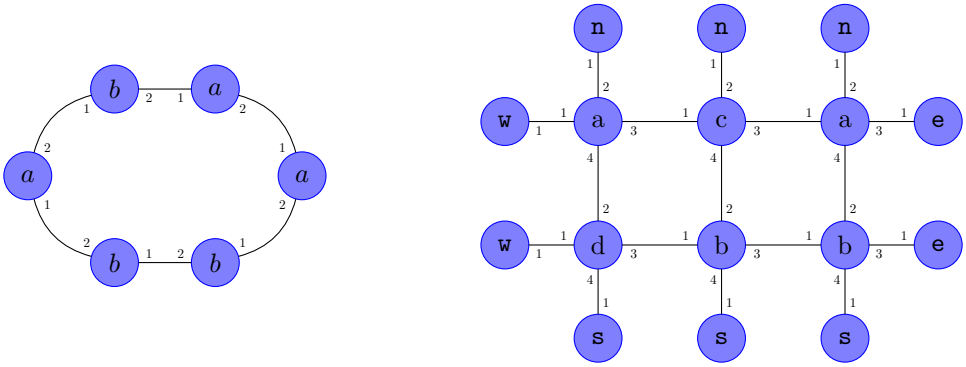


Figure 2.6: (left) The circular string $circ(abaabb)$. (right) Graph $graph(p)$ associated with the picture $\begin{smallmatrix} aca \\ dbb \end{smallmatrix}$.

- $\{(v_{i,j}, 3), (v_{i,j+1}, 1)\}$ for $i \in [m]$, $1 \leq j < n$,
- $\{(v_{i,j}, 4), (v_{i+1,j}, 2)\}$ for $1 \leq i < m$, $j \in [n]$,
- $\{(v_{i,1}, 1), (\bar{w}_i, 1)\}$ and $\{(v_{i,n}, 3), (\bar{e}_i, 1)\}$ for $i \in [m]$,
- $\{(v_{1,j}, 2), (\bar{n}_j, 1)\}$ and $\{(v_{m,j}, 4), (\bar{s}_j, 1)\}$ for $j \in [n]$.

2.2.3 Graph Weighted Models

We now formally introduce graph weighted models. A graph weighted model is a computational model that assigns a value to any closed graph on a given ranked alphabet. It is defined by a set of tensors, one for each symbol in the alphabet, and it naturally associates a graph with a tensor network computing the corresponding value. The simple idea behind this model comes from the analogy between recognizable series on strings/trees and tensor networks that has been presented in Section 2.1.1.

Definition 2. *Let $\mathbb{F}$ be either $\mathbb{R}$ or $\mathbb{C}$. A* graph weighted model *(GWM) with coefficients in $\mathbb{F}$ on a ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$ is a tuple $(\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma})$ where*

- *n is the dimension (or number of states) and*
- *for each $x \in \Sigma$, $\boldsymbol{\mathcal{M}}^x \in (\mathbb{F}^n)^{\otimes \sharp x}$ is a tensor of order the arity of x.*

Given a GWM $M = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma})$ and a graph $G = (V, E, \ell)$, we denote by $\Gamma_{id}(M, G)$ the set of mappings from $P(G)$ to $[n]$ defined by

$$\Gamma_{id}(M, G) = \{\gamma : P(G) \to [n] \mid \{p, p'\} \in E \Rightarrow \gamma(p) = \gamma(p')\}.$$

Observe that any mapping $\gamma : P(G) \to [n]$ assigns a state to each port in the graph and that $\Gamma_{id}(M, G)$ is the set of all assignments that are compatible with the graph structure.

The GWM M computes a function $f_M : \mathfrak{G}^0_{\mathcal{F}} \to \mathbb{F}$ mapping closed graphs to scalars defined by

$$f_M(G) = \sum_{\gamma \in \Gamma_{id}(M,G)} \prod_{v \in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1), \cdots, \gamma(v, \sharp v)} \quad \text{for all } G \in \mathfrak{G}^0_{\mathcal{F}}.$$

We will say that a function $f : \mathfrak{G}^0_{\mathcal{F}} \to \mathbb{F}$ *is* GWM-recognizable *if there exists a GWM* M *that computes it (i.e.* $f(G) = f_M(G)$ *for any graph* G*).*

Using the notations from the previous definition, let $V = (v_1, \cdots, v_k)$. The tensor

$$\boldsymbol{\mathcal{M}}^{\circ} = \boldsymbol{\mathcal{M}}^{\ell(v_1)} \circ \boldsymbol{\mathcal{M}}^{\ell(v_2)} \circ \cdots \circ \boldsymbol{\mathcal{M}}^{\ell(v_k)}$$

is of order $|P(G)|$ and any $\gamma \in \Gamma_{id}(M, G)$ is an element of $[n]^{P(G)}$. Thus, γ can be seen as a multi-index and $\prod_{v \in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$ is the $(\gamma(v_1, 1), \cdots, \gamma(v_1, \sharp v_1), \cdots, \gamma(v_k, 1), \cdots, \gamma(v_k, \sharp v_k))$-component of the tensor $\boldsymbol{\mathcal{M}}^{\circ}$. Furthermore, since $\Gamma_{id}(G, M)$ is isomorphic to $[n]^E$, the computation of M on a graph $G = (V, E, \ell)$ can be interpreted in the following way:

- Each component $\boldsymbol{\mathcal{M}}^x_{i_1,\cdots,i_{\sharp x}}$ of a tensor $\boldsymbol{\mathcal{M}}^x$ represents the *weight* of a vertex labeled by x when its first port is in state i_1, its second port in state i_2, etc.
- Each *configuration* in $[n]^E$ (i.e. a labeling of the edges of G with states in $[n]$) assigns a state to each port of the graph, thus a configuration assigns a weight to each vertex of G (using the tensors $\boldsymbol{\mathcal{M}}^x$). The product of these weights represents the weight of the configuration.
- The value computed by M is the sum of the weights of all possible configurations in $[n]^E$.

The fact that $\gamma \in \Gamma_{id}(G, M)$ implies that if two ports p and p' are connected in G, then the corresponding indices of the tensor $\boldsymbol{\mathcal{M}}^{\circ}$ will be contracted during the GWM computation. In fact, as shown in Example 2 below, the GWM computation boils down to the tensor network naturally constructed from the graph G and the tensors $\{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma}$.

Following this analogy between tensor networks and GWMs, any graph $G \in \mathfrak{G}^k_{\mathcal{F}}$ with k free ports can be associated with a kth order tensor in $(\mathbb{F}^n)^{\otimes k}$. Formally, given a GWM $M = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma})$ we define the mapping $\mu_M : \mathfrak{G}^k_{\mathcal{F}} \to (\mathbb{F}^n)^{\otimes k}$ for any $k \geq 0$ inductively by

- $\mu_M(G^x) = \boldsymbol{\mathcal{M}}^x$ for any singleton graph G^x and $x \in \Sigma$,
- $\mu_M(G_1 \cup G_2) = \mu_M(G_1) \circ \mu_M(G_2)$ for any $G_1, G_2 \in \mathfrak{G}_{\mathcal{F}}$,
- $\mu_M(\langle\!\langle G \rangle\!\rangle_{(p,p')}) = \langle\!\langle \mu_M(G) \rangle\!\rangle_{(i,i')}$ for any graph $G \in \mathfrak{G}_{\mathcal{F}}$ with two distinct free ports p and p', where i (resp. i') is the mode of the tensor $\mu_M(G)$ corresponding to the port p (resp. p').

Note that identifying $(\mathbb{F}^n)^{\otimes 0}$ with $\mathbb{F}$ we have $f_M(G) = \mu_M(G)$ for any $G \in \mathfrak{G}^0_{\mathcal{F}}$. In the following example we detail the computations of a GWM on the graphs from Example 1 and we show how these computations can naturally be expressed with tensor networks.

Example 2. *Let* $M = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma})$ *be a GWM. The value computed by* M *on the graph* $G \in \mathfrak{G}^0_{\mathcal{F}}$ *from Example 1 (page 29) is given by*

$$f_M(G) = \sum_{i_1,i_2,i_3,i_4 \in [n]} \boldsymbol{\mathcal{M}}^h_{i_1,i_2} \boldsymbol{\mathcal{M}}^h_{i_2,i_3} \boldsymbol{\mathcal{M}}^g_{i_3,i_1,i_4} \boldsymbol{\mathcal{M}}^a_{i_4}.$$

The summation over i_1 corresponds to the edge between v_1 and v_3, the summation over i_2 to the edge between v_1 and v_2, etc. Using the contraction operator we can write

$$f_M(G) = \langle\langle \boldsymbol{\mathcal{M}}^h \circ \boldsymbol{\mathcal{M}}^h \circ \boldsymbol{\mathcal{M}}^g \circ \boldsymbol{\mathcal{M}}^a \rangle\rangle_{(1,6),(2,3),(4,5),(7,8)}.$$

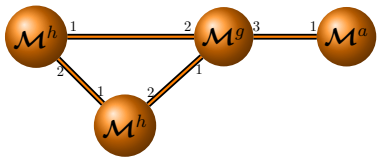
The mapping μ_M maps the graph $G' \in \mathfrak{G}^1_{\mathcal{F}}$ from Example 1 to the vector $\mathbf{v} \in \mathbb{F}^n$ defined by

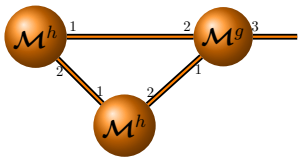
$$\mathbf{v}_j = \sum_{i_1,i_2,i_3 \in [n]} \boldsymbol{\mathcal{M}}^h_{i_1,i_2} \boldsymbol{\mathcal{M}}^h_{i_2,i_3} \boldsymbol{\mathcal{M}}^g_{i_3,i_1,j}$$

for all $j \in [n]$. Using the contraction operator we have

$$\mu_M(G') = \langle\langle \boldsymbol{\mathcal{M}}^h \circ \boldsymbol{\mathcal{M}}^h \circ \boldsymbol{\mathcal{M}}^g \rangle\rangle_{(1,6),(2,3),(4,5)}.$$

These computations of the GWM M naturally correspond to the following tensor networks:

$f_M(G) =$ 

$\mu_M(G') =$ 

The following example shows that the computation of a GWM on a circular string can be simply expressed in terms of the trace of the product of the matrices associated with the symbols in the string.

Example 3. *Let $M = (\mathbb{F}^n, \{\mathbf{M}^\sigma\}_{\sigma \in \Sigma})$ be a GWM defined on the family of circular strings on a finite alphabet Σ. It can easily be checked that M computes the function*

$$circ(w) \mapsto \mathrm{Tr}(\mathbf{M}^{w_1} \mathbf{M}^{w_2} \cdots \mathbf{M}^{w_k})$$

for any string $w = w_1 w_2 \cdots w_k \in \Sigma^+$.

Observe that the invariance of the trace of a product of matrices under cyclic permutations nicely translates the fact that two circular strings obtained by cyclic permutation (e.g. $circ(w_1 w_2 \cdots w_k)$ and $circ(w_k w_1 \cdots w_{k-1})$) are the same object (strictly speaking they are isomorphic graphs).

We now define a slightly richer model than GWMs: $\diamond$-GWMs. We introduce this model for two main reasons. First, it will help us simplify the proof of Theorem 2 which is one of the main results of this chapter. Second, this model is closer to the computational model on hypergraphs that we originally considered in our paper (Bailly, Denis, and Rabusseau, 2015).

Definition 3. *A $\diamond$-GWM on a ranked alphabet $(\Sigma, \sharp)$ is a tuple $(\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma}, \diamond)$ where*

- *n is the dimension (or number of states),*
- *for each $x \in \Sigma$, $\boldsymbol{\mathcal{M}}^x \in (\mathbb{F}^n)^{\otimes \sharp x}$ is a tensor of order the arity of x and*
- *$\diamond : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ is a symmetric bilinear form.*

Let $(\mathbf{e}_1, \cdots, \mathbf{e}_n)$ be the canonical basis of $\mathbb{F}^n$. A GWM $M = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma}, \diamond)$ computes a function $f_M : \mathfrak{G}^0_{(\Sigma,\sharp)} \to \mathbb{F}$ defined by

$$f_M(G) = \sum_{\gamma\in[n]^{P(G)}} \boldsymbol{\mathcal{M}}_\gamma \prod_{\{p,p'\}\in E} (\mathbf{e}_{\gamma(p)} \diamond \mathbf{e}_{\gamma(p')}) \quad \textit{for all } G \in \mathfrak{G}^0_{(\Sigma,\sharp)},$$

where $\boldsymbol{\mathcal{M}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$.

It is easy to check that any GWM-recognizable series can be computed by a $\diamond$-GWM: it suffices to define the bilinear form $\diamond$ by $\mathbf{e}_i \diamond \mathbf{e}_j = 1$ if $i = j$ and 0 otherwise. However, we will see below that while the converse is true for GWMs with complex coefficients, it is not the case anymore for GWMs with coefficients in $\mathbb{R}$. We first show in the following proposition that any $\diamond$-GWM can be transformed into an equivalent GWM by applying a (complex) linear transformation along each mode of the tensors of the $\diamond$-GWM.

Proposition 1. *Any graph series recognizable by a $\diamond$-GWM is recognizable by a GWM* with coefficients in $\mathbb{C}$.

More precisely, given a $\diamond$-GWM $A = (\mathbb{F}^n, \{\boldsymbol{\mathcal{A}}^x\}_{x\in\Sigma}, \diamond)$ where $\mathbb{F}$ is either $\mathbb{R}$ or $\mathbb{C}$, there exists a matrix $\mathbf{Q} \in \mathbb{C}^{n\times n}$ such that the GWM $B = (\mathbb{C}^n, \{\boldsymbol{\mathcal{A}}^x \times_1 \mathbf{Q} \times_2 \cdots \times_{\sharp x} \mathbf{Q}\}_{x\in\Sigma})$ computes the same function as A.

Proof. Let $\mathbb{F}$ be either $\mathbb{R}$ or $\mathbb{C}$ and let $A = (\mathbb{F}^n, \{\boldsymbol{\mathcal{A}}^x\}_{x\in\Sigma}, \diamond)$ be a $\diamond$-GWM. Let $\mathbf{M} \in \mathbb{F}^{n\times n}$ be the matrix defined by $\mathbf{M}_{i,j} = \mathbf{e}_i \diamond \mathbf{e}_j$. Since $\diamond$ is symmetric, the matrix $\mathbf{M}$ is symmetric and there exists a matrix $\mathbf{Q} \in \mathbb{C}^{n\times n}$ such that $\mathbf{M} = \mathbf{Q}\mathbf{Q}^\top$. Let $B = (\mathbb{C}^n, \{\boldsymbol{\mathcal{B}}^x\}_{x\in\Sigma})$ be the GWM defined by $\boldsymbol{\mathcal{B}}^x = \boldsymbol{\mathcal{A}}^x \times_1 \mathbf{Q}^\top \times_2 \cdots \times_{\sharp x} \mathbf{Q}^\top$ for all $x \in \Sigma$. We claim that B computes the same function as A. Indeed, let $G = (V, E, \ell)$ be a closed graph. Using the notation $\boldsymbol{\mathcal{A}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{A}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$ we have

$$\begin{aligned}
f_A(G) &= \sum_{\gamma\in[n]^{P(G)}} \boldsymbol{\mathcal{A}}_\gamma \prod_{\{p,p'\}\in E} (\mathbf{e}_{\gamma(p)} \diamond \mathbf{e}_{\gamma(p')}) \\
&= \sum_{\gamma\in[n]^{P(G)}} \boldsymbol{\mathcal{A}}_\gamma \prod_{\{p,p'\}\in E} \sum_{i=1}^{n} \mathbf{Q}_{\gamma(p),i}\mathbf{Q}_{\gamma(p'),i} \\
&= \sum_{\gamma\in[n]^{P(G)}} \boldsymbol{\mathcal{A}}_\gamma \sum_{\tilde\gamma\in\Gamma_{id}(G,B)} \prod_{p\in P(G)} \mathbf{Q}_{\gamma(p),\tilde\gamma(p)} \\
&= \sum_{\gamma\in[n]^{P(G)}} \sum_{\tilde\gamma\in\Gamma_{id}(G,B)} \prod_{v\in V} \boldsymbol{\mathcal{A}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \mathbf{Q}_{\gamma(v,1),\tilde\gamma(v,1)} \cdots \mathbf{Q}_{\gamma(v,\sharp v),\tilde\gamma(v,\sharp v)} \\
&= \sum_{\tilde\gamma\in\Gamma_{id}(G,B)} \prod_{v\in V} \left(\boldsymbol{\mathcal{A}}^{\ell(v)} \times_1 \mathbf{Q}^\top \times_2 \cdots \times_{\sharp v} \mathbf{Q}^\top\right)_{\tilde\gamma(v,1),\cdots,\tilde\gamma(v,\sharp v)} \\
&= \sum_{\tilde\gamma\in\Gamma_{id}(G,B)} \prod_{v\in V} \boldsymbol{\mathcal{B}}^{\ell(v)}_{\tilde\gamma(v,1),\cdots,\tilde\gamma(v,\sharp v)} = f_B(G).
\end{aligned}$$

□

It is important to note that while $\diamond$-GWMs and GWMs with coefficients in $\mathbb{C}$ are strictly equivalent, it is not the case when the coefficients are in $\mathbb{R}$. This is explained by the fact that there exist functions on graphs that can only be computed by GWMs with complex

coefficients: consider the graph $G = a - a$ connecting two vertices labeled by the arity-one symbol a; for any GWM M we have $f_M(G) = \sum_i (\boldsymbol{\mathcal{M}}_i^a)^2$, which can only be negative if M has its coefficients in $\mathbb{C}$. However, it is easy to construct a $\diamond$-GWM with coefficients in $\mathbb{R}$ computing a function such that $f_M(G) < 0$, but the corresponding GWM *will have its coefficients in* $\mathbb{C}$.

A first fundamental difference between GWMs and recognizable series on strings follows from the previous discussion: it is known that any recognizable string series taking its values in $\mathbb{R}$ can be computed by a weighted automaton with real weights (Berstel and Reutenauer, 1988, Theorem 2.1); this is not the case for GWM-recognizable series. However, we conjecture that it is true for GWMs defined on families of in-out graphs. We will go back to this question in Section 2.4 where we will show that under some assumptions on a family of in-out graphs $\mathcal{S}$, any real-valued function $f : \mathcal{S} \to \mathbb{R}$ with finite support can be computed by a GWM with real weights. This is one of the results that led us to state the previous conjecture.

2.3 Properties of Graph Weighted Models

In the previous section, we introduced the novel computational model on graphs of GWMs. The definition of this model was inspired by our interpretation of the computations of classical weighted automata on strings and trees in terms of tensor networks. In this section, we will start by formally showing that GWMs generalize the classical models of weighted automata on strings and trees and the more recent model of weighted automata on pictures: we will show that any function computed by a weighted automaton on strings/trees/pictures can be computed by a GWM (on the associated graphs described in the previous section). This is in some sense a first step towards showing that GWMs define a legitimate computational model on graphs.

In order to go further in this direction, we will then study closure properties of GWM-recognizable functions. It is well known that the set of recognizable functions on string (or trees) is closed under sum, (Hadamard) product and scalar multiplication. Such properties are desirable for a model extending the notion of recognizable series to graphs. We will show that GWM-recognizable functions defined over families of connected graphs are closed under sum and Hadamard product. However, we will see that the set of GWM-recognizable functions is not closed under scalar multiplication. This is one of the fundamental properties that is not satisfied by GWMs defined over the whole set of graphs $\mathfrak{G}_{\mathcal{F}}$ in general. Nonetheless, we will show that this property holds if we consider GWMs defined over any family of rooted graphs. The recognizability of finite support series (i.e. functions that are different from zero only on a finite number of graphs) is also a fundamental property that is satisfied by recognizable functions on strings and trees but not by GWMs in general. The study of this property is deferred to the next section.

Finally, we will conclude this section by showing that GWMs can induce compressed representations of functions computed by weighted automata. Indeed, we will show that there exist functions that can be computed by a GWM with n states on (rooted) circular strings but that cannot be computed by a weighted automaton on strings with less than n^2 states.

2.3.1 GWMs and Weighted Automata of Strings, Trees and Pictures

Strings and trees. The following propositions show that GWMs encompass the classical notions of recognizable series on strings and trees.

Proposition 2. *Let $A = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ be a weighted string automaton over Σ^*. For any word $w \in \Sigma^*$, let $graph(w)$ be the associated graph on the ranked alphabet $(\Sigma \cup \{\alpha, \omega\}, \sharp)$, whose construction is described in Section 2.2.2. Consider the GWM $M = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma\cup\{\alpha,\omega\}})$ where $\boldsymbol{\mathcal{M}}^\alpha = \boldsymbol{\alpha}$, $\boldsymbol{\mathcal{M}}^\omega = \boldsymbol{\omega}$ and $\boldsymbol{\mathcal{M}}^\sigma = \mathbf{A}^\sigma$ for all $\sigma \in \Sigma$.*

Then, $f_A(w) = f_M(graph(w))$ for all strings $w \in \Sigma^*$.

Proof. Let $w = \sigma_1 \cdots \sigma_k$. We have

$$f_A(w) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1} \cdots \mathbf{A}^{\sigma_k} \boldsymbol{\omega} = \sum_{i_0,\ldots,i_k=1}^{n} \boldsymbol{\alpha}_{i_0} \mathbf{A}^{\sigma_1}_{i_0,i_1} \cdots \mathbf{A}^{\sigma_k}_{i_{k-1},i_k} \boldsymbol{\omega}_{i_k}$$

$$= \sum_{i_0,\ldots,i_k=1}^{n} \boldsymbol{\mathcal{M}}^{\alpha}_{i_0} \boldsymbol{\mathcal{M}}^{\sigma_1}_{i_0,i_1} \cdots \boldsymbol{\mathcal{M}}^{\sigma_k}_{i_{k-1},i_k} \boldsymbol{\mathcal{M}}^{\omega}_{i_k} = f_M(graph(w)).$$

□

Proposition 3. *Let $T = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{T}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ be a weighted tree automaton over trees on the ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$. For any tree $t \in \mathfrak{T}_\mathcal{F}$, let $graph(t) = (V_t, E_t, \ell_t)$ be the associated graph on the ranked alphabet $(\Sigma\cup\{\alpha\}, \sharp')$, whose construction is described in Section 2.2.2. Consider the GWM $M : \langle \mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma\cup\{\alpha\}} \rangle$ where $\boldsymbol{\mathcal{M}}^\alpha = \boldsymbol{\alpha}$, $\boldsymbol{\mathcal{M}}^f = \boldsymbol{\mathcal{T}}^f$ for all $f \in \mathcal{F}_{\geq 1}$ and $\boldsymbol{\mathcal{M}}^\sigma = \boldsymbol{\omega}^\sigma$ for all $\sigma \in \mathcal{F}_0$.*

Then, $f_T(t) = f_M(graph(t))$ for all $t \in \mathfrak{T}_\mathcal{F}$.

Proof. For any tree t let $\tilde{G}^t$ be the graph with one free port obtained by removing the root vertex in $graph(t)$ (see Section 2.2.2, page 31). We first show by induction on the height of t that $\mu(\tilde{G}^t) = \boldsymbol{\omega}(t)$ for all tree t where μ is the graph-tensor mapping induced by the GWM M. If $t = \sigma$ for some $\sigma \in \mathcal{F}_0$ we have $\mu(\tilde{G}^t) = \boldsymbol{\mathcal{M}}^\sigma = \boldsymbol{\omega}^\sigma = \boldsymbol{\omega}(\sigma)$. Suppose the result true for trees of height up to $k-1$ and let $t = g(t_1, \cdots, t_p)$ be a tree of height k. We have

$$\begin{aligned}\mu(\tilde{G}^t) &= \langle\!\langle \boldsymbol{\mathcal{M}}^g \circ \mu(\tilde{G}^{t_1}) \circ \mu(\tilde{G}^{t_2}) \circ \cdots \circ \mu(\tilde{G}^{t_p}) \rangle\!\rangle_{(2,p+2),(3,p+3),\cdots,(p+1,2p+1)} \\ &= \boldsymbol{\mathcal{M}}^g \bullet_2 \mu(\tilde{G}^{t_1}) \bullet_3 \cdots \bullet_{p+1} \mu(\tilde{G}^{t_p}) \\ &= \boldsymbol{\mathcal{T}}^g \bullet_2 \boldsymbol{\omega}(t_1) \bullet_3 \cdots \bullet_{p+1} \boldsymbol{\omega}(t_p) = \boldsymbol{\omega}(t).\end{aligned}$$

It then follows that $f_M(graph(t)) = \langle\!\langle \boldsymbol{\mathcal{M}}^\alpha \circ \mu(\tilde{G}^t) \rangle\!\rangle_{(1,2)} = \boldsymbol{\alpha}^\top \boldsymbol{\omega}(t) = f_T(t)$. □

Two-dimensional words. Motivated by problems arising in image processing and patter recognition, various computational models on two-dimensional arrays of symbols have been proposed. We will now present such a model that extends string weighted automata to two-dimensional words and we will show that functions computed by this model can be computed by GWMs.

A *2d-word language* (or picture language) is a set of 2d-words and a *2d-word series* (or picture series) is a function from Σ^{++} to a commutative semi-ring. Regular 2d-

word languages can equivalently be described in terms of automata, set of tiles, rational operations or monadic second order logic (Giammarresi and Restivo, 1996; Giammarresi et al., 1996; Inoue and Nakamura, 1977; Latteux and Simplot, 1997). The extension of regular 2d-word languages to the quantitative setting led to the definition of *recognizable 2d-word series* whose theoretical study has been of recent interest (Bozapalidis and Grammatikopoulou, 2005; Mäurer, 2005; Mäurer, 2006; Babari and Droste, 2015). Recognizable 2d-word series have been first introduced in (Bozapalidis and Grammatikopoulou, 2005) by means of *weighted picture automaton*.

Definition 4. *A* weighted (quadropolic) picture automaton (WPA) *on a finite alphabet* Σ *is a tuple* $A = \langle Q, R, F_w, F_n, F_e, F_s, \delta\rangle$ *consisting of a finite set of states* Q*, a finite set of rules* $R \subseteq \Sigma \times Q^4$*, four poles of acceptance* $F_w, F_n, F_e, F_s \subseteq Q$*, and a weight function* $\delta : R \to \mathbb{F}$*.*

Given a rule $r = (\sigma, q_w, q_n, q_e, q_s) \in R$ we denote by $\ell(r)$ its label σ, and by $west(r) = q_w, north(r) = q_n, east(r) = q_e$ and $south(r) = q_s$ the states corresponding to its four poles.

A *run* c of A on a picture $p \in \Sigma^{m\times n}$ is an element in $R^{m\times n}$ satisfying the following compatibility properties:

$$\begin{array}{c} \forall i \leq m-1, \forall j \leq n : south(c_{i,j}) = north(c_{i+1,j}) \\ \forall i \leq m, \forall j \leq n-1 : east(c_{i,j}) = west(c_{i,j+1}) \end{array} \tag{2.1}$$

and $\ell(c_{i,j}) = p_{i,j}$ for all $i \leq m, j \leq n$. A run is *successful* if its outer pole-states are in the respective poles of acceptance, that is

$$\forall i \leq m, \forall j \leq n : west(c_{i,1}) \in F_w, south(c_{m,j}) \in F_s, east(c_{i,n}) \in F_e, north(c_{1,j}) \in F_n.$$

We denote by $R(p)$ the set of all successful runs on a picture p.

The weight function δ is extended to runs by setting $\delta(c) = \prod_{i,j} \delta(c_{i,j})$. The *weight* of a picture p is the sum of the weights of all successful runs on p. It defines a picture series $f_A : \Sigma^{++} \to \mathbb{F}$ with $f_A(p) = \sum_{c\in R(p)} \delta(c)$. If there are no successful run on p then $f_A(p) = 0$.

The following proposition shows that any function that can be computed by a weighted picture automaton can be computed by a GWM.

Proposition 4. *Let* $A = \langle Q, R, F_w, F_n, F_e, F_s, \delta\rangle$ *be a WPA with* d *states* $(q_1, \cdots, q_d)$ *on the alphabet* Σ *over the field* $\mathbb{F} = \mathbb{R}$ *or* $\mathbb{F} = \mathbb{C}$*. For any picture* p*, let* $graph(p)$ *be the associated graph on the ranked alphabet* $\mathcal{F} = (\Sigma \cup \{\mathtt{w},\mathtt{n},\mathtt{e},\mathtt{s}\}, \sharp)$*, whose construction is described in Section 2.2.2. Consider the GWM* $M : \langle \mathbb{F}^d, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\mathcal{F}}\rangle$ *where*

$$\boldsymbol{\mathcal{M}}^x_i = \begin{cases} 1 & \text{if } q_i \in F_x \\ 0 & \text{otherwise} \end{cases} \text{ for any } x \in \{\mathtt{w},\mathtt{n},\mathtt{e},\mathtt{s}\}, \quad \text{and}$$

$$\boldsymbol{\mathcal{M}}^\sigma_{i_1 i_2 i_3 i_4} = \begin{cases} \delta(r) & \text{if } r = (\sigma, q_{i_1}, q_{i_2}, q_{i_3}, q_{i_4}) \in R \\ 0 & \text{otherwise} \end{cases} \text{ for any } \sigma \in \Sigma.$$

Then, $f_M(graph(p)) = f_A(p)$ *for all pictures* $p \in \Sigma^{++}$*.*

Proof. Without loss of generality, we suppose that $Q = [d]$ and that $\delta(r) \neq 0$ for all rules $r \in R$. Let $p \in \Sigma^{m\times n}$ and $G = graph(p)$, we have $f_M(G) = \sum_{\gamma\in\Gamma_{id}(G,M)} \boldsymbol{\mathcal{M}}_\gamma$ where $\boldsymbol{\mathcal{M}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1)\cdot\gamma(v,\sharp v)}$. We associate each multi-index $\gamma \in \Gamma_{id}(G, M)$ with a run c^γ of A on p by letting $c^\gamma_{i,j} = (\ell(v_{i,j}), \gamma(v_{i,j}, 1), \cdots, \gamma(v_{i,j}, 4))$ for all i, j. Note that it follows from the definition of $\Gamma_{id}(G, M)$ that c is a valid run of A (i.e. it satisfies Eq. 2.1). However, c^γ may not be in $R^{m\times n}$, in which case we say that c^γ is *unsuccessful*.

We now show that the only non-zero summands $\boldsymbol{\mathcal{M}}_\gamma$ in $f_M(G)$ are those for which c^γ is a successful run, which implies that $f_M(G) = \sum_{\gamma:c^\gamma\in R(p)} \boldsymbol{\mathcal{M}}_\gamma$. Indeed, a run c^γ can be unsuccessful for two reasons: either one of its components $c_{i,j}$ is not in R, in which case $\boldsymbol{\mathcal{M}}_{\gamma(v_{ij})} = 0$ (where $\boldsymbol{\mathcal{M}}_{\gamma(v)} = \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$), or one of its outer pole-states, let say the north pole of $c_{1,j}$ for some $j \in [n]$, is not in the corresponding pole of acceptance F_n, in which case $\boldsymbol{\mathcal{M}}_{\gamma(\bar{n}_j)} = 0$.

It remains to show that $\boldsymbol{\mathcal{M}}_\gamma = \delta(c^\gamma)$ for all $c^\gamma \in R(p)$. This follows from the fact that since c^γ is successful, $\boldsymbol{\mathcal{M}}_{\gamma(v)} = 1$ for all vertex v such that $\ell(v) \in \{\mathtt{w}, \mathtt{n}, \mathtt{e}, \mathtt{s}\}$, and $\boldsymbol{\mathcal{M}}_{\gamma(v_{ij})} = \delta(c^\gamma_{ij})$ for all $i \in [m], j \in [n]$. Hence $f_M(G) = \sum_{c\in R(p)} \delta(c) = f_A(p)$. □

2.3.2 Closure Properties

The following propositions show that the set of GWM-recognizable series is closed under sum and Hadamard product. The constructions we use are similar to the classical ones used to prove the same closure properties for recognizable series over strings and trees (i.e. we use direct sum for the sum and Kronecker product for the Hadamard product).

Proposition 5. *Let $A = (\mathbb{F}^m, \{\boldsymbol{\mathcal{A}}^x\}_{x\in\Sigma})$ and $B = (\mathbb{F}^n, \{\boldsymbol{\mathcal{B}}^x\}_{x\in\Sigma})$ be two GWMs on the same ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$. The GWM $C = (\mathbb{F}^{m+n}, \{\boldsymbol{\mathcal{C}}^x\}_{x\in\Sigma})$ defined by $\boldsymbol{\mathcal{C}}^x = \boldsymbol{\mathcal{A}}^x \oplus \boldsymbol{\mathcal{B}}^x$ for all $x \in \Sigma$ is such that $f_C(G) = f_A(G) + f_B(G)$ for any connected graph $G \in \mathfrak{G}^0_{\mathcal{F}}$.*

Proof. Let $G = (V, E, \ell) \in \mathfrak{G}^0_{\mathcal{F}}$. Let $\Gamma_A = \{\gamma \in \Gamma_{id}(G, C) : 1 \leq \gamma(p) \leq m \text{ for all } p \in P(G)\}$ and $\Gamma_B = \{\gamma \in \Gamma_{id}(G, C) : m < \gamma(p) \leq m + n \text{ for all } p \in P(G)\}$ and note that $\Gamma_A \cap \Gamma_B = \emptyset$. We use the notation $\boldsymbol{\mathcal{C}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{C}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$ and similarly for $\boldsymbol{\mathcal{A}}_\gamma$ and $\boldsymbol{\mathcal{B}}_\gamma$.

We first show that for any $\gamma \in \Gamma_{id}(G, C)$, if $\boldsymbol{\mathcal{C}}_\gamma \neq 0$ then $\gamma \in \Gamma_A \cup \Gamma_B$. Let $\gamma \in \Gamma_{id}(G, C)$ be such that $\boldsymbol{\mathcal{C}}_\gamma \neq 0$ and let $v \in V$ be any vertex of G. Since $\boldsymbol{\mathcal{C}}_\gamma \neq 0$ we have $\boldsymbol{\mathcal{C}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \neq 0$, thus by definition of the tensor $\boldsymbol{\mathcal{C}}^{\ell(v)}$ we must have either $\gamma(v, 1), \cdots, \gamma(v, \sharp v) \in \{1, \cdots, m\}$ or $\gamma(v, 1), \cdots, \gamma(v, \sharp v) \in \{m+1, \cdots, m+n\}$. Suppose $\gamma(v, i) \in \{1, \cdots, m\}$ for all $i \in [\sharp v]$, then for any $(v', i') \in P(G)$ such that $\{(v, i), (v', i')\} \in E$ for some $i \in [\sharp v]$ we have $\gamma(v, i) = \gamma(v', i') \in \{1, \cdots, m\}$ (since $\gamma \in \Gamma_{id}(G, C)$), and since $\boldsymbol{\mathcal{C}}_\gamma \neq 0$ we must have $\boldsymbol{\mathcal{C}}^{\ell(v')}_{\gamma(v',1),\cdots,\gamma(v',\sharp v')} \neq 0$ from which it follows that $\gamma(v', i') \in \{1, \cdots, m\}$ for all $i' \in [\sharp v']$. Since G is connected this argument can be extended from neighbor to neighbor to all vertices of G which implies $\gamma \in \Gamma_A$. The same reasoning can be used to show that if $\gamma(v, 1), \cdots, \gamma(v, \sharp v) \in \{m + 1, \cdots, m + n\}$ then $\gamma \in \Gamma_B$. Hence if $\boldsymbol{\mathcal{C}}_\gamma \neq 0$ then $\gamma \in \Gamma_A \cup \Gamma_B$.

It then follows that

$$\begin{aligned} f_C(G) &= \sum_{\gamma\in\Gamma_{id}(G,C)} \boldsymbol{\mathcal{C}}_\gamma = \sum_{\gamma\in\Gamma_{id}(G,C):\boldsymbol{\mathcal{C}}_\gamma\neq 0} \boldsymbol{\mathcal{C}}_\gamma = \sum_{\gamma\in\Gamma_A} \boldsymbol{\mathcal{C}}_\gamma + \sum_{\gamma\in\Gamma_B} \boldsymbol{\mathcal{C}}_\gamma \\ &= \sum_{\gamma\in\Gamma_{id}(G,A)} \boldsymbol{\mathcal{A}}_\gamma + \sum_{\gamma\in\Gamma_{id}(G,B)} \boldsymbol{\mathcal{B}}_\gamma = f_A(G) + f_B(G). \end{aligned}$$

□

Proposition 6. *Let $A = (\mathbb{F}^m, \{\boldsymbol{\mathcal{A}}^x\}_{x\in\Sigma})$ and $B = (\mathbb{F}^n, \{\boldsymbol{\mathcal{B}}^x\}_{x\in\Sigma})$ be two GWMs on the same ranked alphabet $\mathcal{F}(\Sigma, \sharp)$. The GWM $C = (\mathbb{F}^{mn}, \{\boldsymbol{\mathcal{C}}^x\}_{x\in\Sigma})$ defined by $\boldsymbol{\mathcal{C}}^x = \boldsymbol{\mathcal{A}}^x \otimes \boldsymbol{\mathcal{B}}^x$ for all $x \in \Sigma$ is such that $f_C(G) = f_A(G)f_B(G)$ for all $G \in \mathfrak{G}^0_{\mathcal{F}}$.*

Proof. Let $\phi_1 : [mn] \to [m]$ and $\phi_2 : [mn] \to [n]$ be the mappings defined by $(\mathbf{u} \otimes \mathbf{v})_i = \mathbf{u}_{\phi_1(i)}\mathbf{v}_{\phi_2(i)}$ for all $\mathbf{u} \in \mathbb{F}^m$, $\mathbf{v} \in \mathbb{F}^n$ and $i \in [mn]$. Using the notation $\boldsymbol{\mathcal{C}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{C}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$ and similarly for $\boldsymbol{\mathcal{A}}_\gamma$ and $\boldsymbol{\mathcal{B}}_\gamma$, we have

$$\begin{aligned} f_C(G) &= \sum_{\gamma\in\Gamma_{id}(G,C)} \boldsymbol{\mathcal{C}}_\gamma = \sum_{\gamma\in\Gamma_{id}(G,C)} \prod_{v\in V} \left(\boldsymbol{\mathcal{A}}^{\ell(v)} \otimes \boldsymbol{\mathcal{B}}^{\ell(v)}\right)_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \\ &= \sum_{\gamma\in\Gamma_{id}(G,C)} \prod_{v\in V} \boldsymbol{\mathcal{A}}^{\ell(v)}_{\phi_1(\gamma(v,1)),\cdots,\phi_1(\gamma(v,\sharp v))} \boldsymbol{\mathcal{B}}^{\ell(v)}_{\phi_2(\gamma(v,1)),\cdots,\phi_2(\gamma(v,\sharp v))} \\ &= \sum_{\substack{\gamma_1\in\Gamma_{id}(G,A)\\ \gamma_2\in\Gamma_{id}(G,B)}} \prod_{v\in V} \boldsymbol{\mathcal{A}}^{\ell(v)}_{\gamma_1(v,1),\cdots,\gamma_1(v,\sharp v)} \boldsymbol{\mathcal{B}}^{\ell(v)}_{\gamma_2(v,1),\cdots,\gamma_2(v,\sharp v)} \\ &= \sum_{\gamma_1\in\Gamma_{id}(G,A)} \boldsymbol{\mathcal{A}}_\gamma \sum_{\gamma_1\in\Gamma_{id}(G,B)} \boldsymbol{\mathcal{B}}_\gamma = f_A(G)f_B(G). \end{aligned}$$

□

Interestingly, the set of GWM-recognizable series is not closed under scalar multiplication in general. To see this, consider the GWM-recognizable series on circular strings on the one-letter alphabet $\Sigma = \{a\}$ defined by $f(a^n) = \kappa^n$ for some $\kappa \in \mathbb{R} \setminus \{0, 1\}$. Then, for any real number $\alpha \notin \mathbb{N}$ the series $f' : a^n \mapsto \alpha f(a^n)$ is not GWM-recognizable. Indeed, this would imply that there exists some matrix $\mathbf{M}$ such that $\mathrm{Tr}(\mathbf{M}^n) = \alpha\kappa^n$ for all n, but it follows from the following lemma that if $\mathrm{Tr}((\frac{\mathbf{M}}{\kappa})^n) = \alpha$ for all n, then $\alpha \in \mathbb{N}$.

Lemma 2. *Let $\mathbf{M} \in \mathbb{R}^{n\times n}$. If there exists an integer k such that $\mathrm{Tr}(\mathbf{M}^k) = \mathrm{Tr}(\mathbf{M}^{k+1}) = \cdots = \mathrm{Tr}(\mathbf{M}^{k+n})$, then for all $m \in \mathbb{N}, \mathrm{Tr}(\mathbf{M}^m) = \mathrm{Tr}(\mathbf{M}) \in \mathbb{N}$.*

Proof. Let $\lambda_1, \ldots, \lambda_p \in \mathbb{C}$ be the distinct non zero eigenvalues of $\mathbf{M}$, with multiplicities $n_1, \ldots, n_p$ respectively. If $p = 0$, the spectrum of $\mathbf{M}$ is reduced to 0 and $\forall m \in \mathbb{N}, \mathrm{Tr}(\mathbf{M}^m) = \mathrm{Tr}(\mathbf{M}) = 0$.

Suppose that $p > 0$. Let $\mathbf{D} = diag(\lambda_1, \ldots, \lambda_p)$ and let $\mathbf{N} \in \mathbb{R}^{p\times p}$ be the square matrix defined by $\mathbf{N}_{i,j} = \lambda_j^{i-1}$. The matrix $\mathbf{N}$ is full rank and its determinant is equal to $\prod_{i<j}(\lambda_j - \lambda_i)$. For any $k \in \mathbb{N}$, let $\mathbf{u}_k = (\lambda_1^k n_1, \ldots, \lambda_p^k n_p)^\top$. We have $\mathbf{N}\mathbf{u}_k = (\mathrm{Tr}(\mathbf{M}^k), \ldots, \mathrm{Tr}(\mathbf{M})^{k+p-1})$. Now, suppose that there exists an integer k such that $\mathrm{Tr}(\mathbf{M}^k) = \cdots = \mathrm{Tr}(\mathbf{M})^{k+p} = \alpha$. Then, $\mathbf{N}\mathbf{u}_k = \mathbf{N}\mathbf{u}_{k+1}$ and since $\mathbf{N}$ is invertible, $\mathbf{u}_k = \mathbf{u}_{k+1}$. Hence, $\lambda_1 = \cdots = \lambda_p = 1$ and $p = 1$. Therefore, $\forall m \in \mathbb{N}, \mathrm{Tr}(\mathbf{M}^m) = \mathrm{Tr}(\mathbf{M}) = n_1$. □

Nonetheless, there are families of graphs for which GWMs are closed under scalar multiplication. This is for example the case for strings and trees. More generally, this is the case for any family of rooted graphs: multiplication by a scalar λ is simply achieved by multiplying the tensor $\boldsymbol{\mathcal{M}}^{\alpha_0}$ associated with the root symbol α_0 by λ. Thus, even though GWMs are not closed under scalar multiplication in general, we managed to identify a family of graphs that generalizes strings and trees and for which all the closure properties we considered in this section hold; this motivates the following theorem.

Theorem 1. *Let $\mathcal{S}$ be a family of rooted connected graphs. Then the set of functions $f : \mathcal{S} \to \mathbb{F}$ that can be computed by GWMs is closed under addition, Hadamard product and scalar multiplication.*

2.3.3 Rooted Circular Strings and Compressed Representations

To conclude this section, we show on a simple example how GWMs can induce compressed representations of functions defined over strings. More precisely, we will show that there exists a function $f : \Sigma^* \to \mathbb{F}$ that can be computed by a GWM with n states and that cannot be computed by a weighted string automaton with less than n^2 states.

We consider the family of rooted circular strings with root symbol λ: for any string $w = w_1 \cdots w_k \in \Sigma^*$, we associate w with the circular string G_w on the ranked alphabet $(\Sigma' = \Sigma \cup \{\lambda\}, \sharp)$ where λ is a new symbol, $\sharp\sigma = 2$ for any $\sigma \in \Sigma \cup \{\lambda\}$. G_w has vertices $V = \{0, \cdots, k\}$, labels $l(0) = \lambda$ and $l(i) = w_i$ for $i \in [k]$, and edges $\{(k, 2), (0, 1)\}$ and $\{(i, 2), (i + 1, 1)\}$ for $i \in [k - 1]$. It is easy to check that a GWM $M = (\mathbb{F}^n, \{\mathbf{M}^\sigma\}_{\sigma\in\Sigma\cup\{\lambda\}})$ computes the function $G_w \mapsto \mathrm{Tr}(\mathbf{M}^\lambda \mathbf{M}^w)$ where $\mathbf{M}^w = \mathbf{M}^{w_1}\mathbf{M}^{w_2} \cdots \mathbf{M}^{w_k}$.

Let $A = (\mathbb{F}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ be a string weighted automaton on Σ^*. We define the GWM $M = (\mathbb{F}^n, \{\mathbf{M}^\sigma\}_{\sigma\in\Sigma'})$ where $\mathbf{M}^\sigma = \mathbf{A}^\sigma$ for all $\sigma \in \Sigma$ and $\mathbf{M}^\lambda = \boldsymbol{\omega}\boldsymbol{\alpha}^\top$. For all $w \in \Sigma^*$, $f_M(G_w) = f_A(w)$: indeed, $f_A(w) = \boldsymbol{\alpha}^\top \mathbf{A}^w \boldsymbol{\omega} = \mathrm{Tr}(\boldsymbol{\alpha}^\top \mathbf{A}^w \boldsymbol{\omega}) = \mathrm{Tr}(\mathbf{M}^\lambda \mathbf{M}^w)$ since the trace operator is invariant under cyclic permutations.

Now consider m string recognizable series $f_1, \ldots, f_m$ whose linear representations $(\mathbb{F}^n, \boldsymbol{\alpha}_i, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega}_i)$ for $i \in [m]$ share the same transition matrices and let $f = f_1 + \cdots + f_m$. It can be checked that $f(w) = \mathrm{Tr}(\mathbf{A}^\lambda \mathbf{A}^w)$, where $\mathbf{A}^\lambda = \sum_{i=1}^m \boldsymbol{\omega}_i \boldsymbol{\alpha}_i^\top$. Hence, the n-dimensional GWM $M = (\mathbb{F}^n, \{\mathbf{M}^\sigma\}_{\sigma\in\Sigma'})$, where $\mathbf{M}^\sigma = \mathbf{A}^\sigma$ for all $\sigma \in \Sigma$, is such that $f_M(G_w) = f(w)$ for all $w \in \Sigma^*$.

It can easily be shown, by decomposing $\mathbf{A}^\lambda$ as a sum of at most n rank-one matrices, that the rank of f is at most n^2 (while M has only n states). The following proposition shows that this upper bound can be achieved (an example of such a WA is shown in Figure 2.7).

Proposition 7. *There exists a recognizable string series of rank n^2 that can be computed by a GWM with n states on rooted circular strings.*

Proof. Let $(\mathbf{E}^{i,j})_{1\le i,j\le n}$ be the canonical basis of $\mathbb{R}^{n\times n}$ (i.e. $\mathbf{E}^{i,j} = \mathbf{e}_i\mathbf{e}_j^\top$). Let $\Sigma = \{a, b\}$ and let $\mathbf{A}^a, \mathbf{A}^b \in \mathbb{R}^{n\times n}$ be defined by

$$\mathbf{A}^a = \mathbf{E}^{1,1} \text{ and } \mathbf{A}^b = \mathbf{E}^{2,1} + \mathbf{E}^{3,2} + \cdots + \mathbf{E}^{n,n-1} + \mathbf{E}^{1,n}.$$

For any $i \in [n]$, let f_i be the function computed by the WA $A_i = (\mathbb{R}^n, \mathbf{e}_i, \{\mathbf{A}^a, \mathbf{A}^b\}, \mathbf{e}_i)$, and let $f = f_1 + \cdots + f_n$. We have $f(w) = \mathrm{Tr}(\mathbf{A}^w)$ for any $w \in \Sigma^*$, hence f can be

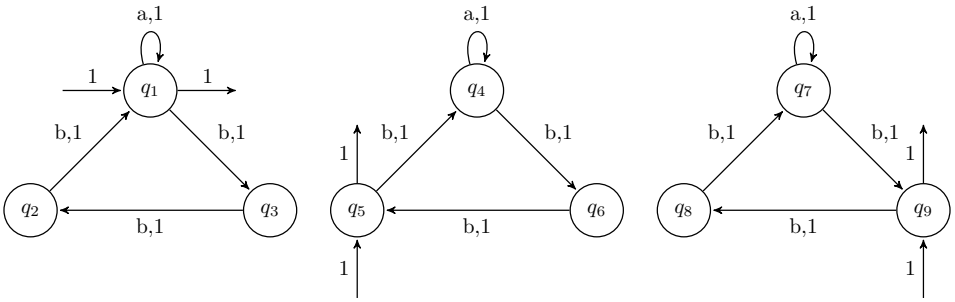


Figure 2.7: A minimal weighted automaton with 9 states computing a series that can be computed by a 3-dimensional GWM on rooted circular strings.

computed by an n-dimensional GWM on rooted circular strings (with $\mathbf{A}^{\lambda} = \mathbf{I}$). We claim that the rank of the string series f is n^2. Since f is the sum of n recognizable functions of rank n, it is of rank at most n^2 (see Proposition 5). We now show that $\text{rank}(f) \geq n^2$. Indeed, it can be shown that the matrix algebra generated by $\mathbf{A}^a$ and $\mathbf{A}^b$ is equal to $\mathbb{R}^{n\times n}$, thus there exist $w_1, \ldots, w_{n^2} \in \Sigma^*$ such that the matrices $\mathbf{A}^{w_1}, \cdots, \mathbf{A}^{w_{n^2}}$ are linearly independent. Let $\mathbf{H} \in \mathbb{R}^{n^2\times n^2}$ be the so-called Hankel matrix defined by $\mathbf{H}_{i,j} = \text{Tr}(\mathbf{A}^{w_i w_j})$. Using the fact that $\text{Tr}(\mathbf{XY}) = \text{vec}(\mathbf{X}^\top)^\top \text{vec}(\mathbf{Y})$ for any matrices $\mathbf{X}$ and $\mathbf{Y}$ we have $\mathbf{H} = \mathbf{PS}$ where $\mathbf{P}$ and $\mathbf{S}$ are the full rank $n^2 \times n^2$ matrices defined by

$$\mathbf{P}_{i,:} = \text{vec}((\mathbf{A}^{w_i})^\top)^\top \text{ and } \mathbf{S}_{:,j} = \text{vec}(\mathbf{A}^{w_j})$$

for all $i, j \in [n^2]$. It follows from Sylvester's rank inequality that $\mathbf{H}$ has rank n^2. From a fundamental theorem on recognizable string series (Carlyle and Paz, 1971), the rank of $\mathbf{H}$ is a lower bound of the rank of f, which entails the result. □

2.4 Recognizability of Finite Support Series

In this section, we study the recognizability of finite support series. Throughout the section, *we only consider connected graphs* without always explicitly mentioning it. Let $\mathcal{S} \subseteq \mathfrak{G}^0_{\mathcal{F}}$ be a family of closed graphs. The support of a series $f : \mathcal{S} \to \mathbb{F}$ is the set of graphs to which f assigns a non-zero value. Thus, a *finite support series* is a series f satisfying $|\{G \in \mathcal{S} \ : \ f(G) \neq 0\}| < \infty$. Since the set of GWM-recognizable series is closed under sum, the recognizability of finite support series boils down to the following problem: for any graph $\hat{G} \in \mathcal{S}$ and any scalar $\lambda \in \mathbb{F}$, does there exist a GWM M such that $f_M(\hat{G}) = \lambda$ and $f_M(G) = 0$ for any $G \neq \hat{G}$ (considering that two isomorphic graphs are equal)?

The question of the recognizability of finite support series over some family of graphs $\mathcal{S}$ is of great interest because it is related to whether the set of (square-summable) GWM-recognizable series is dense in the set of (square-summable) functions from $\mathcal{S}$ to $\mathbb{F}$. Indeed, suppose that $f : \mathcal{S} \to \mathbb{F}$ is a function such that $\sum_{G\in\mathcal{S}} f(G)^2 < \infty$, then the fact that finite support series on $\mathcal{S}$ are recognizable imply that we can approximate f

arbitrarily well with a GWM: for any $\varepsilon > 0$ there exists a GWM M such that

$$\|f - f_M\|_2^2 = \sum_{G \in \mathcal{S}} (f(G) - f_M(G))^2 < \varepsilon.$$

We first show that finite support series are not recognizable for all families of graphs on a simple example. Consider the family of circular strings $\mathfrak{G}_\Sigma^{circ}$ on the one letter alphabet $\Sigma = \{a\}$. Let M be a GWM such that $f_M(circ(a)) = \mathrm{Tr}(\mathbf{A}) = \lambda \neq 0$ where $\mathbf{A} = \boldsymbol{\mathcal{M}}^a$ is the only tensor of the GWM M. Suppose now that $f_M(circ(a^n)) = \mathrm{Tr}(\mathbf{A}^n) = 0$ for any $n > 1$, by Lemma 2 this would imply that $\mathrm{Tr}(\mathbf{A}) = 0$, a contradiction. Hence finite support series are not recognizable on the family of circular strings.

In the following, we show that the contradiction exhibited by this simple example is closely related to the graph theoretic notion of *covering*, and we provide a characterization of families of graphs for which finite support series are recognizable.

2.4.1 Graph Coverings

In this section, we define the notion of *coverings* (or *lifts*) of a graph which is a fundamental notion from graph theory (Reidemeister, 1950; Angluin, 1980). Intuitively, a *covering* of a graph $\hat{G}$ is a graph G made of copies of $\hat{G}$ (see Figure 2.8). More precisely,

Definition 5. *Let $\hat{G} = (\hat{V}, \hat{E}, \hat{\ell})$ be a graph over a ranked alphabet $(\Sigma, \sharp)$. A graph $G = (V, E, \ell)$ on the same alphabet $(\Sigma, \sharp)$ is a* covering *of $\hat{G}$ if and only if there exists a mapping $\psi : V \to \hat{V}$ such that*

(i) $\ell(v) = \hat{\ell}(\psi(v))$ for any $v \in V$,

(ii) for any edge $\{(u, i), (v, j)\} \in E$ of G, $\{(\psi(u), i), (\psi(v), j)\} \in \hat{E}$ is an edge of $\hat{G}$.

We will call such a mapping ψ a covering map *from G to $\hat{G}$.*

Note that it directly follows from the definition of a covering that any graph G is a covering of itself through the covering map $\psi(v) = v$ for all vertices v of G. Similarly, any graph G' that is isomorphic to G is a covering of G. Actually, the definition of covering of a graph only differs from the definition of isomorphic graphs in the fact that a covering map *need not* to be a bijection between the sets of vertices of the two graphs. Let us consider a less trivial example on circular strings: let Σ be a finite alphabet, then for any $w = w_1 \cdots w_k \in \Sigma^*$ the circular string $circ(ww) = (V, E, \ell)$ with $V = \{v_1, \cdots, v_{2k}\}$ is a covering of $circ(w) = (\hat{V}, E, \ell)$ with $\hat{V} = \{\hat{v}_1, \cdots, \hat{v}_k\}$ through the covering map $\psi(v_i) = \hat{v}_{\mathrm{mod}(i,k)+1}$ (where $\mathrm{mod}(i, k)$ is the remainder of the euclidean division of i by k). Another example of a covering built from 3 copies of the closed graph G from Example 1 (see page 29) is shown in Figure 2.8.

The following proposition will help us formalize our intuition of coverings as graphs built from copies of an initial graph.

Proposition 8. *Let ψ be a covering map from a graph $G = (V, E, \ell)$ to a connected closed graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\ell})$. Then for any $\hat{v} \in \hat{V}$, the cardinal of $\psi^{-1}(\{\hat{v}\})$ is a constant.*

Proof. Let $\hat{v} \in \hat{V}$ be a vertex of $\hat{G}$ and let $m = |\psi^{-1}(\{\hat{v}\})|$ where $\psi^{-1}(\{\hat{v}\}) = \{v \in V : \psi(v) = \hat{v}\}$. Let $i \in [\sharp v]$ and let $(\hat{u}, j) \in P(\hat{G})$ be the unique port of $\hat{G}$ satisfying

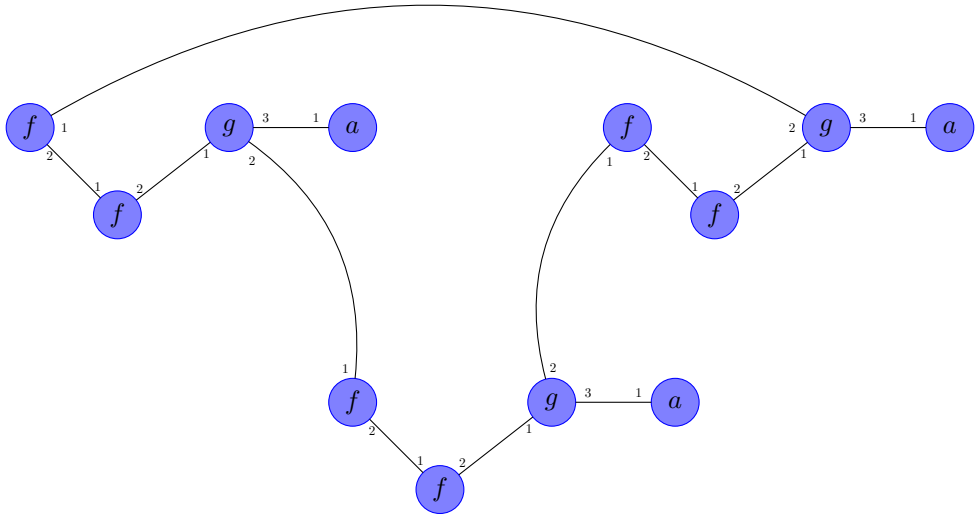


Figure 2.8: A covering made of three copies of the closed graph G from Example 1 (see page 29). The original graph contains 4 vertices, in each copy the edge between the first two vertices (labeled by f and g) is replaced with an edge between the corresponding vertices in distinct copies of G.

$\{(\hat{v}, i), (\hat{u}, j)\} \in \hat{E}$. Since ψ is a covering map, for all $v \in \psi^{-1}(\{\hat{v}\})$ there exists $u \in \psi^{-1}(\{\hat{u}\})$ such that $\{(v, i), (u, j)\} \in E$. Suppose $|\psi^{-1}(\{\hat{u}\})| < m$, then by the pigeonhole principle there exists $v_1, v_2 \in \psi^{-1}(\{\hat{v}\})$ and $u \in \psi^{-1}(\{\hat{u}\})$ such that $v_1 \neq v_2$ and both $\{(v_1, i), (u, j)\}$ and $\{(v_2, i), (u, j)\}$ are in E, a contradiction with the fact that E is a partition of $P(G)$. By switching the role of $\hat{u}$ and $\hat{v}$ the same contradiction can be obtained if $|\psi^{-1}(\{\hat{u}\})| > m$. Hence $|\psi^{-1}(\{\hat{u}\})| = m$. Since $\hat{G}$ is connected this argument can be extended from neighbor to neighbor to all the vertices of the graph, hence $|\psi^{-1}(\{\hat{w}\})| = m$ for all $\hat{w} \in \hat{V}$. □

A direct consequence of the previous proposition is that any covering map is surjective. This proposition also shows that for a connected graph, the formal definition of covering is equivalent to the intuition of a graph made of copies of the original one. Indeed, let ψ be a covering map from $G = (V, E, \ell)$ to a graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\ell})$. Let $\sim$ be the equivalence relation defined on V by $v \sim v'$ if and only if $\psi(v) = \psi(v')$. Let $\widetilde{V} = V/\sim$ be the quotient set of V by $\sim$ and let $\widetilde{E}$ be defined by $\{([u]_\sim, i), ([v]_\sim, j)\} \in \widetilde{E}$ if and only if $\{(u, i), (v, j)\} \in E$, where $[u]_\sim$ denotes the equivalence class of u for the relation $\sim$. It can easily be checked that the quotient graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \hat{\ell})$ is well defined (i.e. $\widetilde{E}$ is a partition of $P(\widetilde{G})$). The surjectivity of ψ clearly entails that $\widetilde{G}$ is isomorphic to $\hat{G}$, and the fact that the sets $\psi^{-1}(\{v\})$ for each $v \in V$ have the same cardinality formalizes our intuition that G is made of copies of $\hat{G}$.

The example showing that finite support series are not recognizable on the family of circular strings can be generalized in the following way. Let M be a GWM and let G be a graph such that $f_M(G) \neq 0$. It is easy to check that the computation of M on the graph $G' = G \cup G$ (obtained by juxtaposing two copies of G) boils down to $f_M(G') = f_M(G)^2$. Hence there exists a covering of G (the graph G') satisfying $f_M(G') \neq 0$. Moreover, it

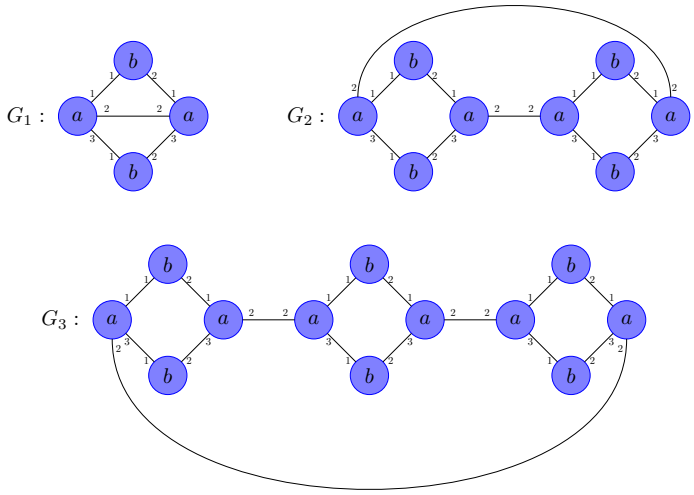


Figure 2.9: Each graph G_i is constructed by copying the initial graph ($G = G_1$) i times, splitting the edge between the two vertices labeled by a in each copy, and reconnecting the free ports to obtain a circular chain of copies of G.

can be shown that if G contains a cycle, then there exists a *connected* covering G' of G such that $f(G') \neq 0$. The construction consists in connecting copies of G by breaking the same edge in each copy of G and regrouping the freed ports into edges connecting the different copies (e.g. take two copies of the circular string a, split the edge in both copies and reconnect the ports to obtain the circular string aa). Since G contains a cycle this can be done in such a way that G' is connected (choose the edge to split in a cycle). If we denote by G_i the graph obtained by this process from i copies of the initial graph (see Figure 2.9), one can show that there exists a matrix $\mathbf{M}$ such that $f_M(G_i) = \mathrm{Tr}(\mathbf{M}^i)$. The trace argument from Lemma 2 can then be used to exhibit a contradiction, thus showing that for any graph G containing a cycle and any recognizable series f such that $f(G) \neq 0$, there exists a connected graph G' distinct from G (i.e. not isomorphic to G) such that $f(G') \neq 0$

2.4.2 Finite Support Series and Graph Coverings

We say that a family of graph $\mathcal{S}$ is *covering-free* if for any graph G in $\mathcal{S}$ the only coverings of G in $\mathcal{S}$ are graphs that are isomorphic to G, i.e. for all $G \in \mathcal{S}$ there are no non-trivial coverings of G in $\mathcal{S}$. We now show that finite support series are recognizable on *covering free* families of graphs. Our first result (Corollay 3) shows that it is true for GWMs *with complex coefficients* defined over any covering free family of graphs. This result relies on the following theorem showing that for a given graph $\hat{G}$, one can build a GWM whose support is exactly the set of coverings of $\hat{G}$.

Theorem 2. *Given a closed and connected graph $\hat{G} = (\hat{V}, \hat{E}, \hat{l})$ over $\mathcal{F} = (\Sigma, \sharp)$, there exists a GWM M with $|P(\hat{G})|$ states and* with coefficients in $\mathbb{C}$ *such that for any closed and connected graph G, $f_M(G) \neq 0$ if and only if G is a covering of $\hat{G}$.*

Proof. For any symbol $x \in \Sigma$, we denote by $\hat{V}(x)$ the set of vertices in $\hat{V}$ labeled by

x. Let $n = |P(\hat{G})|$. Instead of indexing the canonical basis $(\mathbf{e}_1, \cdots, \mathbf{e}_n)$ of $\mathbb{R}^n$ with integers in $[n]$, we will index it with elements of $P(\hat{G})$. Let $M = (\mathbb{R}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma}, \diamond)$ be the $\diamond$-GWM defined by

$$\boldsymbol{\mathcal{M}}^x = \begin{cases} \sum_{\hat{v}\in\hat{V}(x)} \mathbf{e}_{(\hat{v},1)} \circ \mathbf{e}_{(\hat{v},2)} \circ \cdots \circ \mathbf{e}_{(\hat{v},\sharp\hat{v})} & \text{if } \hat{V}(x) \neq \emptyset \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

$$\mathbf{e}_p \diamond \mathbf{e}_{p'} = \begin{cases} 1 & \text{if } \{p, p'\} \in \hat{E} \\ 0 & \text{otherwise.} \end{cases}$$

for all $x \in \Sigma$, and all $p, p' \in P(\hat{G})$.

Let $G = (V, E, \ell)$ be a closed graph on $\mathcal{F}$, we have

$$f_M(G) = \sum_{\gamma\in P(\hat{G})^{P(G)}} \boldsymbol{\mathcal{M}}_\gamma \prod_{\{p,p'\}\in E} \mathbf{e}_{\gamma(p)} \diamond \mathbf{e}_{\gamma(p')} \tag{2.2}$$

where $\boldsymbol{\mathcal{M}}_\gamma = \prod_{v\in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$. We claim that $f_M(G) \neq 0$ if and only if G is a covering of $\hat{G}$, which will imply the result of the theorem by Proposition 1.

Since all the components of the tensors $\{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma}$ are non-negative, we have $f_M(G) \neq 0$ if and only if there exists a $\gamma : P(G) \to P(\hat{G})$ such that (i) $\boldsymbol{\mathcal{M}}_\gamma \neq 0$ and (ii) $\prod_{\{p,p'\}\in E} \mathbf{e}_{\gamma(p)} \diamond \mathbf{e}_{\gamma(p')} \neq 0$. We show that these two conditions are satisfied if and only if there exists a covering map from G to $\hat{G}$. Let $\gamma : P(G) \to P(\hat{G})$ be such that (i) and (ii) are satisfied. Since γ satisfies (i), it follows from the definition of the tensors $\{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma}$ that for all $v \in V$, there exists a vertex $\hat{v} \in \hat{V}$ such that $\ell(v) = \hat{\ell}(\hat{v})$ and $\gamma(v, i) = (\hat{v}, i)$ for all $i \in [\sharp v]$. Let $\psi : v \mapsto \hat{v}$ be the mapping defined by this relation. It is easy to check that ψ is a covering map from G to $\hat{G}$. Indeed, let $\{(v, i), (v', i')\} \in E$, then (ii) implies that $\mathbf{e}_{\gamma(v,i)} \diamond \mathbf{e}_{\gamma(v',i')} \neq 0$, which by definition of $\diamond$ is equivalent to $\{\gamma(v, i), \gamma(v', i')\} \in \hat{E}$, hence $\{(\psi(v), i), (\psi(v'), i')\} \in \hat{E}$. Conversely, let ψ be a covering map from G to $\hat{G}$, and let $\gamma : P(G) \to P(\hat{G})$ be defined by $\gamma(v, i) = (\psi(v), i)$ for all $v \in V$, $i \in [\sharp v]$. It can easily be checked that (i) is satisfied since $\psi(v) \in \hat{V}(\ell(v))$ for all $v \in V$, and (ii) is satisfied since $\{\gamma(v, i), \gamma(v', i')\} = \{(\psi(v), i), (\psi(v'), i')\} \in \hat{E}$ for all $\{(v, i), (v', i')\} \in E$. □

Corollary 3. *Let $\mathcal{S}$ be a covering-free family of closed and connected graphs. Then any finite support series on $\mathcal{S}$ is recognizable by a GWM* with coefficients in the field $\mathbb{C}$.

Proof. Let $\hat{G}$ be a graph and let M be the $\diamond$-GWM from the proof of Theorem 2. For any $\lambda \in \mathbb{C}$, there exists a GWM $\tilde{M}$ such that $f_{\tilde{M}}(\hat{G}) = \lambda$. Indeed, let $\lambda' = f_M(\hat{G})$. It is easy to check that the GMW obtained by replacing each tensor $\boldsymbol{\mathcal{M}}^x$ from M by $\left(\frac{\lambda}{\lambda'}\right)^{1/|\hat{V}|} \boldsymbol{\mathcal{M}}^x$ for all $x \in \Sigma$ assigns the value λ to $\hat{G}$. The result then directly follows from Proposition 1 and Proposition 5. □

After showing on a simple example that finite support series are not GWM-recognizable on the whole set of graphs $\mathfrak{G}_\mathcal{F}$ in general, we managed to identify a large class of families of graphs for which finite support series are recognizable: covering free families. The results presented above state that any finite support function defined on such a family of graphs is recognizable by a GWM *with coefficients in* $\mathbb{C}$. In particular, even if a finite

support function takes its values in $\mathbb{R}$ there are cases where any GWM computing it will necessary have complex coefficients. We go back again to the following simple example: consider the graph $G = a - a$ where a is a symbol of arity one, it is easy to check that any GWM with real coefficients will assign a non-negative value to G. Thus there does not exist any GWM A with real coefficients such that e.g. $f_A(G) = -1$. However, we would ideally like to have a similar result that holds for real valued graph series using GWMs with coefficients in $\mathbb{R}$. The second result of this section (Corollary 5) shows that if we only consider covering-free families *of in-out graphs*, then real-valued finite support series are recognizable by GWMs with real coefficients (note that the families of strings, trees and 2d-words are such families). This result relies on the following theorem.

Theorem 4. *Let $\mathcal{S}$ be a family of closed and connected in-out graphs over a ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$, and let $\hat{G} = (\hat{V}, \hat{E}, \hat{l})$ be a graph in $\mathcal{S}$. For any $G \in \mathcal{S}$, let Ψ_G be the set of covering maps from G to $\hat{G}$.*

Then, for any set $\{\epsilon_{\hat{v}} : \hat{v} \in \hat{V}\} \subset \mathbb{R} \setminus \{0\}$ of non-zero real numbers, there exists a GWM M with $|\hat{E}|$ states and with coefficients in $\mathbb{R}$ *such that*

$$f_M(G) = \sum_{\psi \in \Psi_G} \prod_{v \in V} \epsilon_{\psi(v)} \qquad \text{for all graphs } G = (V, E, \ell) \in \mathcal{S}.$$

Proof. The complete proof is given in Appendix 2.A. We present here the construction of the GWM satisfying the result of the theorem along with the main arguments used in the proof.

For any symbol $x \in \Sigma$, we denote by $\hat{V}(x)$ the set of vertices in $\hat{V}$ labeled by x. Let $n = |\hat{E}|$. Instead of indexing the canonical basis of $\mathbb{R}^n$ with integers in $[n]$, we will index it with elements of $\hat{E}$. Let $\eta : P(\hat{G}) \to \hat{E}$ be the mapping associating each port $p \in P(\hat{G})$ with the edge in $\hat{E}$ it belongs to, that is $\eta(p)$ is the unique $e \in \hat{E}$ such that $p \in e$.

Let $M = \langle \mathbb{R}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma} \rangle$ be the GWM defined by

$$\boldsymbol{\mathcal{M}}^x = \sum_{\hat{v} \in \hat{V}(x)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})})$$

for all $x \in \Sigma$. For any graph $G = (V, E, \ell)$, we have $f_M(G) = \sum_{\gamma \in \Gamma_{id}(G,M)} \boldsymbol{\mathcal{M}}_\gamma$ where $\boldsymbol{\mathcal{M}}_\gamma = \prod_{v \in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$.

Let $\Gamma_G = \{\gamma \in \Gamma_{id}(G, M) : \boldsymbol{\mathcal{M}}_\gamma \neq 0\}$ be the set of γ's corresponding to non-zero summands in $f_M(G)$. We first show in the appendix that for any $\gamma \in \Gamma_G$, there exists a *unique* $\hat{v} \in \hat{V}$ such that $\ell(v) = \hat{\ell}(\hat{v})$ and $\gamma(v, i) = \eta(\hat{v}, i)$ for all $i \in [\sharp v]$. Let $\psi_\gamma : v \mapsto \hat{v}$ be the mapping defined by this relation. For any $\gamma \in \Gamma_G$ and any $v \in V$ it follows that

$$\begin{aligned}
\boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} &= \left(\sum_{\hat{v} \in \hat{V} \,:\, \ell(\hat{v}) = \ell(v)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})}) \right)_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \\
&= \sum_{\hat{v} \in \hat{V} \,:\, \ell(\hat{v}) = \ell(v)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})})_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \\
&= \epsilon_{\psi_\gamma(v)} \ .
\end{aligned}$$

Hence,

$$f_M(G) = \sum_{\gamma \in \Gamma_{id}(G,M)} \mathcal{M}_\gamma = \sum_{\gamma \in \Gamma_G} \mathcal{M}_\gamma = \sum_{\gamma \in \Gamma_G} \prod_{v \in V} \mathcal{M}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} = \sum_{\gamma \in \Gamma_G} \prod_{v \in V} \epsilon_{\psi_\gamma(v)} \ .$$

The proof then relies on showing that Γ_G is in one-to-one correspondence with the set of covering maps Ψ_G through the mapping $\gamma \mapsto \psi_\gamma$, which entails the result.

□

Corollary 5. *Let $\mathcal{S}$ be a covering-free family of in-out closed and connected graphs. Then any real-valued finite support series on $\mathcal{S}$ is recognizable by a GWM* with coefficients in the field $\mathbb{R}$.

Proof. We want to show that for any graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\ell}) \in \mathcal{S}$ and any real number λ, there exists a GWM A with coefficients in $\mathbb{R}$ such that $f_A(\hat{G}) = \lambda$ and $f_A(G) = 0$ for any $G \neq \hat{G}$. First note that the strategy used in the proof of Corollary 3, that consists in multiplying each tensor of the GWM M from the previous proof by the scalar $s = (\lambda/\lambda')^{1/|\hat{V}|}$ where $\lambda' = f_M(\hat{G})$, cannot be applied here since s may be a complex number.

Let $\hat{v}_0$ be any vertex of $\hat{G}$. Let M be the GWM from Theorem 4 with $\epsilon_{\hat{v}} = 1$ for all $\hat{v} \neq \hat{v}_0$ and $\epsilon_{\hat{v}_0} = \frac{\lambda}{|\Psi_{\hat{G}}|}$. We have $f_M(G) = \sum_{\psi \in \Psi_G} \prod_{v \in V} \epsilon_{\psi(v)}$. Since $\mathcal{S}$ is a covering-free family, we have $f_M(G) \neq 0$ if and only if $G = \hat{G}$. Furthermore, we have $f_M(\hat{G}) = \sum_{\psi \in \Psi_{\hat{G}}} \prod_{\hat{v} \in \hat{V}} \epsilon_{\psi(\hat{v})}$. Since any $\psi \in \Psi_{\hat{G}}$ is a permutation of $\hat{V}$, $\epsilon_{\hat{v}_0}$ appears only once in $\prod_{\hat{v} \in \hat{V}} \epsilon_{\psi(\hat{v})}$, hence $f_M(\hat{G}) = \sum_{\psi \in \Psi_{\hat{G}}} \epsilon_{\hat{v}_0} = \lambda$.

The result then directly follows from Proposition 5. □

It is worth mentioning that the only property of in-out graphs we used in the proof of Theorem 4 is that an in-out graph does not contain any edge $\{(u,i),(v,i)\}$ such that $\ell(u) = \ell(v)$. Thus Corollary 5 would also hold for any family of graphs satisfying this property.

Discussion. Before going into the second part of this chapter, we will summarize the various results that we obtained until now. We introduced GWMs that extend the notion of recognizable functions on strings, trees and pictures to general graphs on a ranked alphabet. The definition of this computational model on graphs was quite natural once we interpreted the computations of classical weighted automata in terms of tensor networks. Even though it was easy to show that GWMs naturally generalize recognizable functions on strings, trees and pictures, we showed on simple examples that some of the fundamental properties of these classical models were not satisfied by GWMs defined over arbitrary families of graphs. We thus tried to identify smaller families of graphs for which these properties hold; we summarize our results below.

1. GWM-recognizable functions are closed under sum and Hadamard product *for any family of connected graphs*.

2. GWM-recognizable functions defined on *rooted families of graphs* are closed under scalar multiplication.

3. Finite support series defined on *covering free families of (connected) graphs* are recognizable by GWMs *with complex coefficients*.

4. *Real-valued* finite support series defined on *covering free families of in-out (connected) graphs* are recognizable by GWMs *with real coefficients*.

5. Real-valued recognizable functions *can not all be computed by a GWM with real coefficients* but we conjectured that this is the case for real-valued recognizable functions defined over families of in-out graphs.

It is easy to check that rooted families of graphs are covering free. Indeed, non trivial coverings of a graph are obtained by duplicating this graph, which results in a graph where the special root symbol appears more than once. It follows that any family of rooted in-out graphs satisfies all the desirable properties listed above (assuming our conjecture is true). We thus managed to identify two simple structural constraints on graphs (which are satisfied by strings, trees and 2d-words[a]) that allow us to define large families of graphs on which GWMs inherit several fundamental properties from the classical notion of recognizable functions on strings and trees.

We conclude this section by observing that the 4th result mentioned above is somehow a first step towards proving the conjecture stated in result 5. Indeed, we think that the two conditions on the family on graphs in result 4 are related to different aspects: the covering free assumption allows us to obtain the recognizability of finite support series, while the in-out graphs assumption allows us to show that real-valued (finite support) functions can be computed by GWMs with real weights.

2.5 Learning GWMs over Circular Strings.

In this section, we present preliminary results on learning GWMs by studying the problem of learning GWMs defined on the family of circular strings. While circular strings do not seem to be of particular practical interest, they can be seen as the simplest family of graphs with cycles that is not covering free, which makes them an interesting family to study from a theoretical perspective. Even though this family is very simple (compared to the whole set of graphs $\mathfrak{G}_{\mathcal{F}}$) we will see that learning GWMs defined over circular strings already is a challenging task.

We will consider the learning paradigm of *identification in the limit* (Gold, 1967). In this framework, the problem consists in recovering a target function after receiving a finite number of examples: given a set of input/output examples

$$\{(w_1, f_A(w_1)), \cdots, (w_N, f_A(w_N))\} \subset \Sigma^+ \times \mathbb{R}$$

where f_A is the function computed by a GWM $A = (\mathbb{F}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x\in\Sigma})$, the learner has to identify a GWM $\hat{A} = (\mathbb{R}^{\hat{n}}, \{\hat{\mathbf{M}}^\sigma\}_{\sigma\in\Sigma})$ such that $f_A = f_{\hat{A}}$. A class of functions is said to identifiable in the limit if there exists an algorithm that identifies any function in the class after examining a finite set of examples.

[a] As we defined it the family of 2d-words is not a rooted family of graphs but one could systematically replace one of the labels of a border vertex by a special root symbol to make it a rooted family.

Recall that for any word $w = \sigma_1 \cdots \sigma_k \in \Sigma^+$ the circular string $circ(w)$ is the graph obtained by closing w onto itself, and that a GWM $A = (\mathbb{R}^n, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma})$ computes the function $circ(w) \mapsto \mathrm{Tr}(\mathbf{A}^w)$ where $\mathbf{A}^w = \mathbf{A}^{\sigma_1} \cdots \mathbf{A}^{\sigma_k}$. Hence the problem of learning GWMs defined over circular strings boils down to *recovering matrices from a set of trace observables*. We will first show how functions that can be computed by GWMs on circular strings are related to classical recognizable string series. This will suggest a first approach to learn GWMs on circular strings by using classical learning algorithms for string weighted automata (WA). However, this learning approach is improper in the sense that it does not allow one to recover a GWM computing the target function but rather a WA that computes it. We will then propose an alternative learning method relying on tensor decomposition techniques that directly tries to learn the target function as a GWM and seems more promising to be extended to the problem of learning GWMs over more general families of graphs. In the following, we consider GWMs with real coefficients but the results we present can easily be extended to the case of GWMs with complex coefficients.

2.5.1 GWM on Circular Strings and Weighted Automata

The following proposition shows that a function f computed by a GWM on circular strings can be computed by a string weighted automaton with a quadratic number of states.

Proposition 9. *For any GWM $M = (\mathbb{R}^n, \{\mathbf{M}^\sigma\}_{\sigma\in\Sigma})$ over circular strings on Σ, there exists a string weighted automaton $A = (\mathbb{R}^{n^2}, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ such that $f_M(circ(w)) = f_A(w)$ for all $w \in \Sigma^*$.*

Proof. For any $w = w_1 \cdots w_k \in \Sigma^*$ we have $f_M(circ(w)) = Tr(\mathbf{M}^w) = \sum_{i\in[n]} \mathbf{M}^w_{i,i} = \sum_{i\in[n]} \mathbf{e}_i^\top \mathbf{M}^w \mathbf{e}_i$ where $\mathbf{M}^w = \mathbf{M}^{w_1} \cdots \mathbf{M}^{w_k}$ and $\mathbf{e}_i$ is the ith vector of the canonical basis of $\mathbb{R}^n$. Let $\boldsymbol{\alpha} = \boldsymbol{\omega} = (\mathbf{e}_1^\top, \cdots, \mathbf{e}_n^\top)^\top \in \mathbb{R}^{n^2}$ and let $\mathbf{A}^\sigma = \mathbf{I}_n \otimes \mathbf{M}^\sigma \in \mathbb{R}^{n^2\times n^2}$ be the block-diagonal matrix with $\mathbf{M}^\sigma$ repeated n times on the diagonal. We have

$$f_A(w) = \boldsymbol{\alpha}^\top \mathbf{A}^w \boldsymbol{\omega} = \begin{bmatrix} \mathbf{e}_1^\top & \mathbf{e}_2^\top & \cdots & \mathbf{e}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{M}^w & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^w & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M}^w \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_n \end{bmatrix}$$
$$= \sum_{i\in[n]} \mathbf{e}_i^\top \mathbf{M}^w \mathbf{e}_i = f_M(circ(w)).$$

□

It follows from the construction used in the proof of the previous proposition that for any GWM $M = (\mathbb{R}^n, \{\mathbf{M}^\sigma\}_{\sigma\in\Sigma})$ the weighted automaton (WA)

$$A = (\mathbb{R}^{n^2}, \boldsymbol{\alpha} = \mathrm{vec}(\mathbf{I}_{n^2}), \{\mathbf{A}^\sigma = \mathbf{I}_n \otimes \mathbf{M}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega} = \mathrm{vec}(\mathbf{I}_{n^2}))$$

is such that $f_M(circ(w)) = f_A(w)$ for all $w \in \Sigma^+$. Here $\mathbf{I}_n \otimes \mathbf{M} = \mathbf{M} \oplus \cdots \oplus \mathbf{M}$ is the block diagonal matrix with n diagonal blocks all equal to $\mathbf{M}$. Similarly, we have $\boldsymbol{\alpha} = \boldsymbol{\omega} = \mathbf{e}_1 \oplus \cdots \oplus \mathbf{e}_n$ where $\mathbf{e}_i$ is the ith vector of the canonical basis. Thus the

computation of an n-dimensional GWM is equivalent to the sum of n WAs with n states. These n WAs are identical except for their initial and final weights: each one has a unique both initial and final state with weight one. Thus, the value assigned to a string is the sum of the weights of all its paths in the WA starting and ending in the same state. It is the internal dynamic of this WA (and not its initial and final weights) that is relevant to the computation of the GWM on circular strings.

From a learning perspective, Proposition 9 suggests an approach to learn GWM-recognizable functions over circular strings by using learning algorithms for weighted automata. It is well known that WAs are identifiable in the limit, thus one could recover a function computed by a GWM with n states from a finite set of examples in the form of a weighted automata computing the same function. However, this WA could have up to n^2 states and recovering a GWM with n states computing the same function from this WA is not a trivial task. Suppose first that we are given the WA A with n^2 states and that we want to construct a GWM on circular strings that computes the same function. One way to proceed would be to leverage the fact that all the matrices $\mathbf{A}^\sigma$ of this WA are simultaneously block diagonalizable as

$$\mathbf{A}^\sigma = \mathbf{P}(\mathbf{I} \otimes \hat{\mathbf{M}}^\sigma)\mathbf{P}^{-1}$$

for some invertible $n^2 \times n^2$ matrix $\mathbf{P}$ and $n \times n$ matrices $\hat{\mathbf{M}}^\sigma$ that are each similar to the matrices $\mathbf{M}^\sigma$ from the target GWM (which follows from the proof of Proposition 9 and Theorem 2.4 in Berstel and Reutenauer, 1988). Existing methods for simultaneous block diagonalization could be used to recover the matrices $\hat{\mathbf{M}}^\sigma$ (Maehara and Murota, 2011; Murota et al., 2010; Klerk, Dobre, and Pasechnik, 2011; Ghennioui et al., 2007; Nion, 2011) from which one could easily construct a GWM computing the same function. However if the WA A is not minimal, we may only have access to a WA B with $k < n^2$ that computes the same function as A (and thus as the target GWM M). In this case the matrices $\mathbf{B}^\sigma$ are not simultaneously block diagonalizable anymore (even though there exists a $k \times n^2$ matrix $\mathbf{P}$ such that $\mathbf{B}^\sigma = \mathbf{P}(\mathbf{I} \otimes \hat{\mathbf{M}}^\sigma)\mathbf{P}^+$) and the simultaneous block diagonalization methods mentioned above cannot be straightforwardly applied. This minimality condition on the WA A is closely related to the notion of *full rank GWMs* that we will define in the next section, where we propose an alternative approach relying on tensor decomposition techniques that has the benefit of not going through the intermediate step of learning a weighted automata with a potentially quadratic number of states. Before that, we show in the following proposition that GWM-recognizable functions defined over circular strings are invariant under a change a basis of the operators of a GWM computing it.

Proposition 10. *Let $A = (\mathbb{R}^n, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma})$ and $B = (\mathbb{R}^n, \{\mathbf{B}^\sigma\}_{\sigma\in\Sigma})$ be two GWMs over circular strings such that $\mathbf{A}^\sigma = \mathbf{P}\mathbf{B}^\sigma\mathbf{P}^{-1}$ for all $\sigma \in \Sigma$ for some invertible matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$. Then $f_A(circ(w)) = f_B(circ(w))$ for all $w \in \Sigma^+$.*

Proof. This directly follows from the fact that $\mathbf{A}^w = \mathbf{P}\mathbf{B}^w\mathbf{P}^{-1}$ for all $w \in \Sigma^+$ and from the invariance of the trace under similarity transformation. □

The previous proposition could easily be extended to any family on in-out graphs: multiplying each mode of the tensors of a GWM corresponding to an in-port by some matrix $\mathbf{P}$ and each mode corresponding to an out-port by $\mathbf{P}^{-1}$ would result in a GWM

computing the same function on in-out graphs (since every in-port is connected to an out-port, $\mathbf{P}$ and $\mathbf{P}^{-1}$ would cancel out everywhere in the computations of this new GWM).

2.5.2 Learning GWMs with Tensor Decomposition

In this section, we propose a learning algorithm for GWMs on circular strings that relies on decomposing (a sub-block of) the 3rd order Hankel tensor $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{\Sigma^* \times \Sigma^* \times \Sigma^*}$ defined by

$$\boldsymbol{\mathcal{H}}_{x,y,z} = f(circ(xyz))$$

for all $x, y, z \in \Sigma^*$.

Linear presever of traces of triple products. We first prove a general result on *linear preservers* of traces of triple products that is key to the learning method we propose. Linear preserver problems concern the characterization of operators on matrix spaces that leave certain functions invariant (see e.g. (C.-K. Li and S. Pierce, 2001) for an introduction). For example, Proposition 1.1 in (Chan, C.-K. Li, and Sze, 2007) shows that for any (not necessarily linear) mapping $\phi : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, if $\mathrm{Tr}(\mathbf{AB}) = \mathrm{Tr}(\phi(\mathbf{A})\phi(\mathbf{B}))$ for all matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ then ϕ is linear and invertible.

In the following theorem we will extend this result to trace of triple products by showing that if some operator on a matrix space leaves traces of triple products invariant, then this operator is a simple similarity transformation. The relevance to the problem of learning GWMs comes from Proposition 10: if we are able to recover matrices $\hat{\mathbf{M}}^\sigma$ for $\sigma \in \Sigma$ that are similar to the matrices $\mathbf{M}^\sigma$ of the target GWM, then we would obtain a GWM computing the same function as the target.

Theorem 6. *Let $S \subseteq \mathbb{R}^{n\times n}$ be a subset of $\mathbb{R}^{n\times n}$ such that $span(S) = \mathbb{R}^{n\times n}$ and $\mathbf{I} \in S$. If $\phi : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is a (not necessarily linear) mapping satisfying*

$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\phi(\mathbf{A})\phi(\mathbf{B})\phi(\mathbf{C})) \tag{2.3}$$

for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \in S$, then ϕ is linear and of the form $\phi(\mathbf{X}) = \mathbf{PXP}^{-1}$ for some invertible matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$.

Proof. We first show that ϕ is linear and invertible. Let $(\mathbf{E}_1, \cdots, \mathbf{E}_{n^2}) \subseteq S$ be a basis of $\mathbb{R}^{n\times n}$. Let $\mathbf{M} \in \mathbb{R}^{n^2\times n^2}$ be the matrix with rows $\mathrm{vec}(\mathbf{E}_i)^\top$, and let $\mathbf{M}_\phi \in \mathbb{R}^{n^2\times n^2}$ be the matrix with rows $\mathrm{vec}(\phi(\mathbf{E}_i)\phi(\mathbf{I}))^\top$. It follows from (2.3) and from the identity $\mathrm{Tr}(\mathbf{ABC}) = \mathrm{vec}(\mathbf{AB})^\top \mathrm{vec}(\mathbf{C}^\top)$ that

$$\mathbf{M}\mathrm{vec}(\mathbf{X}^\top) = \mathbf{M}_\phi \mathrm{vec}(\phi(\mathbf{X})^\top)$$

for all $\mathbf{X} \in S$. Let $\mathbf{N} \in \mathbb{R}^{n^2\times n^2}$ be the matrix with columns $\mathrm{vec}(\mathbf{E}_i^\top)$ and let $\mathbf{N}_\phi \in \mathbb{R}^{n^2\times n^2}$ be the matrix with columns $\mathrm{vec}(\phi(\mathbf{E}_i)^\top)$, we have $\mathbf{MN} = \mathbf{M}_\phi\mathbf{N}_\phi$ where $\mathbf{M}$ and $\mathbf{N}$ are invertible, hence $\mathbf{M}_\phi$ is invertible. It follows that

$$\mathrm{vec}(\phi(\mathbf{X})^\top) = \mathbf{M}_\phi^{-1}\mathbf{M}\mathrm{vec}(\mathbf{X}^\top)$$

for all $\mathbf{X} \in S$, thus ϕ is linear and invertible on $span(S) = \mathbb{R}^{n\times n}$. Since both ϕ and the trace operator are linear, Eq. (2.3) extends to all matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n\times n}$.

We now show that ϕ preserves the matrix product. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$. For any $\mathbf{B} \in \mathbb{R}^{n\times n}$ we have

$$\mathrm{Tr}(\phi(\mathbf{A})\phi(\mathbf{I})\phi(\mathbf{B})) = \mathrm{Tr}(\mathbf{AIB}) = \mathrm{Tr}(\mathbf{IAB}) = \mathrm{Tr}(\phi(\mathbf{I})\phi(\mathbf{A})\phi(\mathbf{B})).$$

Since the linear span of the $\phi(\mathbf{B})$'s is the whole space $\mathbb{R}^{n\times n}$, we have $\phi(\mathbf{A})\phi(\mathbf{I}) = \phi(\mathbf{I})\phi(\mathbf{A})$ for any $\mathbf{A} \in \mathbb{R}^{n\times n}$, thus $\phi(\mathbf{I}) = \lambda\mathbf{I}$ for some $\lambda \in \mathbb{R}$. Using (2.3) we can deduce $\lambda^3 = 1$, hence $\lambda = 1$.

It follows that for any $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n\times n}$ we have

$$\begin{aligned}\mathrm{Tr}(\phi(\mathbf{A})\phi(\mathbf{B})\phi(\mathbf{C})) &= \mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}((\mathbf{AB})\mathbf{CI}) \\ &= \mathrm{Tr}(\phi(\mathbf{AB})\phi(\mathbf{C})\phi(\mathbf{I})) = \mathrm{Tr}(\phi(\mathbf{AB})\phi(\mathbf{C})),\end{aligned}$$

and since the $\phi(\mathbf{C})$'s span $\mathbb{R}^{n\times n}$ we have $\phi(\mathbf{A})\phi(\mathbf{B}) = \phi(\mathbf{AB})$ for all $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$.

To conclude, we have that ϕ is an automorphism of the algebra $\mathbb{R}^{n\times n}$ (it is linear, bijective (since it is invertible), and preserves the matrix product), and it is well known that every automorphism of the full matrix algebra is inner[b], that is $\phi(\mathbf{X}) = \mathbf{PXP}^{-1}$ for some invertible matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$. □

A tensor decomposition approach. We now show how Theorem 6 allows us to reduce the problem of learning GWMs over circular strings to a constrained tensor decomposition of the 3rd order Hankel tensor $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{\Sigma^*\times\Sigma^*\times\Sigma^*}$. First observe that the hypothesis $span(S) = \mathbb{R}^{n\times n}$ of the previous theorem requires that we have access to the trace observables of a set of matrices that generate the full matrix algebra, which is not generally the case for the matrices of a GWM. This leads us to first define the notion of full rank GWMs.

Given a GWM $A = (\mathbb{R}^n, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma})$ defined on circular strings, we will say that it is of *full rank* if there exists a set of words $W \subset \Sigma^*$ such that the vector space spanned by the matrices $\mathbf{A}^w$ for $w \in W$ has dimension n^2 (where $\mathbf{A}^w = \mathbf{A}^{w_1}\mathbf{A}^{w_2}\cdots\mathbf{A}^{w_{|w|}}$), i.e. $span(\{\mathbf{A}^w : w \in W\}) = \mathbb{R}^{n\times n}$. We will call such a set of words W a *basis* for the full rank GWM A. Note that not every GWM defined over circular strings is of full rank. In particular, it is easy to check that $dim(span(\{\mathbf{A}^k \mid k \geq 0\})) \leq n$ for any matrix $n \times n$ matrix $\mathbf{A}$, hence a GWM defined over circular strings on a one-letter alphabet cannot be of full rank. However, the following proposition shows that *generic* GWMs (i.e. GWMs whose parameters $\{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}$ are drawn at random from a continuous distribution over $\mathbb{R}^{n\times n}$) defined over circular strings on an alphabet of size at least 2 are of full rank.

Proposition 11. *The set* $\{\mathbf{A}^a, \mathbf{A}^b \in \mathbb{R}^{n\times n} : dim(span(\{\mathbf{A}^w : w \in \{a,b\}^*\})) < n^2\}$ *has Lebesgue measure 0.*

Proof. For any couple of matrices $(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$, let

$$P(\mathbf{A}, \mathbf{B}) = \{\mathbf{A}^i\mathbf{B}^j : 0 \leq i, j < n\} \quad \text{and}$$
$$S = \{(\mathbf{A}, \mathbf{B}) : P(\mathbf{A}, \mathbf{B}) \text{ is not a basis of } \mathbb{R}^{n\times n}\}.$$

[b] This is a special case of the Skolem-Noether theorem (see (Semrl, 2006), (Alperin, 1993, Ex. 4 p. 12) or (R. S. Pierce, 1982, Lemma p. 230)).

We will show that S has Lebesgue measure zero in $\mathbb{R}^{n\times n}\times\mathbb{R}^{n\times n}$, which will entail the result. For any couple of matrices $(\mathbf{A},\mathbf{B})$ let $M(\mathbf{A},\mathbf{B})\in\mathbb{R}^{n^2\times n^2}$ be the matrix whose columns are the vectorizations of $\mathbf{A}^i\mathbf{B}^j$ for $0\leq i,j<n$. Then $(\mathbf{A},\mathbf{B})\in S$ if and only if the determinant $det(M(\mathbf{A},\mathbf{B}))=0$. Since $det(M(\mathbf{A},\mathbf{B}))$ is a polynomial in the entries of $\mathbf{A}$ and $\mathbf{B}$, S is an algebraic subvariety of $(\mathbb{R}^{n\times n})^2$. We now show that the polynomial $det(M(\mathbf{A},\mathbf{B}))$ is not uniformly 0. Let $\mathbf{U}$ be the upper $n\times n$ shift matrix (i.e. $\mathbf{U}$ is the matrix with ones only on the superdiagonal and zeroes elsewhere). We claim that $det(M(\mathbf{U},\mathbf{U}^\top))\neq 0$, which is equivalent to $P(\mathbf{U},\mathbf{U}^\top)$ being a basis of $\mathbb{R}^{n\times n}$. Indeed, let $(\mathbf{E}^{i,j})_{1\leq i,j\leq n}$ be the canonical basis of $\mathbb{R}^{n\times n}$ (i.e. $\mathbf{E}^{i,j}=\mathbf{e}_i\mathbf{e}_j^\top$), one can check that

$$\mathbf{E}^{i,j}=\mathbf{U}^{n-i}(\mathbf{U}^\top)^{n-j}-\mathbf{U}^{n-i+1}(\mathbf{U}^\top)^{n-j+1}$$

for all $1\leq i,j\leq n$ (note that $\mathbf{U}^{n-i+1}(\mathbf{U}^\top)^{n-j+1}$ is either in $P(\mathbf{U},\mathbf{U}^\top)$ or equal to 0). In conclusion, S is a proper algebraic subvariety of $(\mathbb{R}^{n\times n})^2$ and hence has Lebesgue measure zero (Federer, 2014, Section 2.6.5). □

Suppose now that given a basis W for a full rank GWM $A=(\mathbb{R}^n,\{\mathbf{A}^\sigma\}_{\sigma\in\Sigma})$ we can find matrices $\hat{\mathbf{A}}^w$ for $w\in W$ such that $\mathrm{Tr}(\mathbf{A}^x\mathbf{A}^y\mathbf{A}^z)=\mathrm{Tr}(\hat{\mathbf{A}}^x\hat{\mathbf{A}}^y\hat{\mathbf{A}}^z)$ for all $x,y,z\in W$. Using Theorem 6, one can show that all couples of matrices $\mathbf{A}^w,\hat{\mathbf{A}}^w$ for $w\in W$ are similar under a same similarity transform. Furthermore, if $\Sigma\subset W$ then one can recover a GWM $\hat{A}$ computing the same function as A. We formally prove this result in the following theorem, where we also show that finding the matrices $\hat{\mathbf{A}}^w$ for $w\in W$ boils down to a constrained decomposition of a sub-block of the 3rd-order Hankel tensor $\boldsymbol{\mathcal{H}}$.

Theorem 7. *Let $M=(\mathbb{R}^n,\{\mathbf{M}^\sigma\}_{\sigma\in\Sigma})$ be a full rank GWM over circular strings on a finite alphabet Σ. Let $W\subset\Sigma^*$ be a basis for M such that $\Sigma\cup\{\varepsilon\}\subset W$ (where we assume $\mathbf{M}^\varepsilon=\mathbf{I}$). Finally, let $\boldsymbol{\mathcal{T}}\in\mathbb{R}^{n^2\times n^2\times n^2}$ be the tensor defined by the relation*

$$\boldsymbol{\mathcal{T}}\bullet_1\mathrm{vec}(\mathbf{A})\bullet_2\mathrm{vec}(\mathbf{B})\bullet_3\mathrm{vec}(\mathbf{C})=\mathrm{Tr}(\mathbf{ABC})$$

for any $\mathbf{A},\mathbf{B},\mathbf{C}\in\mathbb{R}^{n\times n}$, and let $\boldsymbol{\mathcal{H}}^W\in\mathbb{R}^{W\times W\times W}$ be the third order Hankel tensor defined by

$$\boldsymbol{\mathcal{H}}^W_{x,y,z}=f_M(circ(xyz))=\mathrm{Tr}(\mathbf{M}^x\mathbf{M}^y\mathbf{M}^z)$$

for any $x,y,z\in W$.

Then, for any matrix $\mathbf{U}\in\mathbb{R}^{W\times n^2}$ satisfying $\boldsymbol{\mathcal{H}}^W=\boldsymbol{\mathcal{T}}\times_1\mathbf{U}\times_2\mathbf{U}\times_3\mathbf{U}$, the GWM $\hat{M}=(\mathbb{R}^n,\{\hat{\mathbf{M}}^\sigma\}_{\sigma\in\Sigma})$, where $\hat{\mathbf{M}}^\sigma$ is defined by the relation $\mathrm{vec}(\hat{\mathbf{M}}^\sigma)^\top=\mathbf{U}_{\sigma,:}$, computes the same function as M.

Proof. This is a direct consequence of Theorem 6. Indeed, $\boldsymbol{\mathcal{H}}=\boldsymbol{\mathcal{T}}\times_1\mathbf{U}\times_2\mathbf{U}\times_3\mathbf{U}$ implies $\mathrm{Tr}(\mathbf{M}^x\mathbf{M}^y\mathbf{M}^z)=\mathrm{Tr}(\hat{\mathbf{M}}^x\hat{\mathbf{M}}^y\hat{\mathbf{M}}^z)$ for all $x,y,z\in W$. Let $\phi:\mathbb{R}^{n\times n}\to\mathbb{R}^{n\times n}$ be a mapping such that $\phi(\mathbf{M}^x)=\hat{\mathbf{M}}^x$ for all $x\in W$. Since W is a basis for the full rank GWM M, the span of the matrices $\mathbf{M}^w$ for $w\in W$ is the whole matrix algebra $\mathbb{R}^{n\times n}$, we can thus invoke Theorem 6 to show that there exists an invertible matrix $\mathbf{P}$ such that $\mathbf{M}^\sigma=\mathbf{P}^{-1}\hat{\mathbf{M}}^\sigma\mathbf{P}$ for all $\sigma\in\Sigma$. Hence $\hat{M}=(\mathbb{R}^n,\{\mathbf{P}\mathbf{M}^\sigma\mathbf{P}^{-1}\}_{\sigma\in\Sigma})$ and the result follows from Proposition 10. □

Tucker decomposition with constrained core tensor and equal factors. It follows from the previous theorem that given a basis W for a full rank GWM and the tensor $\boldsymbol{\mathcal{H}}^W$,

one can recover a GWM computing the same function by finding a matrix $\mathbf{U}$ satisfying $\boldsymbol{\mathcal{H}}^W = \boldsymbol{\mathcal{T}} \times_1 \mathbf{U} \times_2 \mathbf{U} \times_3 \mathbf{U}$ (where we used the definitions of Theorem 7). We now propose an alternating minimization algorithm to tackle this problem. Note that this problem can be seen as finding a Tucker decomposition of the tensor $\boldsymbol{\mathcal{H}}^W$ where the core tensor is given and all the factor matrices are equal.

We start by reformulating the problem in a more general setting. Given a tensor $\boldsymbol{\mathcal{X}} \in (\mathbb{R}^m)^{\otimes p}$ and a core tensor $\boldsymbol{\mathcal{G}} \in (\mathbb{R}^n)^{\otimes p}$, we want to find a matrix $\mathbf{U} \in \mathbb{R}^{m \times n}$ such that

$$\boldsymbol{\mathcal{X}} \simeq \boldsymbol{\mathcal{G}} \times_1 \mathbf{U} \times_2 \cdots \times_p \mathbf{U}.$$

We first formulate the problem as a (non-convex) minimization problem:

$$\min_{\mathbf{U} \in \mathbb{R}^{m \times n}} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{U} \times_2 \cdots \times_p \mathbf{U}\|_F^2. \tag{2.4}$$

This is equivalent to the following constrained minimization problem:

$$\min_{\mathbf{U}_1, \cdots, \mathbf{U}_p \in \mathbb{R}^{m \times n}} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}_1 \times_2 \cdots \times_p \mathbf{U}_p\|_F^2 \text{ such that } \mathbf{U}_1 = \mathbf{U}_2 = \cdots = \mathbf{U}_p, \tag{2.5}$$

which can be addressed by minimizing the penalized objective function

$$\mathcal{L}(\mathbf{U}_1, \cdots, \mathbf{U}_p) = \frac{1}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{U} \times_2 \cdots \times_p \mathbf{U}\|_F^2 + \frac{\lambda}{2}\left(\sum_{i=2}^{p} \|\mathbf{U}_1 - \mathbf{U}_i\|_F^2\right) \tag{2.6}$$

where the equality constraints in (2.5) are enforced by the right hand term for large enough values of the hyper-parameter λ.

This minimization problem is not jointly convex but it is convex in each variable $\mathbf{U}_1, \cdots, \mathbf{U}_p$, and can be tackled using the alternating least squares (ALS) method. Recall that the ALS algorithm consists in solving problem (2.6) for one of the variable $\mathbf{U}_i$ while keeping the other ones fixed, and using the solution as an update for $\mathbf{U}_i$. This process is repeated in turn for all variables $\mathbf{U}_1, \cdots, \mathbf{U}_p$ until convergence. While there are no guaranty that ALS will converge to a global minimizer, this approach often performs well in practice.

For any $i \in [p]$, let $\mathbf{M}_i = \mathbf{G}_{(i)}(\mathbf{U}_p \otimes \cdots \otimes \mathbf{U}_{i+1} \otimes \mathbf{U}_{i-1} \otimes \cdots \otimes \mathbf{U}_1)^\top$. Using Eq. (1.3) we have $\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{U} \times_2 \cdots \times_p \mathbf{U}\|_F^2 = \|\mathbf{X}_{(i)} - \mathbf{U}_i \mathbf{M}_i\|_F^2$ for any $i \in [p]$, from which we obtain the partial derivatives

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{U}_1} &= \mathbf{U}_1(\mathbf{M}_1\mathbf{M}_1^\top + \lambda \mathbf{I}) - \mathbf{X}_{(1)}\mathbf{M}_1^\top - \lambda \sum_{i=2}^{p} \mathbf{U}_i \\
\frac{\partial \mathcal{L}}{\partial \mathbf{U}_j} &= \mathbf{U}_j(\mathbf{M}_j\mathbf{M}_j^\top + \lambda \mathbf{I}) - \mathbf{X}_{(j)}\mathbf{M}_j^\top - \lambda \mathbf{U}_1 \qquad \text{for } j = 2, \cdots, p.
\end{aligned}$$

Setting these derivatives to zero leads to the following update rules:

$$\mathbf{U}_1 \leftarrow \left(\mathbf{X}_{(1)}\mathbf{M}_1^\top + \lambda \sum_{i=2}^{p} \mathbf{U}_i\right)\left(\mathbf{M}_1\mathbf{M}_1^\top + \lambda\mathbf{I}\right)^{-1}$$
$$\mathbf{U}_j \leftarrow \left(\mathbf{X}_{(j)}\mathbf{M}_1^\top + \lambda\mathbf{U}_1\right)\left(\mathbf{M}_j\mathbf{M}_j^\top + \lambda\mathbf{I}\right)^{-1} \qquad \text{for } j = 2, \cdots, p.$$

The overall procedure to find an approximate solution of problem (2.4) is summarized in Algorithm 1.

Algorithm 1 `Constrained Tucker Decomposition`

Input: Target tensor $\boldsymbol{\mathcal{X}} \in (\mathbb{R}^m)^{\otimes p}$ and core tensor $\boldsymbol{\mathcal{G}} \in (\mathbb{R}^n)^{\otimes p}$.
Output: A matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$ such that $\boldsymbol{\mathcal{X}} \simeq \boldsymbol{\mathcal{G}} \times_1 \mathbf{U} \times_2 \cdots \times_p \mathbf{U}$.

Randomly initialize $\mathbf{U}_1, \cdots, \mathbf{U}_p \in \mathbb{R}^{m\times n}$
repeat
  $\mathbf{M}_1 \leftarrow \mathbf{G}_{(1)}(\mathbf{U}_p \otimes \cdots \otimes \mathbf{U}_2)^\top$
  $\mathbf{U}_1 \leftarrow \left(\mathbf{X}_{(1)}\mathbf{M}_1^\top + \lambda \sum_{i=2}^{p} \mathbf{U}_i\right)\left(\mathbf{M}_1\mathbf{M}_1^\top + \lambda\mathbf{I}\right)^{-1}$
  for $j = 2$ **to** p **do**
    $\mathbf{M}_j \leftarrow \mathbf{G}_{(j)}(\mathbf{U}_p \otimes \cdots \otimes \mathbf{U}_{j+1} \otimes \mathbf{U}_{j-1} \otimes \cdots \otimes \mathbf{U}_1)^\top$
    $\mathbf{U}_j \leftarrow \left(\mathbf{X}_{(j)}\mathbf{M}_1^\top + \lambda\mathbf{U}_1\right)\left(\mathbf{M}_j\mathbf{M}_j^\top + \lambda\mathbf{I}\right)^{-1}$
  end for
until convergence
return $\mathbf{U}_1$

Discussion. We reduced the problem of learning full rank GWMs on circular strings to the problem of finding a constrained Tucker decomposition of a sub-block of the 3rd order Hankel tensor $\mathbf{H} \in \mathbb{R}^{\Sigma^*\times\Sigma^*\times\Sigma^*}$. The learning method we proposed is thus close to spectral learning methods for weighted automata on strings and trees that rely on finding a low rank decomposition of a sub-block of the Hankel matrix (which is defined by $\mathbf{H}_{x,y} = f(xy)$ for any $x, y \in \Sigma^*$ in the string case). However, we cannot conclude yet that the set of functions recognizable by full rank GWMs is identifiable in the limit. This is because the algorithm we proposed to tackle the constrained Tucker decomposition problem is not guaranteed to return an exact solution. Nonetheless, the learning approach we propose relies on fundamental algebraic properties of GWMs on circular strings that are related to the problem of recovering matrices from a set of trace observables, and constitute a first step towards developing a learning theory for GWMs. The problem of learning GWMs on circular strings that are not of full rank remains open.

Learning methods for WAs rely on the fundamental notion of minimality. A WA with n states is minimal if there does not exist any WA with strictly less than n states computing the same function. This notion of minimality is closely related to the equivalence problem: given two WAs, can we decide if they compute the same function? The intrinsic relations between learning, equivalence and minimization have been investigated in (Maruvsic and Worrell, 2015) for weighted tree automata. A similar notion can be

defined for GWMs but the theoretical study of minimality and of the equivalence problem for GWMs remains to be done. We think that such a study may reveal important properties that could be leveraged to develop learning algorithms for GWMs.

2.6 Conclusion

We proposed a computational model on graphs (GWM) that generalizes the notion of linear representations of recognizable series on strings and trees. The definition of this model was motivated by interpreting the computations of weighted automata on strings and trees in terms of tensor networks. We then studied some fundamental properties of GWMs: closure properties and recognizability of finite support series. We showed that some of these properties are not satisfied by GWMs defined over arbitrary families of graphs but we identified smaller families of graphs (encompassing strings and trees) on which these properties are satisfied. In particular, we showed that families of rooted in-out graphs satisfy all the desirable properties that we considered (except for our conjecture on the recognizability of real-valued series by GWMs with real coefficients). As an illustration of our general research program of learning functions defined over families of graphs, we proposed a learning algorithm for full rank GWMs on circular strings by reducing the learning problem to the constrained Tucker decomposition of a third-order Hankel tensor. The question of learning GWMs on circular strings that are not of full rank remain open.

There are several lines of work that we would like to investigate in the future. A study of GWMs from a formal language perspective would be of great interest. This includes extending the results we presented to the setting where $\mathbb{F}$ is an arbitrary commutative semi-ring, and investigating the connections between GWMs and the models proposed in (Droste and Dück, 2015) and (Makowsky and Kotek, 2014) (which may entail characterizations of GWM-recognizable series in terms of second-order logic). A thorough study of the notion of minimality for GWMs remains to be done, and could shed an interesting light on the problem of learning GWMs. From a machine learning perspective, we are currently investigating the question of learning GWM-recognizable functions defined over 2-dimensional words with potential applications in image processing tasks. We also plan to tackle algorithmic issues and to study how techniques and methods developed in the field of graphical models (such as message passing, variational methods, etc.) and quantum physics (e.g. density matrix renormalization group algorithms), could be adapted to the setting of GWMs.

Appendix

2.A Proof of Theorem 4

Theorem. *Let $\mathcal{S}$ be a family of closed in-out graphs over a ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$, and let $\hat{G} = (\hat{V}, \hat{E}, \hat{l})$ be a graph in $\mathcal{S}$. For any $G \in \mathcal{S}$, let Ψ_G be the set of covering maps from G to $\hat{G}$.*

Then, for any set $\{\epsilon_{\hat{v}} : \hat{v} \in \hat{V}\} \subset \mathbb{R} \setminus \{0\}$ of non-zero real numbers, there exists a GWM M with $|\hat{E}|$ states and with coefficients in $\mathbb{R}$ *such that*

$$f_M(G) = \sum_{\psi \in \Psi_G} \prod_{v \in V} \epsilon_{\psi(v)} \quad \text{for all graphs } G = (V, E, \ell) \in \mathcal{S}.$$

Proof. For any symbol $x \in \Sigma$, we denote by $\hat{V}(x)$ the set of vertices in $\hat{V}$ labeled by x. Let $n = |\hat{E}|$. Instead of indexing the canonical basis of $\mathbb{R}^n$ with integers in $[n]$, we will index it with elements of $\hat{E}$. Let $\eta : P(\hat{G}) \to \hat{E}$ be the mapping associating each port $p \in P(\hat{G})$ with the edge in $\hat{E}$ it belongs to, that is $\eta(p)$ is the unique $e \in \hat{E}$ such that $p \in e$.

Let $M = \langle \mathbb{R}^n, \{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma} \rangle$ be the GWM defined by

$$\boldsymbol{\mathcal{M}}^x = \sum_{\hat{v} \in \hat{V}(x)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})})$$

for all $x \in \Sigma$. For any graph $G = (V, E, \ell)$, we have $f_M(G) = \sum_{\gamma \in \Gamma_{id}(G,M)} \boldsymbol{\mathcal{M}}_\gamma$ where $\boldsymbol{\mathcal{M}}_\gamma = \prod_{v \in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$.

Let $\Gamma_G = \{\gamma : \Gamma_{id}(G, M) \mid \boldsymbol{\mathcal{M}}_\gamma \neq 0\}$ be the set of γ's corresponding to non-zero summands in $f_M(G)$. We first show that for any $\gamma \in \Gamma_G$, there exists a *unique* $\hat{v} \in \hat{V}$ such that $\ell(v) = \hat{\ell}(\hat{v})$ and $\gamma(v, i) = \eta(\hat{v}, i)$ for all $i \in [\sharp v]$. The existence of $\hat{v}$ is a direct consequence of the definition of the tensors $\{\boldsymbol{\mathcal{M}}^x\}_{x \in \Sigma}$: $\boldsymbol{\mathcal{M}}_\gamma \neq 0$ implies that for all $v \in V$ at least one of the summands in

$$\sum_{\hat{v} \in \hat{V}(\ell(v))} \epsilon_{\hat{v}} \cdot \left(\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})}\right)_{\gamma(v,1),\cdots,\gamma(v,\sharp v)}$$

is non-zero, thus there must exists some $\hat{v} \in \hat{V}(\ell(v))$ such that $\gamma(v, i) = \eta(\hat{v}, i)$ for all $i \in [\sharp v]$. Furthermore, $\hat{v}$ is unique because $\hat{G}$ *is an in-out graph.* Indeed, suppose that there exist two distinct vertices $\hat{v}_1, \hat{v}_2 \in \hat{V}$ such that $\ell(v) = \ell(\hat{v}_1) = \ell(\hat{v}_2)$ and $\gamma(v, i) = \eta(\hat{v}_1, i) = \eta(\hat{v}_2, i)$ for all $i \in [\sharp v]$, by defintition of η this would imply that $\{(\hat{v}_1, i), (\hat{v}_2, i)\} \in \hat{E}$, a contradiction with the fact that $\hat{G}$ is an in-out graph.

Now for any $\gamma \in \Gamma_G$ let $\psi_\gamma : v \mapsto \hat{v}$ be the mapping defined by this relation, i.e. $\psi_\gamma(v)$ is the unique $\hat{v} \in \hat{V}$ such that $\ell(v) = \hat{\ell}(\hat{v})$ and $\gamma(v, i) = \eta(\hat{v}, i)$ for all $i \in [\sharp v]$. For any

$\gamma \in \Gamma_G$ and any $v \in V$ we have

$$\begin{aligned}
\boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} &= \left(\sum_{\hat{v}\in\hat{V} \,:\, \ell(\hat{v})=\ell(v)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})}) \right)_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \\
&= \sum_{\hat{v}\in\hat{V} \,:\, \ell(\hat{v})=\ell(v)} \epsilon_{\hat{v}} \cdot (\mathbf{e}_{\eta(\hat{v},1)} \circ \cdots \circ \mathbf{e}_{\eta(\hat{v},\sharp\hat{v})})_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} \\
&= \epsilon_{\psi_\gamma(v)} \ .
\end{aligned}$$

Hence,

$$f_M(G) = \sum_{\gamma\in\Gamma_{id}(G,M)} \boldsymbol{\mathcal{M}}_\gamma = \sum_{\gamma\in\Gamma_G} \boldsymbol{\mathcal{M}}_\gamma = \sum_{\gamma\in\Gamma_G} \prod_{v\in V} \boldsymbol{\mathcal{M}}^{\ell(v)}_{\gamma(v,1),\cdots,\gamma(v,\sharp v)} = \sum_{\gamma\in\Gamma_G} \prod_{v\in V} \epsilon_{\psi_\gamma(v)} \ .$$

We now show that Γ_G and Ψ_G are in one-to-one correspondence through the mapping $\phi : \gamma \mapsto \psi_\gamma$, which will entail the result of the theorem. We start by showing that $\psi_\gamma \in \Psi_G$ for all $\gamma \in \Gamma_G$. Let $\gamma \in \Gamma_G$, we need to show that (i) $\ell(v) = \ell(\psi_\gamma(v))$ for all $v \in V$ and (ii) $\{(\psi_\gamma(v), i), (\psi_\gamma(v'), i')\} \in \hat{E}$ for all $\{(v, i), (v', i')\} \in E$. Condition (i) is directly satisfied by definition of ψ_γ. For condition (ii), let $\{(v, i), (v', i')\} \in E$. On the one hand we have $\gamma(v, i) = \gamma(v', i')$ (because $\gamma \in \Gamma_{id}(G, M)$ and $\{(v, i), (v', i')\} \in E$) and on the other hand we have $\gamma(v, i) = \eta(\psi_\gamma(v), i)$ and $\gamma(v', i') = \eta(\psi_\gamma(v'), i')$ by definition of ψ_γ. It follows that $\eta(\psi_\gamma(v), i) = \eta(\psi_\gamma(v'), i)$, which will imply (by definition of η) that $\{(\psi_\gamma(v), i), (\psi_\gamma(v'), i')\} \in \hat{E}$ if $(\psi_\gamma(v), i) \neq (\psi_\gamma(v'), i')$. Indeed, suppose that $\psi_\gamma(v) = \psi_\gamma(v')$ and $i = i'$, then $\ell(v) = \ell(v')$ and since $\{(v, i), (v', i)\} \in E$ we obtain a contradiction with the fact that *G is an in-out graph*.

To conclude, we need to show that $\phi : \gamma \mapsto \psi_\gamma$ is bijective. Let $\gamma, \gamma' \in \Gamma_G$ be such that $\gamma \neq \gamma'$, then there exists a port $(v, i) \in P(G)$ such that $\gamma(v, i) \neq \gamma'(v, i)$, hence $\eta(\psi_\gamma(v), i) \neq \eta(\psi_{\gamma'}(v), i)$ which implies $\psi_\gamma(v) \neq \psi_{\gamma'}(v)$, thus showing that ϕ is injective. Now let $\psi \in \Psi_G$ and let $\gamma : P(G) \to P(\hat{G})$ be defined by $\gamma(v, i) = \eta(\psi(v, i))$ for all $(v, i) \in P(G)$. It is easy to check that $\gamma \in \Gamma_G$ and $\psi = \psi_\gamma$, showing that ϕ is surjective. □

3 Low-Rank Approximation of Weighted Tree Automata

Contents

3.1 Introduction

In this chapter, we will stay in the world of weighted automata: we will focus on one of the classical models that graph weighted models (introduced in the previous chapter) generalize and consider a problem of *model reduction* for weighted tree automata. Model reduction is a common problem in machine learning, where one wants to reduce the size of some model in order to reduce the computational cost of learning or making new predictions, or simply to reduce the memory required to store the model. Such an example of model reduction for neural networks was mentioned in the introduction: in e.g. (Novikov et al., 2015) the authors use tensor decomposition techniques to compress and speedup the computations of a neural network.

The class of weighted tree automata encompasses probabilistic/weighted context free grammars and is thus relevant to the field of *natural language processing*. Probabilistic context-free grammars (PCFG) provide a powerful statistical formalism for modeling important phenomena occurring in natural language. In fact, learning and parsing algorithms for PCFG are now standard tools in natural language processing pipelines. Most

of these algorithms can be naturally extended to the superclass of weighted context-free grammars (WCFG), and closely related models like weighted tree automata (WTA) and latent probabilistic context-free grammars (LPCFG). The complexity of these algorithms depends on the size of the grammar/automaton, typically controlled by the number of rules/states. Being able to control this complexity is essential in operations like parsing, which is typically executed every time the model is used to make a prediction.

In this chapter we present an algorithm that given a WTA with n states and a target number of states $\hat{n} < n$, returns a WTA with $\hat{n}$ states that is a good approximation of the original automaton. This can be interpreted as a low-rank approximation method for WTA through the direct connection between number of states of a WTA and the rank of its associated Hankel matrix. This opens the door to reducing the complexity of algorithms working with WTA at the price of incurring a small, controlled amount of error in the output of such algorithms. For example, the complexity for parsing a tree t using a WTA is cubic in the number of states (Maletti and Satta, 2009), thus reducing the number of states can lead to a significant speed-up for inference time with a minimized model.

Our techniques are inspired by recent developments in spectral learning algorithms for different classes of models on sequences (Hsu, Kakade, and T. Zhang, 2008; Bailly, Denis, and Ralaivola, 2009; Boots, Siddiqi, and Gordon, 2011; Balle et al., 2014) and trees (Bailly, Habrard, and Denis, 2010; Cohen et al., 2014), and subsequent investigations into low-rank spectral learning for predictive state representations (Kulesza, Rao, and Singh, 2014; Kulesza, N. Jiang, and Singh, 2015) and approximate minimization of weighted automata (Balle, Panangaden, and Precup, 2015). In spectral learning algorithms, data is used to reconstruct a finite block of a Hankel matrix and an SVD of such a matrix then reveals a low-dimensional space where a linear regression recovers the parameters of the model. In contrast, our approach computes the SVD of the *infinite* Hankel matrix associated with a WTA, and then uses it to obtain a low-rank approximation of the initial WTA. Our main result is an efficient algorithm for computing this singular value decomposition by operating directly on the WTA representation of the Hankel matrix; that is, without the need to explicitly represent this infinite matrix at any point. Section 3.2 presents the main ideas underlying our approach, an efficient algorithmic implementation of these ideas is discussed in Section 3.3, and theoretical approximation guarantees are given in Section 3.4.

The connection between tensors and the problem we consider here comes from the fact that the parameters of a WTA are tensors. The approach we propose extends the one given in (Balle, Panangaden, and Precup, 2015) for weighted string automata to the tree case. In this paper, the authors introduce a canonical form for weighted string automata from which approximate minimization can be done in a principled way. They also propose an efficient algorithm to compute this canonical form that relies on the computations of some Gram matrices associated with the automaton. We introduce a similar canonical form for weighted tree automata, but whereas computing the Gram matrices mentioned above could be done by solving a linear system of equations in the string case, in the tree case these Gram matrices are defined by a polynomial system of equations which does not have an analytic solution in general. This comes from the facts that weighted string automata are parameterized by matrices while higher order tensors appear in WTAs; this shows (once again) that problems can become significantly

more difficult when going from matrices to higher order tensors. Nonetheless, using results from fixed-point theory, we are able to provide an efficient algorithm that can approximate these Gram matrices to an arbitrary precision.

The idea of speeding up parsing with (L)PCFG by approximating the original model with a smaller one was recently studied in (Collins and Cohen, 2012; Cohen, Satta, and Collins, 2013), where a tensor decomposition technique was used in order to obtain the minimized model. We compare that approach to ours in the experiments presented in Section 3.5, where both techniques are used to compute approximations to a grammar learned from a corpus of real linguistic data. It was observed in (Collins and Cohen, 2012; Cohen, Satta, and Collins, 2013) that a side-effect of reducing the size of a grammar learned from data was a slight improvement in parsing performance. The number of parameters in the approximate models is smaller, and as such, generalization improves. We show in our experimental section that our minimization algorithms have the same effect in certain parsing scenarios. In addition, our approach yields models which give lower perplexity on an unseen set of sentences, and provides a better approximation to the original model in terms of ℓ_2 distance. It is important to remark that in contrast with the tensor decompositions in (Collins and Cohen, 2012; Cohen, Satta, and Collins, 2013) which are susceptible to local optima problems, our approach resembles a power-method approach to SVD, which yields efficient globally convergent algorithms. Overall, we observe in our experiments that this renders a more stable minimization method than the one using tensor decompositions.

Summary of the contributions. We propose a principled method for *reducing the number of states of a WTA* to approximate a recognizable tree function by a model with a smaller size. Our method extends the one presented in (Balle, Panangaden, and Precup, 2015) for string automata to the tree case, and relies on an algorithm that *computes the SVD of the infinite Hankel matrix associated with a WTA*. We compare our approach to the one proposed in (Collins and Cohen, 2012; Cohen, Satta, and Collins, 2013) to speed up parsing with PCFGs on a grammar learned on a real world corpus with three different evaluation metrics: perplexity, ℓ_2 distance and parsing.

The works presented in this chapter have been realized in collaboration with Borja Balle (Lecturer at Lancaster University) and Shay B. Cohen (Assistant Professor at Edinburgh University), and they have been partially carried out during my visits at McGill University in December 2014 (where B. Balle was a postdoctoral fellow) and Lancaster University in May 2016, and during B. Balle's visit at Aix-Marseille University in February 2015. They were presented in the international conference AISTATS (poster) in May 2016 and in the french machine learning conference CAp (30mn talk) in July 2016.

3.1.1 Trees and Weighted Tree Automata

We start by recalling the definitions of trees and weighted tree automata and by introducing our notations.

Trees on a ranked alphabet. We now introduce notations for describing trees generated by a weighted tree automaton; see Figure 3.1 for some illustrative examples. Recall

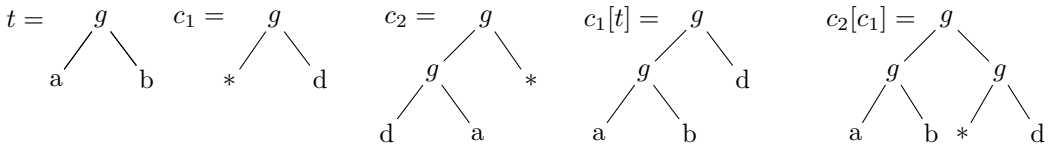


Figure 3.1: Examples of trees $(t, c_1[t] \in \mathfrak{T}_{\mathcal{F}})$ and contexts $(c_1, c_2, c_2[c_1] \in \mathfrak{C}_{\mathcal{F}})$ on the ranked alphabet $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_2$ where $\mathcal{F}_0 = \{a, b, d\}$ and $\mathcal{F}_2 = \{g\}$. With our notations: $c_1[t] = g(g(a, b), d)$, $|c_1[t]| = 5$, $\mathrm{depth}(c_1[t]) = 2$, $\langle t \rangle = ab$, $\mathrm{drop}(c_2[c_1]) = 2$

that given a ranked alphabet $\mathcal{F} = (\Sigma, \sharp)$ we denote by $\mathcal{F}_p = \{g \in \Sigma : \sharp g = p\}$ the set of symbols of arity p, and that the set of trees $\mathfrak{T}_{\mathcal{F}}$ on $\mathcal{F}$ is the smallest set such that

- $\sigma \in \mathfrak{T}_{\mathcal{F}}$ for any $\sigma \in \mathcal{F}_0$,
- $g(t_1, \cdots, t_p) \in \mathfrak{T}_{\mathcal{F}}$ for any $p \geq 1$, $g \in \mathcal{F}_p$ and $t_1, \cdots, t_p \in \mathfrak{T}_{\mathcal{F}}$.

We will call symbols in $\mathcal{F}_0$ *leaf symbols* and symbols in $\mathcal{F}_{\geq 1}$ *internal symbols*. We will sometimes simply write $\mathfrak{T}$ instead of $\mathfrak{T}_{\mathcal{F}}$ when the ranked alphabet is clear from context. The *size* of a tree $t \in \mathfrak{T}_{\mathcal{F}}$ is denoted by $|t|$ and defined recursively by $|\sigma| = 1$ for $\sigma \in \mathcal{F}_0$, and $|g(t_1, \cdots, t_p)| = 1 + |t_1| + \cdots + |t_p|$; that is, the number of nodes in the tree. Given a symbol g and a tree t we will denote by $|t|_g$ the number of nodes in t that are labeled with g. The *depth* of a tree $t \in \mathfrak{T}_{\mathcal{F}}$ is denoted by $\mathrm{depth}(t)$ and defined recursively by $\mathrm{depth}(\sigma) = 0$ for $\sigma \in \mathcal{F}_0$, and $\mathrm{depth}(g(t_1, \cdots, t_p)) = 1 + \max\{\mathrm{depth}(t_1), \cdots, \mathrm{depth}(t_p)\}$; that is, the distance from the root of the tree to the farthest leaf. The *yield* of a tree $t \in \mathfrak{T}_{\mathcal{F}}$ is a string $\langle t \rangle \in \Sigma^*$ defined as the left-to-right concatenation of the symbols in the leafs of t, and can be recursively defined by $\langle \sigma \rangle = \sigma$, and $\langle g(t_1, \cdots, t_p) \rangle = \langle t_1 \rangle \cdots \langle t_p \rangle$.

Let $\mathcal{F}' = (\Sigma \cup \{*\}, \sharp')$, where $*$ is a symbol of arity 0 *not* in Σ. The set of *context trees* is the set $\mathfrak{C}_{\mathcal{F}} = \{c \in \mathfrak{T}_{\mathcal{F}'} : |c|_* = 1\}$; that is, a context $c \in \mathfrak{C}_{\mathcal{F}}$ is a tree in $\mathfrak{T}_{\mathcal{F}'}$ in which the symbol $*$ occurs exactly in one leaf. Note that given a context $c = g(t_1, \cdots, t_p) \in \mathfrak{C}_{\mathcal{F}}$ with $g \in \mathcal{F}_p$, $t_1, \cdots t_p \in \mathfrak{T}_{\mathcal{F}'}$ the symbol $*$ can only appear in one of the t_i's. The *drop* of a context $c \in \mathfrak{C}_{\mathcal{F}}$ is the distance between the root and the leaf labeled with $*$ in c, which can be defined recursively as $\mathrm{drop}(*) = 0$, $\mathrm{drop}(g(t_1, \cdots, t_i, c, t_{i+1}, \cdots, t_p)) = \mathrm{drop}(c)+1$ for any $g \in \mathcal{F}_{p+1}$, $t_1, \cdots, t_p \in \mathfrak{T}_{\mathcal{F}}$.

We usually think of the leaf with the symbol $*$ in a context as a placeholder where the root of another tree or another context can be inserted. Accordingly, given $t \in \mathfrak{T}_{\mathcal{F}}$ and $c \in \mathfrak{C}_{\mathcal{F}}$, we define $c[t] \in \mathfrak{T}_{\mathcal{F}}$ as the tree obtained by replacing the occurrence of $*$ in c with t. Similarly, given $c, c' \in \mathfrak{C}_{\mathcal{F}}$ we can obtain a new context tree $c[c']$ by replacing the occurrence of $*$ in c with c' (see Figure 3.1).

Weighted tree automaton. Recall that a weighted tree automaton (WTA) A is a tuple $(\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$ where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the initial weight vector, $\boldsymbol{\omega}^\sigma \in \mathbb{R}^n$ is the final weight vector associated with the leaf symbol σ for each $\sigma \in \mathcal{F}_0$, and for any symbol g of arity $p \geq 1$, $\boldsymbol{\mathcal{A}}^g \in (\mathbb{R}^n)^{\otimes p+1}$ is the transition tensor of order $p+1$ associated with the internal symbol g. A WTA A *computes* a function $f_A : \mathfrak{T}_{\mathcal{F}} \to \mathbb{R}$ assigning to each tree $t \in \mathfrak{T}_{\mathcal{F}}$ the number computed as $f_A(t) = \boldsymbol{\alpha}^\top \boldsymbol{\omega}_A(t)$, where $\boldsymbol{\omega}_A(t) \in \mathbb{R}^n$ is obtained

recursively as $\boldsymbol{\omega}_A(\sigma) = \boldsymbol{\omega}^\sigma$, and

$$\boldsymbol{\omega}_A(g(t_1, \cdots, t_p)) = \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)).$$

Note that in this chapter we will always use the notation $\boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)) = \boldsymbol{\mathcal{A}}^g \bullet_2 \boldsymbol{\omega}_A(t_1) \bullet_3 \cdots \bullet_{p+1} \boldsymbol{\omega}_A(t_p)$. In many cases we will just write $\boldsymbol{\omega}(t)$ when the automaton A is clear from the context.

An arbitrary function $f : \mathfrak{T}_\mathcal{F} \to \mathbb{R}$ is called *recognizable* (or rational) if there exists a WTA A such that $f = f_A$. The number of states of the smallest such WTA is the *rank* of f — we shall set $\mathrm{rank}(f) = \infty$ if f is not recognizable. A WTA A with n states such that $f_A = f$ and $n = \mathrm{rank}(f)$ is called *minimal*.

Related models. Although WTA are traditionally studied as recognizers for (weighted) regular tree languages, the computation performed by a WTA is closely related to several models typically used in machine learning, including PCFG/WCFGs and recursive tensor neural networks.

The connection with recursive tensor neural networks (RTNN) introduced in (Socher et al., 2013) follows from observing that the computation of a WTA has the same bottom-up computational structure as a RTNN without the non-linearities. The values of the n components of a leaf vector $\boldsymbol{\omega}_\sigma$ correspond to the values of n features representing the symbol $\sigma \in \Sigma$. The computation performed by a WTA processes a tree from the bottom up: whenever a tree of the form $t = g(t_1, \cdots, t_p)$ is encountered, it first processes the subtrees $t_1, \cdots, t_p$ to obtain the feature vectors $\boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)$, and then computes a new feature vector for t as $\boldsymbol{\omega}_A(t) = \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p))$. At the top level the computation ends by producing the scalar $f_A(t) = \boldsymbol{\alpha}_A^\top \boldsymbol{\omega}_A(t)$ corresponding to the inner product of the feature representation of t with the vector $\boldsymbol{\alpha}_A$.

It is well known that the set of derivation trees of a context-free grammar forms a regular tree language, that is a language that can be recognized by a (unweighted) tree automaton. The connection between weighted/probabilistic context free grammars (WCFG/PCFG) and WTAs is of a similar nature. In Section 3.5, we will evaluate our approximate minimization method by reducing the size of a WTA obtained from a PCFG learned on a real world corpus. We conclude this introduction by showing how such a conversion between the two models can be done.

A *weighted context-free grammar* (WCFG) in Chomsky normal form is a tuple $G = (\mathcal{N}, \Sigma, \mathcal{R}, \mathrm{weight})$ where

- $\mathcal{N}$ is the finite set of nonterminal symbols,
- Σ is the finite set of terminal symbols (or letters), with $\Sigma \cap \mathcal{N} = \emptyset$,
- $\mathcal{R}$ is a set of rules having the following forms: initial rules $(\to a)$, internal rules $(a \to bc)$ and final rules $(a \to x)$ for $a, b, c \in \mathcal{N}, x \in \Sigma$,
- $\mathrm{weight} : \mathcal{R} \to \mathbb{R}$ is the weight function which is extended to the set of all possible rules by letting $\mathrm{weight}(\delta) = 0$ for all rules $\delta \notin \mathcal{R}$.

A derivation tree τ of the grammar is a binary tree with internal nodes labeled by nonterminal symbols and leafs labeled by terminal symbols. We say that a rule $(\to a)$ appears in τ if the root of τ is labeled by a, that $(a \to bc)$ appears in τ if there is an

internal node labeled by a whose left and right sons are labeled by b and c respectively, and that $(a \to x)$ appears in τ if there is a leaf labeled by x whose father is labeled by a.

A WCFG G assigns a weight to each derivation tree τ of the grammar given by

$$\text{weight}(\tau) = \prod_{\delta \in \mathcal{R}} (\text{weight}(\delta))^{\sharp_\delta(\tau)}$$

where $\sharp_\delta(\tau)$ is the number of times the rule δ appears in τ, and it computes a function $f_G : \Sigma^+ \to \mathbb{R}$ defined by

$$f_G(w) = \sum_{\tau \in T(w)} \text{weight}(\tau)$$

for any $w \in \Sigma^+$, where $T(w)$ is the set of trees deriving the word w (i.e. $\langle \tau \rangle = w$ for any $\tau \in T(w)$).

Given a WCFG G, we consider the set of binary trees on the ranked alphabet $\mathcal{F} = (\Sigma \cup \{\diamond\}, \sharp)$ where each symbol $\sigma \in \Sigma$ is of arity 0 and $\diamond$ is a new symbol of arity 2. We can then build a WTA that assigns to each binary tree $t \in \mathfrak{T}_\mathcal{F}$ the sum of the weights of all derivation trees of G having the same topology as t. Let $G = (\mathcal{N}, \Sigma, \mathcal{R}, \text{weight})$ be a WCFG in normal form with $\mathcal{N} = [n]$. Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$ be the WTA with n states defined by

- $\boldsymbol{\alpha}_i = \text{weight}(\to i)$ for all $i \in [n]$,
- $\boldsymbol{\mathcal{A}}^\diamond_{i,j,k} = \text{weight}(i \to jk)$ for all $i, j, k \in [n]$,
- $\boldsymbol{\omega}^\sigma_i = \text{weight}(i \to \sigma)$ for all $i \in [n]$, $\sigma \in \Sigma$.

Then for all $w \in \Sigma^+$ we have $f_G(w) = \sum_{t \in \mathfrak{T}_\mathcal{F} : \langle t \rangle = w} f_A(t)$. It is important to note that in this conversion the number of states in A corresponds to the number of non-terminals in G. A similar construction can be used to convert any WTA to a WCFG where each state in the WTA is mapped to a non-terminal in the WCFG.

3.2 Approximate Minimization of Weighted Tree Automata

In this section, we present our method to reduce the size of a WTA. We will first show that the set of WTAs computing a function f is in one-to-one correspondence with the set of rank factorizations of the so-called *Hankel matrix* associated with f. This result will allow us to define a canonical form for WTAs where the states of the automaton are associated with the singular values of the Hankel matrix, which suggests a principled way for approximate minimization by removing the states associated with the smaller singular values.

3.2.1 Rank Factorizations of Hankel Matrices

We start by a crucial observation about WTAs: there exist more than one WTA computing the same function — in fact, there exist infinitely many. An important construction along these lines is the *conjugate* of a WTA A with n states by an invertible matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$. If $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$, its conjugate by $\mathbf{Q}$ is

$$A^\mathbf{Q} = (\mathbb{R}^n, \mathbf{Q}^\top \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g(\mathbf{Q}^{-\top}, \mathbf{Q}, \cdots, \mathbf{Q})\}_{g \in \mathcal{F}_{\geq 1}}, \{\mathbf{Q}^{-1}\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0}) \tag{3.1}$$

where $\mathbf{Q}^{-\top} = (\mathbf{Q}^\top)^{-1}$ denotes the inverse of the transpose. To show that $f_A = f_{A\mathbf{Q}}$ one applies an induction argument on $\mathrm{depth}(t)$ to show that $\boldsymbol{\omega}_{A\mathbf{Q}}(t) = \mathbf{Q}^{-1}\boldsymbol{\omega}_A(t)$ for every $t \in \mathfrak{T}_\mathcal{F}$. The claim is obvious for trees of zero depth $\sigma \in \Sigma$, and for $t = g(t_1, \cdots, t_p)$ we have

$$\begin{aligned}
\boldsymbol{\omega}_{A\mathbf{Q}}(g(t_1, \cdots, t_k)) &= (\boldsymbol{\mathcal{A}}^g(\mathbf{Q}^{-\top}, \mathbf{Q}, \cdots, \mathbf{Q}))(\mathbf{I}, \boldsymbol{\omega}_{A\mathbf{Q}}(t_1), \cdots, \boldsymbol{\omega}_{A\mathbf{Q}}(t_p)) \\
&= (\boldsymbol{\mathcal{A}}^g(\mathbf{Q}^{-\top}, \mathbf{Q}, \cdots, \mathbf{Q}))(\mathbf{I}, \mathbf{Q}^{-1}\boldsymbol{\omega}_A(t_1), \cdots, \mathbf{Q}^{-1}\boldsymbol{\omega}_A(t_p)) \\
&= \boldsymbol{\mathcal{A}}^g(\mathbf{Q}^{-\top}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)) \\
&= \mathbf{Q}^{-1}\boldsymbol{\mathcal{A}}(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)) = \mathbf{Q}^{-1}\boldsymbol{\omega}_A(t),
\end{aligned}$$

where we just used some simple rules of tensor algebra.

Given any $f : \mathfrak{T} \to \mathbb{R}$ we define its *Hankel matrix* as the bi-infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\mathfrak{C}\times\mathfrak{T}}$ with rows indexed by contexts, columns indexed by trees, and whose entries are given by $\mathbf{H}_f(c, t) = f(c[t])$. Note that given a tree $t' \in \mathfrak{T}$ there are exactly $|t'|$ different ways of splitting $t' = c[t]$ with $c \in \mathfrak{C}$ and $t \in \mathfrak{T}$. This implies that $\mathbf{H}_f$ is a highly redundant representation for f, and it turns out that this redundancy is the key to proving the following fundamental result about recognizable tree functions.

Theorem 8 (Bozapalidis and Louscou-Bozapalidou, 1983)**.** *For any $f : \mathfrak{T}_\mathcal{F} \to \mathbb{R}$ we have* $\mathrm{rank}(f) = \mathrm{rank}(\mathbf{H}_f)$.

The theorem above can be rephrased as saying that the rank of $\mathbf{H}_f$ is finite if and only if f is recognizable. When the rank of $\mathbf{H}_f$ is indeed finite — say $\mathrm{rank}(\mathbf{H}_f) = n$ — one can find two rank n matrices $\mathbf{P} \in \mathbb{R}^{\mathfrak{C}\times n}$, $\mathbf{S} \in \mathbb{R}^{n\times\mathfrak{T}}$ such that $\mathbf{H}_f = \mathbf{PS}$. In this case we say that $\mathbf{P}$ and $\mathbf{S}$ give a *rank factorization* of $\mathbf{H}_f$. We shall now refine Theorem 8 by showing that when f is recognizable, the set of all possible rank factorizations of $\mathbf{H}_f$ is in direct correspondence with the set of minimal WTA computing f.

The first step is to show that any minimal WTA $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ computing f induces a rank factorization $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$. We build $\mathbf{S}_A \in \mathbb{R}^{n\times\mathfrak{T}}$ by setting the column corresponding to a tree t to $\mathbf{S}_A(:, t) = \boldsymbol{\omega}_A(t)$. In order to define $\mathbf{P}_A$ we need to introduce a new mapping $\boldsymbol{\Xi}_A : \mathfrak{C} \to \mathbb{R}^{n\times n}$ assigning a matrix to every context as follows: $\boldsymbol{\Xi}_A(*) = \mathbf{I}$ and for any context $c = g(t_1, \cdots, t_{i-1}, c', t_{i+1}, \cdots, t_p)$ where $p \geq 1$, $g \in \mathcal{F}_p$, $t_j \in \mathfrak{T}$ for $j \neq i$ and $c' \in \mathfrak{C}$

$$\boldsymbol{\Xi}_A(c) = \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_{i-1}), \boldsymbol{\Xi}_A(c'), \boldsymbol{\omega}_A(t_{i+1}), \cdots, \boldsymbol{\omega}_A(t_p)). \tag{3.2}$$

If we now define $\boldsymbol{\alpha}_A : \mathfrak{C} \to \mathbb{R}^n$ as $\boldsymbol{\alpha}_A(c)^\top = \boldsymbol{\alpha}^\top\boldsymbol{\Xi}_A(c)$, we can set the row of $\mathbf{P}_A$ corresponding to c to be $\mathbf{P}_A(c, :) = \boldsymbol{\alpha}_A(c)^\top$. With these definitions one can easily show by induction on $\mathrm{drop}(c)$ that $\boldsymbol{\Xi}_A(c)\boldsymbol{\omega}_A(t) = \boldsymbol{\omega}_A(c[t])$ for any $c \in \mathfrak{C}$ and $t \in \mathfrak{T}$. Then it is immediate to check that $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$:

$$\begin{aligned}
\sum_{i=1}^{n} \mathbf{P}_A(c, i)\mathbf{S}_A(i, t) &= \boldsymbol{\alpha}_A(c)^\top\boldsymbol{\omega}_A(t) = \boldsymbol{\alpha}^\top\boldsymbol{\Xi}_A(c)\boldsymbol{\omega}_A(t) \\
&= \boldsymbol{\alpha}^\top\boldsymbol{\omega}_A(c[t]) = f_A(c[t]) = \mathbf{H}_f(c, t).
\end{aligned} \tag{3.3}$$

As before, we will sometimes just write $\boldsymbol{\Xi}(c)$ and $\boldsymbol{\alpha}(c)$ when A is clear from the context. We can now state the main result of this section, which generalizes similar results in

(Balle et al., 2014; Balle, Panangaden, and Precup, 2015) for weighted automata on strings.

Theorem 9. *Let $f : \mathfrak{T} \to \mathbb{R}$ be recognizable. If $\mathbf{H}_f = \mathbf{PS}$ is a rank factorization, then there exists a minimal WTA A computing f such that $\mathbf{P}_A = \mathbf{P}$ and $\mathbf{S}_A = \mathbf{S}$.*

Proof. Let $n = \text{rank}(f)$. Let $B = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{B}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$ be an arbitrary minimal WTA computing f. Suppose B induces the rank factorization $\mathbf{H}_f = \mathbf{P}'\mathbf{S}'$. Since the columns of both $\mathbf{P}$ and $\mathbf{P}'$ are basis for the column-span of $\mathbf{H}_f$, there must exists a change of basis $\mathbf{Q} \in \mathbb{R}^{n \times n}$ between $\mathbf{P}$ and $\mathbf{P}'$. That is, $\mathbf{Q}$ is an invertible matrix such that $\mathbf{P}'\mathbf{Q} = \mathbf{P}$. Furthermore, since $\mathbf{P}'\mathbf{S}' = \mathbf{H}_f = \mathbf{PS} = \mathbf{P}'\mathbf{QS}$ and $\mathbf{P}'$ has full column rank, we must have $\mathbf{S}' = \mathbf{QS}$, or equivalently, $\mathbf{Q}^{-1}\mathbf{S}' = \mathbf{S}$. Thus, we let $A = B^{\mathbf{Q}}$, which immediately verifies $f_A = f_B = f$. It remains to show that A induces the rank factorization $\mathbf{H}_f = \mathbf{PS}$. Note that when proving the equivalence $f_A = f_B$ we already showed $\boldsymbol{\omega}_A(t) = \mathbf{Q}^{-1}\boldsymbol{\omega}_B(t)$, thus $\mathbf{S}_A = \mathbf{Q}^{-1}\mathbf{S}' = \mathbf{S}$. To show $\mathbf{P}_A = \mathbf{P}'\mathbf{Q}$ we need to show that for any $c \in \mathfrak{C}$ we have $\boldsymbol{\alpha}_A(c)^\top = \boldsymbol{\alpha}_B(c)^\top\mathbf{Q}$. This will immediately follow if we show that $\boldsymbol{\Xi}_A(c) = \mathbf{Q}^{-1}\boldsymbol{\Xi}_B(c)\mathbf{Q}$. If we proceed by induction on $\text{drop}(c)$, we see that the case $c = *$ is immediate. For $c = g(c', t_1, \cdots, t_p)$ where $c' \in \mathfrak{C}$, $p \geq 0$, $g \in \mathcal{F}_{p+1}$ and $t_1, \cdots, t_p \in \mathfrak{T}$, we get

$$\begin{aligned}
\boldsymbol{\Xi}_A(g(c', t_1, \cdots, t_p)) &= \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Xi}_A(c'), \boldsymbol{\omega}_A(t_1), \cdots, \boldsymbol{\omega}_A(t_p)) \\
&= (\boldsymbol{\mathcal{B}}^g(\mathbf{Q}^{-\top}, \mathbf{Q}, \cdots, \mathbf{Q}))(\mathbf{I}, \mathbf{Q}^{-1}\boldsymbol{\Xi}_B(c')\mathbf{Q}, \mathbf{Q}^{-1}\boldsymbol{\omega}_B(t_1), \cdots, \mathbf{Q}^{-1}\boldsymbol{\omega}_B(t_p)) \\
&= \boldsymbol{\mathcal{B}}^g(\mathbf{Q}^{-\top}, \boldsymbol{\Xi}_B(c')\mathbf{Q}, \boldsymbol{\omega}_B(t_1), \cdots, \boldsymbol{\omega}_B(t_p)) \\
&= \mathbf{Q}^{-1}\boldsymbol{\mathcal{B}}^g(\mathbf{I}, \boldsymbol{\Xi}_B(c'), \boldsymbol{\omega}_B(t_1), \cdots, \boldsymbol{\omega}_B(t_p))\mathbf{Q} = \mathbf{Q}^{-1}\boldsymbol{\Xi}_B(c)\mathbf{Q}.
\end{aligned}$$

Applying the same argument mutatis mutandis for contexts of the form $c = g(t_1, \cdots, t_{l-1}, c', t_l, \cdots, t_p)$ completes the proof. □

3.2.2 Approximate Minimization with the Singular Value Tree Automaton

Equation (3.3) can be interpreted as saying that given a fixed factorization $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$, the value $f_A(c[t])$ is given by the inner product $\langle \boldsymbol{\alpha}_A(c), \boldsymbol{\omega}_A(t) \rangle = \sum_i (\boldsymbol{\alpha}_A(c))_i (\boldsymbol{\omega}_A(t))_i$. Thus, $(\boldsymbol{\alpha}_A(c))_i$ and $(\boldsymbol{\omega}_A(t))_i$ quantify the influence of state i in the computation of $f_A(c[t])$, and by extension one can use $\|\mathbf{P}_A(:, i)\|$ and $\|\mathbf{S}_A(i, :)\|$ to measure the overall influence of state i in f_A. Since our goal is to approximate a given WTA by a smaller WTA obtained by removing some states in the original one, we shall proceed by removing those states with overall less influence on the computation of f. But because there are infinitely many WTAs computing f, we need to first fix a particular representation for f before we can remove the less influential states. In particular, we seek a representation where each state is decoupled as much as possible from each other state, and where there is a clear ranking of states in terms of overall influence. It turns out that this can all be achieved by a canonical form for WTA we call the singular value tree automaton, which provides an implicit representation for the SVD of $\mathbf{H}_f$. We now show conditions for the existence of such a canonical form, and we develop an algorithm to compute it efficiently in the next section.

Suppose $f : \mathfrak{T} \to \mathbb{R}$ is a rank n recognizable function such that its Hankel matrix admits a reduced singular value decomposition $\mathbf{H}_f = \mathbf{UDV}^\top$. Then we have that $\mathbf{P} =$

$\mathbf{U}\mathbf{D}^{1/2}$ and $\mathbf{S} = \mathbf{D}^{1/2}\mathbf{V}^\top$ is a rank decomposition for $\mathbf{H}_f$, and by Theorem 9 there exists some minimal WTA A with $f_A = f$, $\mathbf{P}_A = \mathbf{U}\mathbf{D}^{1/2}$ and $\mathbf{S}_A = \mathbf{D}^{1/2}\mathbf{V}^\top$. We call such an A a *singular value tree automaton* (SVTA) for f. However, these are not defined for every recognizable function f, because the fact that the columns of $\mathbf{U}$ and $\mathbf{V}$ must be unitary vectors (i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$) imposes some restrictions on which infinite Hankel matrices $\mathbf{H}_f$ admit an SVD — this phenomenon is related to the distinction between compact and non-compact operators in functional analysis. Our next theorem gives a sufficient condition for the existence of an SVD of $\mathbf{H}_f$.

We say that a function $f : \mathfrak{T}_\mathcal{F} \to \mathbb{R}$ is *strongly convergent* if the series $\sum_{t\in\mathfrak{T}_\mathcal{F}} |t||f(t)|$ converges. To see the intuitive meaning of this condition, assume that f is a probability distribution over trees in $\mathfrak{T}$. In this case, strong convergence is equivalent to saying that the expected size of trees generated from the distribution f is finite. It turns out that strong convergence of f is a sufficient condition to guarantee the existence of an SVD for $\mathbf{H}_f$.

Theorem 10. *If $f : \mathfrak{T}_\Sigma \to \mathbb{R}$ is recognizable and strongly convergent, then $\mathbf{H}_f$ admits a singular value decomposition.*

Proof. The result will follow if we show that $\mathbf{H}_f$ is the matrix of a compact operator between Hilbert spaces (Hsing and Eubank, 2015, Theorem 4.3.5). We start by defining the Hilbert spaces of square-summable series indexed by trees and contexts. Given two functions $g, g' : \mathfrak{T}_\Sigma \to \mathbb{R}$ we define their inner product to be $\langle g, g'\rangle_\mathfrak{T} = \sum_{t\in\mathfrak{T}_\Sigma} g(t)g'(t)$ (whenever the sum converges). Let $\|g\|_\mathfrak{T} = \sqrt{\langle g, g\rangle_\mathfrak{T}}$ be the induced norm. We denote by $\ell^2_\mathfrak{T}$ be the real vector space of functions $\{g : \mathfrak{T} \to \mathbb{R} \mid \|g\|_\mathfrak{T} < \infty\}$, which is a separable Hilbert space because the set $\mathfrak{T}$ is countable. Similarly, given functions $g, g' : \mathfrak{C}_\Sigma \to \mathbb{R}$ we define an inner product $\langle g, g'\rangle_\mathfrak{C} = \sum_{c\in\mathfrak{C}_\Sigma} g(t)g'(t)$, a norm $\|g\|_\mathfrak{C} = \sqrt{\langle g, g\rangle_\mathfrak{C}}$, and a separable Hilbert space $\ell^2_\mathfrak{C} = \{g : \mathfrak{C} \to \mathbb{R} | \|g\|_\mathfrak{C} < \infty\}$. With this notation it is possible to see that $\mathbf{H}_f$ is the matrix under the standard basis on $\ell^2_\mathfrak{T}$ and $\ell^2_\mathfrak{C}$ of the operator $H_f : \ell^2_\mathfrak{T} \to \ell^2_\mathfrak{C}$ given by $(H_f g)(c) = \sum_{t\in\mathfrak{T}_\Sigma} f(c[t])g(t)$. Since f is recognizable, $\mathbf{H}_f$ is a finite-rank matrix and therefore H_f is a finite-rank operator. Thus, to show the compactness of H_f it only remains to show that H_f is bounded.

Given $f \in \ell^2_\mathfrak{T}$ and $c \in \mathfrak{C}_\Sigma$ we define a new function $f_c \in \ell^2_\mathfrak{T}$ given by $f_c(t) = f(c[t])$ for $t \in \mathfrak{T}_\Sigma$. Now let $g \in \ell^2_\mathfrak{T}$ with $\|g\|_\mathfrak{T} = 1$ and recall that H_f is bounded if $\|H_f g\|_\mathfrak{C} < \infty$ for every $g \in \ell^2_\mathfrak{T}$ with $\|g\|_\mathfrak{T} = 1$. To show that H_f is bounded observe that

$$
\begin{aligned}
\|H_f g\|^2_\mathfrak{C} &= \sum_{c\in\mathfrak{C}_\Sigma} (H_f g)(c)^2 = \sum_{c\in\mathfrak{C}_\Sigma} \left(\sum_{t\in\mathfrak{T}_\Sigma} f(c[t])g(t) \right)^2 \\
&= \sum_{c\in\mathfrak{C}_\Sigma} \langle f_c, g\rangle^2_\mathfrak{T} \leq \|g\|^2_\mathfrak{T} \sum_{c\in\mathfrak{C}_\Sigma} \|f_c\|^2_\mathfrak{T} \\
&= \sum_{c\in\mathfrak{C}_\Sigma} \sum_{t\in\mathfrak{T}_\Sigma} f_c(t)^2 = \sum_{c\in\mathfrak{C}_\Sigma} \sum_{t\in\mathfrak{T}_\Sigma} f(c[t])^2 \\
&= \sum_{t\in\mathfrak{T}_\Sigma} |t| f(t)^2 \leq \sup_{t\in\mathfrak{T}_\Sigma} |f(t)| \cdot \sum_{t\in\mathfrak{T}_\Sigma} |t||f(t)| \\
&< \infty,
\end{aligned}
$$

where we used the Cauchy–Schwarz inequality, and the fact that $\sup_{t\in\mathfrak{T}_\Sigma} |f(t)|$ is bounded

when f is strongly convergent. $\square$

Together, Theorems 9 and 10 imply that every recognizable strongly convergent $f : \mathfrak{T} \to \mathbb{R}$ can be represented by an SVTA A. If $\mathrm{rank}(f) = n$, then A has n states and for every $i \in [n]$ the ith state contributes to $\mathbf{H}_f$ by generating the ith left and right singular vectors weighted by $\sqrt{\mathfrak{s}_i}$, where $\mathfrak{s}_i = \mathbf{D}_{i,i}$ is the ith singular value. Thus, if we want to obtain a good approximation $\hat{f}$ to f with $\hat{n}$ states, we can take the WTA $\hat{A}$ obtained by removing the last $n - \hat{n}$ states from A, which corresponds to removing from f the contribution of the smallest singular values of $\mathbf{H}_f$. We call such $\hat{A}$ an *SVTA truncation*. Given an SVTA $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$ and $\boldsymbol{\Pi} = \begin{bmatrix} \mathbf{I}_{\hat{n}} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{\hat{n} \times n}$, the SVTA truncation to $\hat{n}$ states can be written as

$$\hat{A} = (\mathbb{R}^{\hat{n}}, \boldsymbol{\Pi}\boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g(\boldsymbol{\Pi}^\top, \cdots, \boldsymbol{\Pi}^\top)\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\Pi}\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0}).$$

Theoretical guarantees on the error induced by the SVTA truncation method are presented in Section 3.4.

3.3 Computing the Singular Value WTA

The previous section shows that in order to compute an approximation of a strongly convergent recognizable function $f : \mathfrak{T} \to \mathbb{R}$ one can proceed by truncating its SVTA. However, the only obvious way to obtain such an SVTA is by computing the SVD of the infinite matrix $\mathbf{H}_f$. In this section, we show that if we are given an arbitrary minimal WTA A for f, then we can transform A into the corresponding SVTA efficiently[a]. In other words, given a representation of $\mathbf{H}_f$ as a WTA, we can compute its SVD *without* the need to operate on infinite matrices. The key observation is to reduce the computation of the SVD of $\mathbf{H}_f$ to the computation of spectral properties of the Gram matrices $\mathbf{G}_{\mathfrak{C}} = \mathbf{P}^\top\mathbf{P}$ and $\mathbf{G}_{\mathfrak{T}} = \mathbf{S}\mathbf{S}^\top$ associated with the rank factorization $\mathbf{H}_f = \mathbf{P}\mathbf{S}$ induced by some minimal WTA computing f. Observe that the Gram matrices associated with the rank factorization $\mathbf{H}_f = \mathbf{P}_A\mathbf{S}_A$ induced by a WTA A satisfy

$$\mathbf{G}_{\mathfrak{C}} = \mathbf{P}_A^\top\mathbf{P}_A = \sum_{c \in \mathfrak{C}} \boldsymbol{\alpha}_A(c)\boldsymbol{\alpha}_A(c)^\top \quad \text{and} \quad \mathbf{G}_{\mathfrak{T}} = \mathbf{S}_A\mathbf{S}_A^\top = \sum_{t \in \mathfrak{T}} \boldsymbol{\omega}_A(t)\boldsymbol{\omega}_A(t)^\top.$$

In the case of weighted automata on strings, (Balle, Panangaden, and Precup, 2015) recently showed a polynomial time algorithm for computing the Gram matrices of a *string* Hankel matrix by solving a system of linear equations. Unfortunately, extending their approach to the tree case requires obtaining a closed-form solution to a system of polynomial equations, which in general does not exist. Thus, we resort to a different algorithmic technique and show that $\mathbf{G}_{\mathfrak{C}}$ and $\mathbf{G}_{\mathfrak{T}}$ can be obtained as fixed points of a certain non-linear operator. This yields the iterative algorithm presented in Algorithm 3 which converges exponentially fast as shown in Theorem 13. The overall procedure to transform a WTA into the corresponding SVTA is presented in Algorithm 2.

[a] If the WTA given to the algorithm is not minimal, a pre-processing step can be used to minimize the input using the algorithm from e.g. Kiefer, Marusic, and Worrell (2015).

In the following proposition, we show how one can construct an SVTA computing a function f (for which $\mathbf{H}_f$ admits an SVD) using the Gram matrices associated with the rank factorization $\mathbf{H}_f = \mathbf{PS}$ induced by any minimal WTA computing f.

Proposition 12. *Let $f : \mathfrak{T} \to \mathbb{R}$ be a recognizable function such that its Hankel matrix $\mathbf{H}_f$ admits an SVD. Let A be a minimal WTA with n states computing f and inducing the rank factorization $\mathbf{H}_f = \mathbf{PS}$, and let $\mathbf{G}_{\mathfrak{C}} = \mathbf{P}^\top\mathbf{P} \in \mathbb{R}^{n\times n}$ and $\mathbf{G}_{\mathfrak{T}} = \mathbf{S}\mathbf{S}^\top \in \mathbb{R}^{n\times n}$ be the corresponding Gram matrices.*

Let $\mathbf{L}_{\mathfrak{C}}, \mathbf{L}_{\mathfrak{T}} \in \mathbb{R}^{n\times n}$ be such that $\mathbf{G}_{\mathfrak{C}} = \mathbf{L}_{\mathfrak{C}}^\top\mathbf{L}_{\mathfrak{C}}$ and $\mathbf{G}_{\mathfrak{T}} = \mathbf{L}_{\mathfrak{T}}^\top\mathbf{L}_{\mathfrak{T}}$ and let $\mathbf{L}_{\mathfrak{C}}\mathbf{L}_{\mathfrak{T}}^\top = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ be a singular value decomposition.

Then, the conjugate WTA $A^{\mathbf{Q}}$ where $\mathbf{Q} = \mathbf{L}_{\mathfrak{C}}^{-1}\mathbf{U}\mathbf{D}^{1/2}$ is an SVTA computing f.

Proof. First observe that since the Gram matrices are symmetric positive semi-definite there exist matrices $\mathbf{L}_{\mathfrak{C}}, \mathbf{L}_{\mathfrak{T}} \in \mathbb{R}^{n\times n}$ such that $\mathbf{G}_{\mathfrak{C}} = \mathbf{L}_{\mathfrak{C}}^\top\mathbf{L}_{\mathfrak{C}}$ and $\mathbf{G}_{\mathfrak{T}} = \mathbf{L}_{\mathfrak{T}}^\top\mathbf{L}_{\mathfrak{T}}$ (one can use Cholesy factorization or eigendecomposition to obtain such matrices). Moreover, since A is a minimal WTA the Gram matrices are of full rank and $\mathbf{L}_{\mathfrak{C}}$ and $\mathbf{L}_{\mathfrak{T}}$ are non-singular.

The conjugate WTA $A^{\mathbf{Q}}$ induces the factorization $\mathbf{H}_f = \mathbf{P}'\mathbf{S}'$ with $\mathbf{P}' = \mathbf{PQ}$ and $\mathbf{S}' = \mathbf{Q}^{-1}\mathbf{S}$. It is easy to check that $\mathbf{Q}^{-1} = \mathbf{D}^{1/2}\mathbf{V}^\top\mathbf{L}_{\mathfrak{T}}^{-\top}$, thus $\mathbf{P}' = \tilde{\mathbf{U}}\mathbf{D}^{1/2}$ and $\mathbf{S}' = \mathbf{D}^{1/2}\tilde{\mathbf{V}}^\top$ with $\tilde{\mathbf{U}} = \mathbf{P}\mathbf{L}_{\mathfrak{C}}^{-1}\mathbf{U}$ and $\tilde{\mathbf{V}}^\top = \mathbf{V}^\top\mathbf{L}_{\mathfrak{T}}^{-\top}\mathbf{S}$. Hence, to show that $A^{\mathbf{Q}}$ is an SVTA it suffices to show that $\mathbf{H}_f = \mathbf{P}'\mathbf{S}' = \tilde{\mathbf{U}}\mathbf{D}\tilde{\mathbf{V}}^\top$ is an SVD which boils down to checking that $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are column-wise orthogonal matrices. Indeed, we have

$$\tilde{\mathbf{U}}^\top\tilde{\mathbf{U}} = \mathbf{U}^\top\mathbf{L}_{\mathfrak{C}}^{-\top}\mathbf{P}^\top\mathbf{P}\mathbf{L}_{\mathfrak{C}}^{-1}\mathbf{U} = \mathbf{U}^\top\mathbf{L}_{\mathfrak{C}}^{-\top}\mathbf{G}_{\mathfrak{C}}\mathbf{L}_{\mathfrak{C}}^{-1}\mathbf{U} = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

and

$$\tilde{\mathbf{V}}^\top\tilde{\mathbf{V}} = \mathbf{V}^\top\mathbf{L}_{\mathfrak{T}}^{-\top}\mathbf{S}\mathbf{S}^\top\mathbf{L}_{\mathfrak{T}}^{-1}\mathbf{V} = \mathbf{V}^\top\mathbf{L}_{\mathfrak{T}}^{-\top}\mathbf{G}_{\mathfrak{T}}\mathbf{L}_{\mathfrak{T}}^{-1}\mathbf{V} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$$

where we used the fact that the matrices $\mathbf{U}$ and $\mathbf{V}$ are orthogonal. □

Algorithm 2 summarizes the overall procedure to construct the SVTA corresponding to a minimal WTA computing a strongly convergent function. Note that the algorithm depends on a procedure for computing the Gram matrices $\mathbf{G}_{\mathfrak{T}}$ and $\mathbf{G}_{\mathfrak{C}}$. In the remaining of this section we present one of our main results: a linearly convergent iterative algorithm for computing these matrices.

Algorithm 2 `ComputeSVTA`

Input: A strongly convergent minimal WTA A
Output: The corresponding SVTA

1: $\mathbf{G}_{\mathfrak{C}}, \mathbf{G}_{\mathfrak{T}} \leftarrow$ `GramMatrices`(A)
2: Compute $\mathbf{L}_{\mathfrak{C}}, \mathbf{L}_{\mathfrak{T}} \in \mathbb{R}^{n\times n}$ such that $\mathbf{G}_{\mathfrak{C}} = \mathbf{L}_{\mathfrak{C}}^\top\mathbf{L}_{\mathfrak{C}}$ and $\mathbf{G}_{\mathfrak{T}} = \mathbf{L}_{\mathfrak{T}}^\top\mathbf{L}_{\mathfrak{T}}$ (using e.g. Cholesky factorizations or eigendecompositions)
3: Let $\mathbf{L}_{\mathfrak{C}}\mathbf{L}_{\mathfrak{T}}^\top = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ be an SVD
4: **return** $A^{\mathbf{Q}}$ where $\mathbf{Q} = \mathbf{L}_{\mathfrak{C}}^{-1}\mathbf{U}\mathbf{D}^{1/2}$

Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ be a strongly convergent WTA of dimension n computing a function f. The following theorem gives two fundamental fixed point equations that are satisfied by the Gram matrices $\mathbf{G}_{\mathfrak{T}}$ and $\mathbf{G}_{\mathfrak{C}}$.

Theorem 11. *The Gram matrices* $\mathbf{G}_{\mathfrak{T}} = \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top$ *and* $\mathbf{G}_{\mathfrak{C}} = \sum_{c\in\mathfrak{C}} \boldsymbol{\alpha}(c)\boldsymbol{\alpha}(c)^\top$ *satisfy the following fixed point equations:*

$$\mathbf{G}_{\mathfrak{T}} = \sum_{\sigma\in\mathcal{F}_0} \boldsymbol{\omega}^\sigma \boldsymbol{\omega}^{\sigma\top} + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} \mathbf{A}^g_{(1)} (\underbrace{\mathbf{G}_{\mathfrak{T}} \otimes \cdots \otimes \mathbf{G}_{\mathfrak{T}}}_{p \text{ times}}) {\mathbf{A}^g_{(1)}}^\top, \tag{3.4}$$

$$\mathbf{G}_{\mathfrak{C}} = \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{i=1}^{p} \mathbf{A}^g_{(i+1)} (\mathbf{G}_{\mathfrak{C}} \otimes \underbrace{\mathbf{G}_{\mathfrak{T}} \otimes \cdots \otimes \mathbf{G}_{\mathfrak{T}}}_{p-1 \text{ times}}) {\mathbf{A}^g_{(i+1)}}^\top. \tag{3.5}$$

Proof. The complete proof is given in Appendix 3.A. Both fixed point equations are obtained by splitting the sum over all trees/contexts between the ones of depth/drop 1 and the ones of depth/drop greater than 1. For trees, the result follows from observing that any tree t of depth greater than 1 can be written as $t = g(t_1, \cdots, t_p)$ for some $p \geq 1$, $g \in \mathcal{F}_p$ and $t_1, \cdots, t_p \in \mathfrak{T}$; similarly, any context c of drop greater than 1 can be written as $c = c'[g(t_1, \cdots, t_{i-1}, *, t_i, \cdots, t_{p-1})]$ for some $c' \in \mathfrak{C}$, $p \geq 1$, $g \in \mathcal{F}_p$ and $t_1, \cdots, t_{p-1} \in \mathfrak{T}$. □

We now show how the Gram matrix $\mathbf{G}_{\mathfrak{T}}$ can be approximated using a simple iterative scheme relying on the fixed point equation from the previous theorem. Let $A^{\otimes} = (\mathbb{R}^{n^2}, \tilde{\boldsymbol{\alpha}}, \{\tilde{\boldsymbol{\mathcal{A}}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\tilde{\boldsymbol{\omega}}^\sigma\}_{\sigma\in\mathcal{F}_0})$ where $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} \otimes \boldsymbol{\alpha}$, $\tilde{\boldsymbol{\mathcal{A}}}^g = \boldsymbol{\mathcal{A}}^g \otimes \boldsymbol{\mathcal{A}}^g \in \mathbb{R}^{n^2\times\cdots\times n^2}$ for all $p \geq 1$, $g \in \mathcal{F}_p$ and $\tilde{\boldsymbol{\omega}}^\sigma = \boldsymbol{\omega}^\sigma \otimes \boldsymbol{\omega}^\sigma$ for all $\sigma \in \mathcal{F}_0$. It is shown in[b] (Berstel and Reutenauer, 1982) that $A^{\otimes}$ computes the function $f_{A^{\otimes}}(t) = f(t)^2$. Note that $\mathbf{G}_{\mathfrak{T}} = \mathbf{S}\mathbf{S}^\top = \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top$, hence $\mathbf{s} \triangleq \mathrm{vec}(\mathbf{G}_{\mathfrak{T}}) = \sum_{t\in\mathfrak{T}} \tilde{\boldsymbol{\omega}}(t)$ since $\tilde{\boldsymbol{\omega}}(t) = \mathrm{vec}(\boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top)$. Thus, computing the Gram matrix $\mathbf{G}_{\mathfrak{T}}$ boils down to computing the vector $\mathbf{s}$. The following theorem shows that this can be done by repeated applications of a non-linear operator until convergence to a fixed point.

Theorem 12. *Let* $F : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ *be the mapping defined by*

$$F(\mathbf{v}) = \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \mathbf{v}, \cdots, \mathbf{v}) \tag{3.6}$$

and let $\mathbf{s} = \sum_{t\in\mathfrak{T}} \tilde{\boldsymbol{\omega}}(t)$. *Then the following hold:*

(i) $\mathbf{s}$ *is a fixed point of* F*; i.e.* $F(\mathbf{s}) = \mathbf{s}$.

(ii) $\mathbf{0}$ *is in the basin of attraction of* $\mathbf{s}$*; i.e.* $\lim_{k\to\infty} F^k(\mathbf{0}) = \mathbf{s}$.

(iii) The iteration defined by $\mathbf{s}_0 = \mathbf{0}$ *and* $\mathbf{s}_{k+1} = F(\mathbf{s}_k)$ *converges linearly to* $\mathbf{s}$*; i.e. there exists* $0 < \rho < 1$ *such that* $\|\mathbf{s}_k - \mathbf{s}\|_2 \leq \mathcal{O}\left(\rho^k\right)$.

Proof. The complete proof is given in Appendix 3.B. First observe that Eq. (3.6) with $\mathbf{v} = \mathbf{s}$ can be seen as a vectorization of the fixed point equation for $\mathbf{G}_{\mathfrak{T}}$ given in Theorem 11; informally this proves (i). For (ii) it suffices to remark that $F^k(\mathbf{0}) = \sum_{t:\mathrm{depth}(t)<k} \tilde{\boldsymbol{\omega}}(t)$, which can be proved by induction on k. For (iii) we use a classical result on fixed point theory stating that if the spectral radius of the Jacobian of F around the fixed point $\mathbf{s}$ is less than 1, then (iii) holds. Thus the proof boils down to showing that this Jacobian (which is actually equal to the matrix $\mathbf{E}$ in Eq. (3.7) below) is less than 1. □

[b] This also directly follows from Proposition 6 since WTAs are a special case of graph weighted models.

Though we could derive a similar iterative algorithm to compute $\mathbf{G}_{\mathfrak{C}}$, it turns out that knowledge of $\mathbf{s} = \mathrm{vec}(\mathbf{G}_{\mathfrak{T}})$ provides an alternative, more efficient procedure to obtain $\mathbf{G}_{\mathfrak{C}}$. As before, we have $\mathbf{G}_{\mathfrak{C}} = \mathbf{P}^\top \mathbf{P} = \sum_{c \in \mathfrak{C}} \boldsymbol{\alpha}(c)\boldsymbol{\alpha}(c)^\top$ and $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(c) \otimes \boldsymbol{\alpha}(c)$ for all $c \in \mathfrak{C}$, hence $\mathbf{q} \triangleq \mathrm{vec}(\mathbf{G}_{\mathfrak{C}}) = \sum_{c \in \mathfrak{C}} \tilde{\boldsymbol{\alpha}}(c)$. Defining the matrix

$$\mathbf{E} = \sum_{p \geq 1} \sum_{g \in \mathcal{F}_p} \sum_{i=0}^{p-1} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{i \text{ times}}, \mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{p-1-i \text{ times}}) \tag{3.7}$$

which only depends on the tensors $\{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}$ and $\mathbf{s}$, we can use the expression $\tilde{\boldsymbol{\alpha}}(c) = \boldsymbol{\Xi}_{A^\otimes}(c)^\top \tilde{\boldsymbol{\alpha}}$ to get

$$\mathbf{q}^\top = \sum_{c \in \mathfrak{C}} \tilde{\boldsymbol{\alpha}}^\top \boldsymbol{\Xi}_{A^\otimes}(c) = \tilde{\boldsymbol{\alpha}}^\top \sum_{k \geq 0} \mathbf{E}^k = \tilde{\boldsymbol{\alpha}}^\top (\mathbf{I} - \mathbf{E})^{-1},$$

where we used the facts that $\mathbf{E}^k = \sum_{c \in \mathfrak{C} : \mathrm{drop}(c) = k} \boldsymbol{\Xi}_{A^\otimes}(c)$ (which can be shown by induction on k) and that the spectral radius of $\mathbf{E}$ is strictly less than 1 (shown in the proof of Theorem 12).

Algorithm 3 summarizes the overall approximation procedure for the Gram matrices, which can be done to an arbitrary precision. There, $\mathrm{reshape}(\cdot, n \times n)$ is an operation that takes an n^2-dimensional vector and returns the $n \times n$ matrix whose first column contains the first n entries in the vector and so on. Theoretical guarantees on the convergence rate of this algorithm are given in the following theorem.

Theorem 13. *There exists $0 < \rho < 1$ such that after k iterations of line 9 in Algorithm 3, the approximations $\hat{\mathbf{G}}_{\mathfrak{C}}$ and $\hat{\mathbf{G}}_{\mathfrak{T}}$ satisfy $\|\mathbf{G}_{\mathfrak{C}} - \hat{\mathbf{G}}_{\mathfrak{C}}\|_F \leq \mathcal{O}(\rho^k)$ and $\|\mathbf{G}_{\mathfrak{T}} - \hat{\mathbf{G}}_{\mathfrak{T}}\|_F \leq \mathcal{O}(\rho^k)$.*

Proof. The proof is given in Appendix 3.C. □

3.4 Approximation Error of an SVTA Truncation

In this section, we analyze the approximation error induced by the truncation of an SVTA. We recall that given an SVTA $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$, its truncation to $\hat{n}$ states is the automaton

$$\hat{A} = (\mathbb{R}^{\hat{n}}, \boldsymbol{\Pi}\boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g(\boldsymbol{\Pi}^\top, \cdots, \boldsymbol{\Pi}^\top)\}_{g \in \mathcal{F}_{\geq 1}}, \{\boldsymbol{\Pi}\boldsymbol{\omega}^\sigma\}_{\sigma \in \mathcal{F}_0})$$

where $\boldsymbol{\Pi} = \begin{bmatrix} \mathbf{I}_{\hat{n}} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{\hat{n} \times n}$ is the projection matrix that removes the states associated with the $n - \hat{n}$ smallest singular values of the Hankel matrix.

Intuitively, the states associated with the smaller singular values are the ones with the less influence on the Hankel matrix, thus they should also be the states having the less effect on the computation of the SVTA. The following theorem supports this intuition by showing a fundamental relation between the singular values of the Hankel matrix of a recognizable function f and the parameters of the SVTA computing it.

Algorithm 3 `GramMatrices`

Input: A strongly convergent minimal WTA $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$
Output: Gram matrices $\hat{\mathbf{G}}_{\mathfrak{C}} \simeq \sum_{c\in\mathfrak{C}} \boldsymbol{\alpha}_A(c)\boldsymbol{\alpha}_A(c)^\top$ and $\hat{\mathbf{G}}_{\mathfrak{T}} \simeq \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}_A(t)\boldsymbol{\omega}_A(t)^\top$

for $g \in \mathcal{F}_{\geq 1}$ **do**
  $\tilde{\boldsymbol{\mathcal{A}}}^g \leftarrow \boldsymbol{\mathcal{A}}^g \otimes \boldsymbol{\mathcal{A}}^g \in (\mathbb{R}^{n^2})^{\otimes(\sharp g+1)}$
end for
for $\sigma \in \mathcal{F}_0$ **do**
  $\tilde{\boldsymbol{\omega}}^\sigma \leftarrow \boldsymbol{\omega}^\sigma \otimes \boldsymbol{\omega}^\sigma \in \mathbb{R}^{n^2}$
end for
$\mathbf{s} \leftarrow \mathbf{0} \in \mathbb{R}^{n^2}$
repeat
  $\mathbf{s} \leftarrow \sum_{\sigma\in\mathcal{F}_0} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \mathbf{s}, \cdots, \mathbf{s})$
until convergence
$\mathbf{E} \leftarrow \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{i=0}^{p-1} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{i \text{ times}}, \mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{p-1-i \text{ times}})$
$\mathbf{q} \leftarrow (\boldsymbol{\alpha}\otimes\boldsymbol{\alpha})^\top (\mathbf{I} - \mathbf{E})^{-1}$
$\hat{\mathbf{G}}_{\mathfrak{T}} \leftarrow \text{reshape}(\mathbf{s}, n \times n)$
$\hat{\mathbf{G}}_{\mathfrak{C}} \leftarrow \text{reshape}(\mathbf{q}, n \times n)$
return $\hat{\mathbf{G}}_{\mathfrak{C}}, \hat{\mathbf{G}}_{\mathfrak{T}}$

Proposition 13. *Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ be an SVTA with n states computing a function f and let $\mathfrak{s}_1 \geq \mathfrak{s}_2 \geq \cdots \geq \mathfrak{s}_n$ be the singular values of the Hankel matrix $\mathbf{H}_f$.*

Then, for all indices $i, i_1, \cdots, i_{p+1} \in [n]$, the following hold:

- $|\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i}$ *for any* $t \in \mathfrak{T}$,
- $|\boldsymbol{\alpha}(c)_i| \leq \sqrt{\mathfrak{s}_i}$ *for any* $c \in \mathfrak{C}$, *and*
- $|\boldsymbol{\mathcal{A}}^g_{i_1\cdots i_{p+1}}| \leq \min_{k\in[p+1]} \frac{\mathfrak{s}_{i_k}}{\sqrt{\mathfrak{s}_{i_1}\cdots\mathfrak{s}_{i_{p+1}}}}$ *for any* $p \geq 1$, $g \in \mathcal{F}_p$.

Proof. For the first point, let $\mathbf{UDV}^\top$ be the SVD of $\mathbf{H}_f$. Since A is an SVTA we have

$$\boldsymbol{\omega}(t)_i^2 = (\mathbf{P}_{i,t})^2 = (\mathbf{D}^{1/2}\mathbf{V}^\top)^2_{i,t} = \mathfrak{s}_i(\mathbf{V}_{t,i})^2$$

and since the rows of $\mathbf{V}$ are orthonormal we have $(\mathbf{V}_{t,i})^2 \leq 1$. The inequality for contexts is proved similarly by reasoning on the rows of $\mathbf{UD}^{1/2}$.

The third point is a direct consequence of the fixed point equations for $\mathbf{G}_{\mathfrak{T}}$ and $\mathbf{G}_{\mathfrak{C}}$ given in Theorem 11. Indeed, since A is an SVTA we have $(\mathbf{G}_{\mathfrak{T}})_{i,i} = (\mathbf{G}_{\mathfrak{C}})_{i,i} = \mathfrak{s}_i$, it is then easy to check that the fixed point equations imply

$$\mathfrak{s}_i = \sum_{\sigma\in\Sigma} (\boldsymbol{\omega}^\sigma)_i^2 + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{j_1,\cdots,j_p=1}^{n} (\boldsymbol{\mathcal{A}}^g_{i,j_1,\cdots,j_p})^2 \mathfrak{s}_{j_1}\mathfrak{s}_{j_2}\cdots\mathfrak{s}_{j_p}$$

and

$$\mathfrak{s}_i = \boldsymbol{\alpha}_i^2 + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{j_1,\cdots,j_p=1}^{n}((\boldsymbol{\mathcal{A}}^g_{j_1,i,j_2,\cdots,j_p})^2+(\boldsymbol{\mathcal{A}}^g_{j_1,j_2,i,j_3,\cdots,j_p})^2+\cdots+(\boldsymbol{\mathcal{A}}^g_{j_1,\cdots,j_p,i})^2)\mathfrak{s}_{j_1}\cdots\mathfrak{s}_{j_p}$$

for all $i \in [n]$. The result follows from observing that all the summands in the two equations are positive. □

Two important properties of SVTAs follow from this proposition. First, the fact that $|\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i}$ implies that the weights associated with states corresponding to small singular values are small. Second, this proposition gives us some intuition on how the states of an SVTA interact with each other. To see this, let $g \in \mathcal{F}_2$ be a symbol of arity 2 and let $\mathbf{M} = \boldsymbol{\mathcal{A}}^g(\boldsymbol{\alpha}, \mathbf{I}, \mathbf{I})$. Then for a tree $t = g(t_1, t_2) \in \mathfrak{T}$ we have $f_A(t) = \boldsymbol{\omega}(t_1)^\top \mathbf{M}\boldsymbol{\omega}(t_2)$. Using the previous proposition one can show[c] that

$$|\mathbf{M}_{ij}| \leq n\sqrt{\frac{\min\{\mathfrak{s}_i, \mathfrak{s}_j\}}{\max\{\mathfrak{s}_i, \mathfrak{s}_j\}}},$$

which tells us that two states corresponding to singular values far away from each other have very little interaction in the computations of the automata.

Proposition 13 is key to proving the following theorem, which is the main result of this section. It shows how the approximation error induced by the truncation of an SVTA is impacted by the magnitudes of the singular values associated with the removed states.

Theorem 14. *Let $P = \max_{g\in\Sigma} \sharp g$ be the maximum arity of symbols in $\mathcal{F} = (\Sigma, \sharp)$. Let $f : \mathfrak{T}_\mathcal{F} \to \mathbb{R}$ be a function computed by an SVTA with n states and let $\hat{f}$ be the function computed by its truncation to $\hat{n}$ states. Then, for any tree $t \in \mathfrak{T}$ we have $|f(t) - \hat{f}(t)| \leq n^{P|t|}\mathfrak{s}_{\hat{n}+1}$.*

Consequently, for any $\varepsilon > 0$ and any tree $t \in \mathfrak{T}$:

$$\text{if } |t| \leq \frac{\log\varepsilon - \log\mathfrak{s}_{\hat{n}+1}}{P\log n} \quad \text{then} \quad |f(t) - \hat{f}(t)| \leq \varepsilon.$$

Proof. See Appendix 3.D. □

Since $\mathfrak{s}_{\hat{n}+1} > \mathfrak{s}_{\hat{n}+2} > \cdots > \mathfrak{s}_n$, this theorem shows that the smaller the singular values associated with the removed states are, the better will be the approximation. As a direct consequence, the error introduced by the truncation grows with the number of states removed. The dependence on the size of the trees comes from the propagation of the error during the contractions of the tensor $\hat{\boldsymbol{\mathcal{T}}}$ of the truncated SVTA.

The decay of singular values can be very slow in the worst case, but in practice it is not unusual to observe an exponential decay on the tail. For example, this is shown to be the case for the SVTA we compute in Section 3.5. Assuming such an exponential decay of the form $\mathfrak{s}_i = C\theta^i$ for some $0 < \theta < 1$, the bound above on the size of the trees for

[c] Indeed, we have $|\mathbf{M}_{i,j}| = |\,(\boldsymbol{\mathcal{A}}^g(\boldsymbol{\alpha}, \mathbf{I}, \mathbf{I}))_{i,j}\,| \leq \sum_{k=1}^n |\boldsymbol{\mathcal{A}}^g_{k,i,j}|\,|\boldsymbol{\alpha}_k|$, hence it follows from Proposition 13 that $|\mathbf{M}_{i,j}| \leq \sum_{k=1}^n \min\{\sqrt{\mathfrak{s}_i/(\mathfrak{s}_j\mathfrak{s}_k)}, \sqrt{\mathfrak{s}_j/(\mathfrak{s}_i\mathfrak{s}_k)}\}\sqrt{\mathfrak{s}_k} = n\cdot\min\{\sqrt{\mathfrak{s}_i/\mathfrak{s}_j}, \sqrt{\mathfrak{s}_j/\mathfrak{s}_i}\}$.

which $|f(t) - \hat{f}(t)| < \varepsilon$ specializes to

$$\frac{\log(\varepsilon) + (\hat{n}+1)\log(1/\theta) - \log(C)}{P \log n}.$$

It is interesting to observe that the dependence of this bound on the number of total/removed states is $\mathcal{O}(\hat{n}/\log(n))$.

To conclude this section, we mention the bound that was obtained for the string case in (Balle, Panangaden, and Precup, 2015):

$$\|f - \hat{f}\|_2^2 = \sum_{x \in \Sigma^*} (f(x) - \hat{f}(x))^2 \leq C_f \sqrt{\mathfrak{s}_{\hat{n}+1} + \cdots + \mathfrak{s}_n}$$

where C_f is a constant that depends only on f. Actually, the tighter bound $\|f - \hat{f}\|_2^2 \leq \mathfrak{s}_{\hat{n}+1}^2 + \cdots + \mathfrak{s}_n^2$ (that does not depend on any constant) can be obtained. In comparison to the tree case, these bounds do not exhibit any dependency on the size of the strings. Simulation studies suggest that the bound $\|f - \hat{f}\|_2^2 \leq \mathfrak{s}_{\hat{n}+1}^2 + \cdots + \mathfrak{s}_n^2$ also hold in the tree case, but despite considerable efforts we did not manage to prove this bound (yet).

3.5 Experiments

In this section, we assess the performance of our method on a model arising from real-world data, by using a PCFG learned from a text corpus as our initial model.

In our experiments, we used the annotated corpus of German newspaper texts NEGRA (Skut et al., 1997). We use a standard setup, in which the first 18,602 sentences are used as a training set, the next 1,000 sentences as a development set and the last 1,000 sentences as a test set S_{test}. All trees are binarized as described in (Cohen et al., 2013). We extract a binary grammar in Chomsky normal form from the data, and then estimate its probabilities using maximum likelihood. The resulting PCFG has $n = 211$ nonterminals. We compare our method against the ones described in (Cohen, Satta, and Collins, 2013), who use tensor decomposition algorithms (Chi and Kolda, 2012) to decompose the tensors of an underlying PCFG[d]. Loosely speaking, their method consists in finding a low CP rank decomposition of a tensor containing the weights of the rules in the PCFG, and performing the PCFG computations using the resulting compressed representation of this tensor.

We used three evaluation measures: ℓ_2 distance (between the functions of type $\mathfrak{T}_{\mathcal{F}} \to \mathbb{R}$ computed by the original WTA and the one computed by its approximation), perplexity on a test set, and parsing accuracy on a test set (comparing the tree topology of parses using the bracketing F-measure). Because the number of states on a WTA and the CP-rank of the tensor decomposition method are not directly comparable, we plot the results using the number of parameters needed to specify the model on the horizontal axis. This number is equal to $\hat{n}^3$ for a WTA with $\hat{n}$ states, and it is equal to $3Rn$ when the tensor containing the weights of the rules from the PCFG is approximated with a tensor of CP-rank R (note that in both cases these are the number of parameters needed to specify

[d] We use two tensor decomposition algorithms from the tensor Matlab toolbox: `pqnr`, which makes use of projected quasi-Newton and `mu`, which uses a multiplicative update. See http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html.

the tensor occurring in the model).

The ℓ_2 distance between the original function f and its minimization $\hat{f}$, $\|f - \hat{f}\|_2^2 = \sum_{t \in \mathfrak{T}} (f(t) - \hat{f}(t))^2$, can be approximated to an arbitrary precision using the Gram matrices of the corresponding WTA (which follows from observing that $(f - \hat{f})^2$ is recognizable). The perplexity of $\hat{f}$ is defined by $2^{-H_{\text{test}}}$, where $H_{\text{test}} = \sum_{t \in S_{\text{test}}} f(t) \log_2 \hat{f}(t)$ and both f and $\hat{f}$ have been normalized to sum to one over the test set. The results are plotted in Figure 3.2, where an horizontal dotted line represents the performance of the original model. We see that our method outperforms the tensor decomposition methods both in terms of ℓ_2 distance and perplexity. We also remark that our method obtains very smooth curves, which comes from the fact that it does not suffer from local optima problems like the tensor decomposition methods.

For parsing we use minimum Bayes risk decoding, maximizing the sum of the marginals for the nonterminals in the grammar, essentially choosing the best tree topology given a string (Goodman, 1996). The results for various length of sentences are shown in Figure 3.3, where we see that our method does not perform as well as the tensor decomposition methods in terms of parsing accuracy on long sentences. In this figure, we also plotted the results for a slight modification of our method (SVTA*) that is able to achieve competitive performances. The SVTA* method gives more importance to long sentences in the minimization process. This is done by finding a high constant $\gamma > 1$ such that the function $f_\gamma : t \mapsto \gamma^{\text{size}(t)} f(t)$ is still strongly convergent (where $\text{size}(t) = |t| - |\langle t \rangle|$ is the number of internal nodes of t). This function is then approximated by a low-rank WTA computing $\hat{f}_\gamma$, and we let $\hat{f} : t \mapsto \gamma^{-\text{size}(t)} \hat{f}_\gamma(t)$ (which is recognizable). In our experiment, we used $\gamma = 2.4$. While the SVTA* method improved the parsing accuracy, it had no significant repercussion on the ℓ_2 and perplexity measures. We believe that the parsing accuracy of our method could be further improved. Seeking techniques that combines the benefits of SVTA and previous works is a promising direction.

Overall, the results are more promising for language modeling (ie. perplexity) than parsing. This is in line with a recent discovery on learning PCFGs (Scicluna and De La Higuera, 2014) showing that it is hard to perform simultaneously well on parsing and language modeling.

3.6 Conclusion

We described a technique for approximate minimization of WTA, yielding a model smaller than the original one which retains good approximation properties. We introduced a canonical form of WTA in which the states of the automaton are associated with singular values of the corresponding Hankel matrix. This canonical form allowed us to achieve approximate minimization in a principled way by removing states corresponding to small singular values of the Hankel matrix. We also provided theoretical approximation guarantees for this minimization scheme. Our main algorithm relies on a singular value decomposition of the infinite Hankel matrix induced by the WTA, and the main technical difficulty to extend the method proposed in (Balle, Panangaden, and Precup, 2015) to the tree case resided in the computation of the Gram matrices associated with the WTA. Even though these Gram matrices do not have a closed form solution in the tree case, we proposed an efficient algorithm to approximate them to an arbitrary precision. This work has connections with spectral learning techniques for WTA, and exhibits sim-

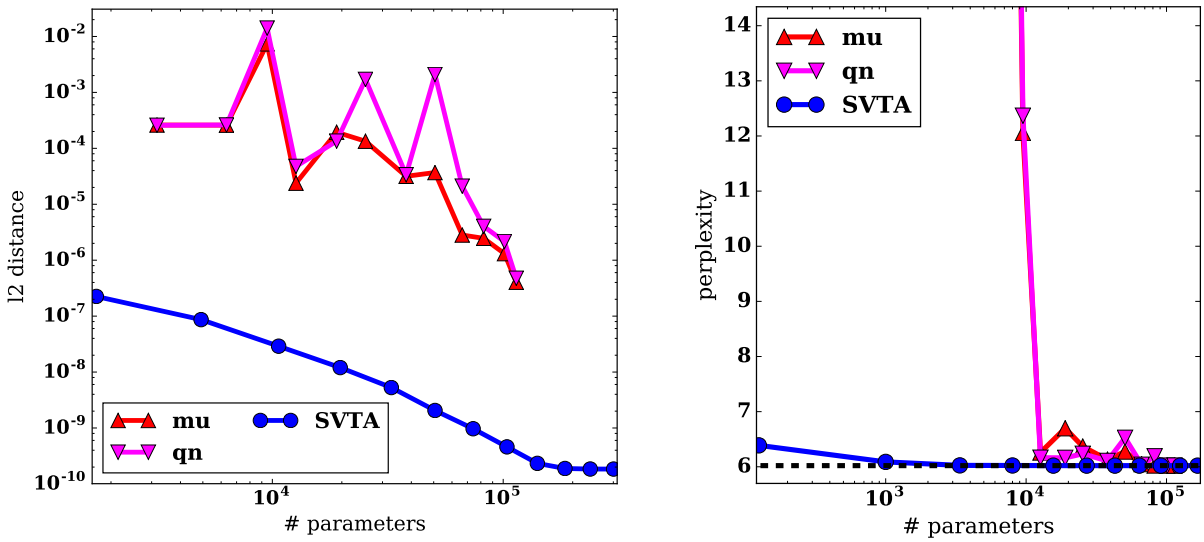


Figure 3.2: Comparison between the SVTA truncation method and the tensor decomposition method proposed in (Cohen, Satta, and Collins, 2013) (`qn` and `mu` correspond to two different optimization algorithms for the tensor decomposition, see footnote d). (left) ℓ_2 distance between functions. (right) Perplexity on the test set. The x axis denotes in both cases the number of parameters used by the approximation. The dotted line corresponds to the perplexity of the original model (e.g. before minimization).

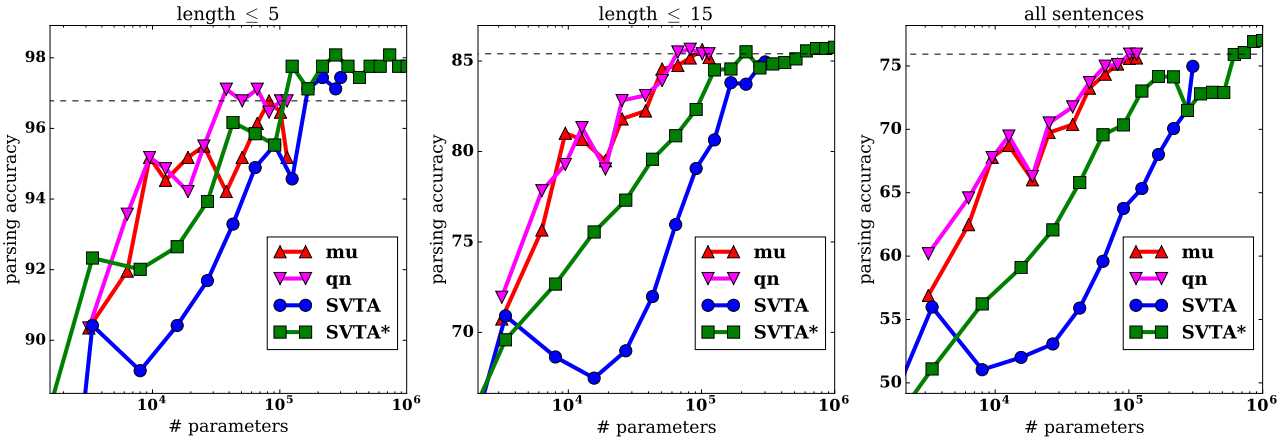


Figure 3.3: Parsing accuracy on the test set for different sentence lengths. The x-axis denote the number of parameters used by the approximation. The y-axis denotes bracketing accuracy. The dotted line corresponds to the accuracy of the original model (e.g. before minimization).

ilar properties as those algorithms; e.g. absence of local optima. Our experiments with real-world parsing data show that the minimized WTA, depending on the number of singular values used, approximates well the original WTA on three measures: perplexity, bracketing accuracy and ℓ_2 distance between tree functions.

In future work we plan to investigate the applications of our approach to the design and analysis of improved spectral learning algorithms for WTA. We will also pursue our efforts to obtain approximation bounds that does not depend on the size of the trees and hopefully to obtain a bound on the ℓ_2 distance between the functions computed by an SVTA and its truncation. On the practical side we want to investigate strategies to sparsify the SVTA obtained after minimization (or to maintain sparsity through the minimization process) in order to further improve the time complexity of WTA algorithms. Finally, we think that investigating the properties of the internal feature representations used by an SVTA might provide useful joint embeddings for the words, non-terminals, and productions rules arising from (L)PCFGs.

Appendix

3.A Proof of Theorem 11

Theorem. *The Gram matrices* $\mathbf{G}_{\mathfrak{T}}$ *and* $\mathbf{G}_{\mathfrak{C}}$ *satisfy the following fixed point equations:*

$$\mathbf{G}_{\mathfrak{T}} = \sum_{\sigma\in\mathcal{F}_0} \boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} \mathbf{A}^g_{(1)}(\underbrace{\mathbf{G}_{\mathfrak{T}}\otimes\cdots\otimes\mathbf{G}_{\mathfrak{T}}}_{p\ times})\mathbf{A}^{g\ \top}_{(1)}, \tag{3.8}$$

$$\mathbf{G}_{\mathfrak{C}} = \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{i=1}^{p} \mathbf{A}^g_{(i+1)}(\mathbf{G}_{\mathfrak{C}}\otimes\underbrace{\mathbf{G}_{\mathfrak{T}}\otimes\cdots\otimes\mathbf{G}_{\mathfrak{T}}}_{p-1\ times})\mathbf{A}^{g\ \top}_{(i+1)}. \tag{3.9}$$

Proof. Using the fact that any tree t of depth greater than 1 can be written as $g(t_1,\cdots,t_p)$ for some $p\geq 1$, $g\in\mathcal{F}_p$ and $t_1,\cdots,t_p\in\mathfrak{T}$ we have

$$\begin{aligned}
\mathbf{G}_{\mathfrak{T}} &= \sum_{t\in\mathfrak{T}}\boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top = \sum_{\sigma\in\mathcal{F}_0}\boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{t\in\mathfrak{T}\,:\,\mathrm{depth}(t)\geq 1}\boldsymbol{\omega}(t)\boldsymbol{\omega}(t)^\top\\
&= \sum_{\sigma\in\mathcal{F}_0}\boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{t_1,\cdots,t_p\in\mathfrak{T}}\boldsymbol{\omega}(g(t_1,\cdots,t_p))\boldsymbol{\omega}(g(t_1,\cdots,t_p))^\top\\
&= \sum_{\sigma\in\mathcal{F}_0}\boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{t_1,\cdots,t_p\in\mathfrak{T}}\boldsymbol{\mathcal{A}}^g(\mathbf{I},\boldsymbol{\omega}(t_1),\cdots,\boldsymbol{\omega}(t_p))\boldsymbol{\mathcal{A}}^g(\mathbf{I},\boldsymbol{\omega}(t_1),\cdots,\boldsymbol{\omega}(t_p))^\top\\
&= \sum_{\sigma\in\mathcal{F}_0}\boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\mathbf{A}^g_{(1)}\sum_{t_1,\cdots,t_p\in\mathfrak{T}}(\boldsymbol{\omega}(t_1)\otimes\cdots\otimes\boldsymbol{\omega}(t_p))(\boldsymbol{\omega}(t_1)\otimes\cdots\otimes\boldsymbol{\omega}(t_p))^\top\mathbf{A}^{g\ \top}_{(1)}\\
&= \sum_{\sigma\in\mathcal{F}_0}\boldsymbol{\omega}_\sigma\boldsymbol{\omega}_\sigma^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\mathbf{A}^g_{(1)}(\underbrace{\mathbf{G}_{\mathfrak{T}}\otimes\cdots\otimes\mathbf{G}_{\mathfrak{T}}}_{p\ times})\mathbf{A}^{g\ \top}_{(1)}.
\end{aligned}$$

To derive a fixed point equation for the Gram matrices for contexts we use the fact that any context $c\in\mathfrak{C}$ of drop greater than 1 can be written as $c'[g(t_1,\cdots,t_{i-1},*,t_i,\cdots,t_{p-1})]$ for some $c'\in\mathfrak{C}$, $p\geq 1$, $g\in\mathcal{F}_p$ and $t_1,\cdots,t_{p-1}\in\mathfrak{T}$. Using the notation $\mathbf{v}^{\circ 2}=\mathbf{v}\mathbf{v}^\top$ for any vector $\mathbf{v}$ we have

$$\begin{aligned}
\mathbf{G}_{\mathfrak{C}} &= \sum_{c\in\mathfrak{C}}\boldsymbol{\alpha}(c)\boldsymbol{\alpha}(c)^\top = \boldsymbol{\alpha}(*)\boldsymbol{\alpha}(*)^\top + \sum_{c\in\mathfrak{C}\,:\,\mathrm{drop}(c)\geq 1}\boldsymbol{\alpha}(c)\boldsymbol{\alpha}(c)^\top\\
&= \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{\substack{p\geq 1,\ g\in\mathcal{F}_p,\ c\in\mathfrak{C},\\ t_1,\cdots,t_{p-1}\in\mathfrak{T}}}\sum_{i=1}^{p}\boldsymbol{\alpha}(c[g(t_1,\cdots,t_{i-1},*,t_i,\cdots,t_{p-1})])^{\circ 2}\\
&= \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{\substack{p\geq 1,\ g\in\mathcal{F}_p,\ c\in\mathfrak{C},\\ t_1,\cdots,t_{p-1}\in\mathfrak{T}}}\sum_{i=1}^{p}\Big(\boldsymbol{\mathcal{A}}^g(\boldsymbol{\alpha}(c),\boldsymbol{\omega}(t_1),\cdots,\boldsymbol{\omega}(t_{i-1}),\mathbf{I},\boldsymbol{\omega}(t_i),\cdots,\boldsymbol{\omega}(t_{p-1}))\Big)^{\circ 2}\\
&= \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{p\geq 1,\ g\in\mathcal{F}_p}\sum_{i=1}^{p}\sum_{\substack{c\in\mathfrak{C},\\ t_1,\cdots,t_{p-1}\in\mathfrak{T}}}\Big(\mathbf{A}^g_{(i+1)}(\boldsymbol{\alpha}(c)\otimes\boldsymbol{\omega}(t_1)\otimes\cdots\otimes\boldsymbol{\omega}(t_{p-1}))\Big)^{\circ 2}\\
&= \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p}\sum_{i=1}^{p}\mathbf{A}^g_{(i+1)}(\mathbf{G}_{\mathfrak{C}}\otimes\underbrace{\mathbf{G}_{\mathfrak{T}}\otimes\cdots\otimes\mathbf{G}_{\mathfrak{T}}}_{p-1\ times})\mathbf{A}^{g\ \top}_{(i+1)}.
\end{aligned}$$

□

3.B Proof of Theorem 12

Theorem. *Let $F : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ be the mapping defined by*

$$F(\mathbf{v}) = \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \mathbf{v}, \cdots, \mathbf{v})$$

and let $\mathbf{s} = \sum_{t\in\mathfrak{T}} \tilde{\boldsymbol{\omega}}(t)$. Then the following hold:

(i) $\mathbf{s}$ *is a fixed point of F; i.e. $F(\mathbf{s}) = \mathbf{s}$.*

(ii) $\mathbf{0}$ *is in the basin of attraction of* $\mathbf{s}$*; i.e. $\lim_{k\to\infty} F^k(\mathbf{0}) = \mathbf{s}$.*

(iii) The iteration defined by $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{s}_{k+1} = F(\mathbf{s}_k)$ converges linearly to $\mathbf{s}$; i.e. there exists $0 < \rho < 1$ such that $\|\mathbf{s}_k - \mathbf{s}\|_2 \leq \mathcal{O}\left(\rho^k\right)$.

Proof. (i) For any $p \geq 1$ and $g \in \mathcal{F}_p$, we have

$$\tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \mathbf{s}, \cdots, \mathbf{s}) = \sum_{t_1,\cdots,t_p\in\mathfrak{T}} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \tilde{\boldsymbol{\omega}}(t_1), \cdots, \tilde{\boldsymbol{\omega}}(t_p)) = \sum_{t_1,\cdots,t_p\in\mathfrak{T}} \tilde{\boldsymbol{\omega}}(g(t_1, \cdots, t_p)) = \sum_{t\in\mathfrak{T}^{\geq 1}} \tilde{\boldsymbol{\omega}}(t)$$

where $\mathfrak{T}^{\geq 1}$ is the set of trees of depth at least 1. Hence $F(\mathbf{s}) = \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{t\in\mathfrak{T}^{\geq 1}} \tilde{\boldsymbol{\omega}}(t) = \mathbf{s}$.

(ii) Let $\mathfrak{T}^{<k}$ denote the set of all trees with depth at most k. We prove by induction on k that $F^k(\mathbf{0}) = \sum_{t\in\mathfrak{T}^{<k}} \tilde{\boldsymbol{\omega}}(t)$, which implies that $\lim_{k\to\infty} F^k(\mathbf{0}) = \mathbf{s}$. This is straightforward for $k = 1$. Assuming it is true for all naturals up to $k-1$, we have

$$\begin{aligned}
F^k(\mathbf{0}) &= \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, F^{k-1}(\mathbf{0}), \cdots, F^{k-1}(\mathbf{0})) \\
&= \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \sum_{t_1,\cdots,t_p\in\mathfrak{T}^{<k-1}} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \tilde{\boldsymbol{\omega}}(t_1), \cdots, \tilde{\boldsymbol{\omega}}(t_p)) \\
&= \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \sum_{t_1,\cdots,t_p\in\mathfrak{T}^{<k-1}} \tilde{\boldsymbol{\omega}}(g(t_1, \cdots, t_p)) \\
&= \sum_{t\in\mathfrak{T}^{<k}} \tilde{\boldsymbol{\omega}}(t).
\end{aligned}$$

(iii) Let $\mathbf{E}$ be the Jacobian of F around $\mathbf{s}$, we show that the spectral radius $\rho(\mathbf{E})$ of $\mathbf{E}$ is less than one, which implies the result by Ostrowski's theorem (see Ortega, 1990, Theorem 8.1.7).

Since A is minimal, there exist trees $t_1, \cdots, t_n \in \mathfrak{T}$ and contexts $c_1, \cdots, c_n \in \mathfrak{C}$ such that both $\{\boldsymbol{\omega}(t_i)\}_{i\in[n]}$ and $\{\boldsymbol{\alpha}(c_i)\}_{i\in[n]}$ are sets of linear independent vectors in $\mathbb{R}^n$ (Bailly, Habrard, and Denis, 2010). Therefore, the sets $\{\boldsymbol{\omega}(t_i) \otimes \boldsymbol{\omega}(t_j)\}_{i,j\in[n]}$ and $\{\boldsymbol{\alpha}(c_i) \otimes \boldsymbol{\alpha}(c_j)\}_{i,j\in[n]}$ are sets of linear independent vectors in $\mathbb{R}^{n^2}$. Let $\mathbf{v} \in \mathbb{R}^{n^2}$ be an eigenvector of $\mathbf{E}$ with eigenvalue $\lambda \neq 0$, and let $\mathbf{v} = \sum_{i,j\in[n]} \beta_{i,j}(\boldsymbol{\omega}(t_i) \otimes \boldsymbol{\omega}(t_j))$ be its expression in terms of the basis given by $\{\boldsymbol{\omega}(t_i) \otimes \boldsymbol{\omega}(t_j)\}_{i,j}$. For any vector $\mathbf{u} \in \{\boldsymbol{\alpha}(c_i) \otimes \boldsymbol{\alpha}(c_j)\}_{i,j}$ we have

$$\lim_{k\to\infty} \mathbf{u}^\top \mathbf{E}^k \mathbf{v} \leq \lim_{k\to\infty} |\mathbf{u}^\top \mathbf{E}^k \mathbf{v}| \leq \sum_{i,j\in[n]} |\beta_{i,j}| \lim_{k\to\infty} |\mathbf{u}^\top \mathbf{E}^k (\boldsymbol{\omega}(t_i) \otimes \boldsymbol{\omega}(t_j))| = 0,$$

where we used Lemma 3 (see below) in the last step. Since this is true for any vector $\mathbf{u}$ in the basis $\{\boldsymbol{\alpha}(c_i) \otimes \boldsymbol{\alpha}(c_j)\}$, we have $\lim_{k\to\infty} \mathbf{E}^k\mathbf{v} = \lim_{k\to\infty} |\lambda|^k\mathbf{v} = \mathbf{0}$, hence $|\lambda| < 1$. This reasoning holds for any eigenvalue of $\mathbf{E}$, hence $\rho(\mathbf{E}) < 1$.

□

Lemma 3. *Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ be a minimal WTA of dimension n computing the strongly convergent function f, and let $\mathbf{E} \in \mathbb{R}^{n^2\times n^2}$ be the Jacobian around $\mathbf{s} = \sum_{t\in\mathfrak{T}} \boldsymbol{\omega}(t) \otimes \boldsymbol{\omega}(t)$ of the mapping*

$$F \;:\; \mathbf{v} \mapsto \sum_{\sigma\in\Sigma} \tilde{\boldsymbol{\omega}}_\sigma + \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \mathbf{v}, \cdots, \mathbf{v}).$$

Then for any $c_1, c_2 \in \mathfrak{C}$ and any $t_1, t_2 \in \mathfrak{T}$ we have

$$\lim_{k\to\infty} |(\boldsymbol{\alpha}(c_1) \otimes \boldsymbol{\alpha}(c_2))^\top \mathbf{E}^k (\boldsymbol{\omega}(t_1) \otimes \boldsymbol{\omega}(t_2))| = 0.$$

Proof. Let $\tilde{\boldsymbol{\Xi}} : \mathfrak{C} \to \mathbb{R}^{n^2\times n^2}$ be the context mapping associated with the WTA $A^{\otimes}$; i.e. $\tilde{\boldsymbol{\Xi}} = \boldsymbol{\Xi}_{A^{\otimes}}$. We start by proving by induction on $\mathrm{drop}(c)$ that $\tilde{\boldsymbol{\Xi}}(c) = \boldsymbol{\Xi}(c) \otimes \boldsymbol{\Xi}(c)$ for all $c \in \mathfrak{C}$. Let $\mathfrak{C}^d$ denote the set of contexts $c \in \mathfrak{C}$ with $\mathrm{drop}(c) = d$. The statement is trivial for $c \in \mathfrak{C}^0$. Assume the statement is true for all naturals up to $d-1$ and let $c = g(c', t_1, \cdots, t_p) \in \mathfrak{C}^d$ for some $p \geq 0$, $g \in \mathcal{F}_{p+1}$, $t_1, \cdots, t_p \in \mathfrak{T}$ and $c' \in \mathfrak{C}^{d-1}$. Then using our induction hypothesis and the fact that $\tilde{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) \otimes \boldsymbol{\omega}(t)$ for any tree t, we have

$$\begin{aligned}
\tilde{\boldsymbol{\Xi}}(c) &= \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}_{n^2}, \tilde{\boldsymbol{\Xi}}(c'), \tilde{\boldsymbol{\omega}}(t_1), \cdots, \tilde{\boldsymbol{\omega}}(t_p)) \\
&= \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}_{n^2}, \boldsymbol{\Xi}(c') \otimes \boldsymbol{\Xi}(c'), \boldsymbol{\omega}(t_1) \otimes \boldsymbol{\omega}(t_1), \cdots, \boldsymbol{\omega}(t_p) \otimes \boldsymbol{\omega}(t_p)) \\
&= \boldsymbol{\mathcal{A}}^g(\mathbf{I}_n, \boldsymbol{\Xi}(c'), \boldsymbol{\omega}(t_1), \cdots, \boldsymbol{\omega}(t_p)) \otimes \boldsymbol{\mathcal{A}}^g(\mathbf{I}_n, \boldsymbol{\Xi}(c'), \boldsymbol{\omega}(t_1), \cdots, \boldsymbol{\omega}(t_p)) \\
&= \boldsymbol{\Xi}(c) \otimes \boldsymbol{\Xi}(c).
\end{aligned}$$

The case $c = g(t_1, \cdots, t_{i-1}, c', t_i, \cdots, t_p)$ for $i > 1$ follows from an identical argument.

Next we use the multi-linearity of F to expand $F(\mathbf{s}+\mathbf{h})$ for a vector $\mathbf{h} \in \mathbb{R}^{n^2}$. Keeping the terms that are linear in $\mathbf{h}$ we obtain

$$\mathbf{E} = \sum_{p\geq 1} \sum_{g\in\mathcal{F}_p} \sum_{i=0}^{p-1} \tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{i \text{ times}}, \mathbf{I}, \underbrace{\mathbf{s}, \cdots, \mathbf{s}}_{p-1-i \text{ times}}).$$

It follows that $\mathbf{E} = \sum_{c\in\mathfrak{C}^1} \tilde{\boldsymbol{\Xi}}(c)$, and it can be shown by induction on k that $\mathbf{E}^k = \sum_{c\in\mathfrak{C}^k} \tilde{\boldsymbol{\Xi}}(c)$.

Writing $d_c = \min(\mathrm{drop}(c_1), \mathrm{drop}(c_2))$ and $d_t = \min(\mathrm{depth}(t_1), \mathrm{depth}(t_2))$, we can see

that

$$\begin{aligned}
\left|(\boldsymbol{\alpha}(c_1)\otimes\boldsymbol{\alpha}(c_2))^\top \mathbf{E}^k(\boldsymbol{\omega}(t_1)\otimes\boldsymbol{\omega}(t_2))\right| &= \left|\sum_{c\in\mathfrak{C}^k}(\boldsymbol{\alpha}(c_1)\otimes\boldsymbol{\alpha}(c_2))^\top\tilde{\boldsymbol{\Xi}}(c)(\boldsymbol{\omega}(t_1)\otimes\boldsymbol{\omega}(t_2))\right| \\
&= \left|\sum_{c\in\mathfrak{C}^k}(\boldsymbol{\alpha}(c_1)^\top\boldsymbol{\Xi}(c)\boldsymbol{\omega}(t_1))\cdot(\boldsymbol{\alpha}(c_2)^\top\boldsymbol{\Xi}(c)\boldsymbol{\omega}(t_2))\right| \\
&= \left|\sum_{c\in\mathfrak{C}^k} f(c_1[c[t_1]])f(c_2[c[t_2]])\right| \\
&\le \left(\sum_{c\in\mathfrak{C}^k}|f(c_1[c[t_1]])|\right)\left(\sum_{c\in\mathfrak{C}^k}|f(c_2[c[t_2]])|\right) \\
&\le \left(\sum_{t\in\mathfrak{T}^{\ge d_c+d_t+k}}|t||f(t)|\right)^2
\end{aligned}$$

which tends to 0 with $k\to\infty$ since f is strongly convergent.

To prove the last inequality, check that any tree of the form $t'=c[c'[t]]$ satisfies $\mathrm{depth}(t')\ge\mathrm{drop}(c)+\mathrm{drop}(c')+\mathrm{depth}(t)$, and that for fixed $c\in\mathfrak{C}$ and $t,t'\in\mathfrak{T}$ we have $|\{c'\in\mathfrak{C}:c[c'[t]]=t'\}|\le|t'|$ (indeed, a factorization $t'=c[c'[t]]$ is fixed once the root of t is chosen in t', which can be done in at most $|t'|$ different ways).

□

3.C Proof of Theorem 13

Theorem. *There exists $0<\rho<1$ such that after k iterations of line 9 in Algorithm 3, the approximations $\hat{\mathbf{G}}_{\mathfrak{C}}$ and $\hat{\mathbf{G}}_{\mathfrak{T}}$ satisfy $\|\mathbf{G}_{\mathfrak{C}}-\hat{\mathbf{G}}_{\mathfrak{C}}\|_F\le\mathcal{O}(\rho^k)$ and $\|\mathbf{G}_{\mathfrak{T}}-\hat{\mathbf{G}}_{\mathfrak{T}}\|_F\le\mathcal{O}(\rho^k)$.*

Proof. The result for the Gram matrix $\mathbf{G}_{\mathfrak{T}}$ directly follows from Theorem 12. We now show how the error in the approximation of $\mathbf{G}_{\mathfrak{T}}=\mathrm{reshape}(\mathbf{s},n\times n)$ affects the approximation of $\mathbf{q}=(\boldsymbol{\alpha}^{\otimes})^\top(\mathbf{I}-\mathbf{E})^{-1}=\mathrm{vec}(\mathbf{G}_{\mathfrak{C}})$. Let $\hat{\mathbf{s}}\in\mathbb{R}^{n^2}$ be such that $\|\mathbf{s}-\hat{\mathbf{s}}\|\le\varepsilon$, let

$$\hat{\mathbf{E}}=\sum_{p\ge1}\sum_{g\in\mathcal{F}_p}\sum_{i=0}^{p-1}\tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I},\underbrace{\hat{\mathbf{s}},\cdots,\hat{\mathbf{s}}}_{i\text{ times}},\mathbf{I},\ \underbrace{\hat{\mathbf{s}},\cdots,\hat{\mathbf{s}}}_{p-1-i\text{ times}})$$

and let $\mathbf{q}=\tilde{\boldsymbol{\alpha}}^\top(\mathbf{I}-\hat{\mathbf{E}})^{-1}$. We first bound the distance between $\mathbf{E}$ and $\hat{\mathbf{E}}$. For any vector $\mathbf{v}$ we denote by $\mathbf{v}^{\otimes k}=\mathbf{v}\otimes\mathbf{v}\otimes\cdots\otimes\mathbf{v}$ (k times) its kth Kronecker power. Using the fact that for any symbol g of arity p we have $\|\tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I},\mathbf{s},\cdots,\mathbf{s},\mathbf{I},\mathbf{s},\cdots,\mathbf{s})-\tilde{\boldsymbol{\mathcal{A}}}^g(\mathbf{I},\hat{\mathbf{s}},\cdots,\hat{\mathbf{s}},\mathbf{I},\hat{\mathbf{s}},\cdots,\hat{\mathbf{s}})\|_F=\|\mathbf{A}(\mathbf{s}^{\otimes p-1}-\hat{\mathbf{s}}^{\otimes p-1})\|_F$ where $\mathbf{A}$ is the matricization of $\tilde{\boldsymbol{\mathcal{A}}}^g$ obtained by mapping the two modes multiplied by $\mathbf{I}$ to rows and the remaining modes

to columns, we obtain

$$\begin{aligned}\|\mathbf{E}-\hat{\mathbf{E}}\|_F &\leq \sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} p\|\tilde{\boldsymbol{\mathcal{A}}}^g\|_F\|\mathbf{s}^{\otimes p-1}-\hat{\mathbf{s}}^{\otimes p-1}\| \\ &\leq \left(\sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} p\|\tilde{\boldsymbol{\mathcal{A}}}^g\|_F(\|\mathbf{s}\|+\|\hat{\mathbf{s}}\|)^{p-2}\right)\|\mathbf{s}-\hat{\mathbf{s}}\| \\ &\leq \left(\sum_{p\geq 1}\sum_{g\in\mathcal{F}_p} p\|\tilde{\boldsymbol{\mathcal{A}}}^g\|_F(2\|\mathbf{s}\|+\varepsilon)^{p-2}\right)\varepsilon \\ &= \mathcal{O}(\varepsilon)\end{aligned}$$

as $\varepsilon \to 0$, where we used the inequality $\|\mathbf{x}^{\otimes k}-\mathbf{y}^{\otimes k}\| \leq (\|\mathbf{x}\|+\|\mathbf{y}\|)^{k-1}\|\mathbf{x}-\mathbf{y}\|$ for any $k \geq 1$.

Let $\delta = \|\mathbf{E}-\hat{\mathbf{E}}\|$ and let $\mathfrak{s}$ be the smallest nonzero eigenvalue of the matrix $\mathbf{I}-\mathbf{E}$. It follows from (El Ghaoui, 2002, Equation 7.2) that if $\delta < \mathfrak{s}$ then $\|(\mathbf{I}-\mathbf{E})^{-1}-(\mathbf{I}-\hat{\mathbf{E}})^{-1}\| \leq \delta/(\mathfrak{s}(\mathfrak{s}-\delta))$. Since $\delta = \mathcal{O}(\varepsilon)$ from our previous bound, the condition $\delta \leq \mathfrak{s}/2$ will be eventually satisfied as $\varepsilon \to 0$, in which case we can conclude that

$$\|\mathbf{G}_{\mathfrak{C}}-\hat{\mathbf{G}}_{\mathfrak{C}}\|_F = \|\mathbf{q}-\hat{\mathbf{q}}\| \leq \|(\mathbf{I}-\mathbf{E})^{-1}-(\mathbf{I}-\hat{\mathbf{E}})^{-1}\|\|\tilde{\boldsymbol{\alpha}}\| \leq \frac{2\delta}{\mathfrak{s}^2}\|\tilde{\boldsymbol{\alpha}}\| = \mathcal{O}(\varepsilon).$$

□

3.D Proof of Theorem 14

Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\boldsymbol{\mathcal{A}}^g\}_{g\in\mathcal{F}_{\geq 1}}, \{\boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$ be an SVTA with n states computing a function f and let $1 \leq \hat{n} < n$. We consider the WTA with n states

$$\hat{A} = (\mathbb{R}^n, \hat{\boldsymbol{\alpha}} = \boldsymbol{\Pi}\boldsymbol{\alpha}, \{\hat{\boldsymbol{\mathcal{A}}}^g = \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}, \cdots, \boldsymbol{\Pi})\}_{g\in\mathcal{F}_{\geq 1}}, \{\hat{\boldsymbol{\omega}}^\sigma = \boldsymbol{\omega}^\sigma\}_{\sigma\in\mathcal{F}_0})$$

where $\boldsymbol{\Pi} \in \mathbb{R}^{n\times n}$ is the projection matrix defined by $\boldsymbol{\Pi}_{i,i} = 1$ if $i \leq \hat{n}$ and 0 otherwise. It is easy to check that the function $\hat{f}$ computed by the WTA $\hat{A}$ is equal to the one computed by the truncation of the SVTA A to $\hat{n}$ states. Note that for any tree t we have

$$|f(t)-\hat{f}(t)| = |\boldsymbol{\alpha}^\top\boldsymbol{\omega}(t)-\hat{\boldsymbol{\alpha}}^\top\hat{\boldsymbol{\omega}}(t)| = |\boldsymbol{\alpha}^\top(\boldsymbol{\omega}(t)-\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))|.$$

We start by bounding the magnitude of the components of the vectors $\hat{\boldsymbol{\omega}}(t)$ and $(\boldsymbol{\omega}(t)-\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))$ for any tree t in the following lemmas.

Lemma 4. *For any tree $t \in \mathfrak{T}$ and any $i \in [n]$ we have $|\hat{\boldsymbol{\omega}}(t)_i| \leq n^{|t|-1}\sqrt{\mathfrak{s}_i}$.*

Proof. We proceed by induction on the size of t. If $t = \sigma \in \mathcal{F}_0$ we have $|\hat{\boldsymbol{\omega}}(\sigma)_i| = |\boldsymbol{\omega}(\sigma)_i| \leq \sqrt{\mathfrak{s}_i}$ by Proposition 13. Suppose the result true for trees of size less than m

and let $t = g(t_1, \cdots, t_p)$ be a tree of size $m+1$. We have

$$\begin{aligned}
|\hat{\boldsymbol{\omega}}(t)_i| &= |(\boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \cdots, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_p)))_i| \\
&= \left| \sum_{j_1,\cdots,j_p=1}^{n} \boldsymbol{\mathcal{A}}^g_{i,j_1,\cdots,j_p} (\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1))_{j_1} \cdots (\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_p))_{j_p} \right| \\
&\leq \sum_{j_1,\cdots,j_p=1}^{n} \left| \boldsymbol{\mathcal{A}}^g_{i,j_1,\cdots,j_p} \right| |\hat{\boldsymbol{\omega}}(t_1)_{j_1}| \cdots |\hat{\boldsymbol{\omega}}(t_p)_{j_p}| \\
&\leq \sum_{j_1,\cdots,j_p=1}^{n} \frac{\sqrt{\mathfrak{s}_i}}{\sqrt{\mathfrak{s}_{j_1} \cdots \mathfrak{s}_{j_p}}} n^{|t_1|-1} \sqrt{\mathfrak{s}_{j_1}} \cdots n^{|t_p|-1} \sqrt{\mathfrak{s}_{j_p}} \\
&= n^p n^{|t_1|-1} \cdots n^{|t_p|-1} \sqrt{\mathfrak{s}_i} = n^{|t|-1} \sqrt{\mathfrak{s}_i}
\end{aligned}$$

where we used the induction hypothesis and Proposition 13 for the last inequality. □

Lemma 5. *Let $P = \max_{g \in \Sigma} \sharp g$ be the maximum arity of symbols in $\mathcal{F} = (\Sigma, \sharp)$. Then for any tree t and any $i \in [n]$ we have $|\boldsymbol{\omega}(t)_i - (\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))_i| \leq n^{P(|t|-1)} \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}}$.*

Proof. If $i > \hat{n}$ then $|\boldsymbol{\omega}(t)_i - (\Pi\hat{\boldsymbol{\omega}}(t))_i| = |\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i} \leq \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}}$ (where we used Proposition 13 for the first inequality). For $i \leq \hat{n}$ we proceed by induction on the size of t. If $t = \sigma \in \mathcal{F}_0$ we have $|\boldsymbol{\omega}(t)_i - (\Pi\hat{\boldsymbol{\omega}}(t))_i| = |\boldsymbol{\omega}(t)_i - \hat{\boldsymbol{\omega}}(t)_i| = 0$ because $\hat{\boldsymbol{\omega}}^\sigma = \boldsymbol{\omega}^\sigma$. Suppose the result true for trees of size less than m and let $t = g(t_1, \cdots, t_p)$ be a tree of size $m+1$. Since $i \leq \hat{n}$ we have $\boldsymbol{\omega}(t)_i - (\Pi\hat{\boldsymbol{\omega}}(t))_i = (\boldsymbol{\omega}(t) - \hat{\boldsymbol{\omega}}(t))_i$. First, we have

$$\begin{aligned}
\boldsymbol{\omega}(t) - \hat{\boldsymbol{\omega}}(t) &= \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}(t_1), \cdots, \boldsymbol{\omega}(t_p)) - \hat{\boldsymbol{\mathcal{A}}}^g(\mathbf{I}, \hat{\boldsymbol{\omega}}(t_1), \cdots, \hat{\boldsymbol{\omega}}(t_p)) \\
&= \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}(t_1), \cdots, \boldsymbol{\omega}(t_p)) - \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \cdots, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_p)) \\
&= \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\omega}(t_1) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \boldsymbol{\omega}(t_2), \cdots, \boldsymbol{\omega}(t_p)) \\
&+ \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \boldsymbol{\omega}(t_2) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_2), \boldsymbol{\omega}(t_3), \cdots, \boldsymbol{\omega}(t_p)) \\
&+ \cdots + \boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \cdots, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_{p-1}), \boldsymbol{\omega}(t_p) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_p)).
\end{aligned}$$

Now for any $k \in [p]$, using the induction hypothesis and the bounds $|\boldsymbol{\omega}(t)_i| \leq \sqrt{\mathfrak{s}_i}$ (from Proposition 13) and $|\hat{\boldsymbol{\omega}}(t)_i| \leq n^{|t|-1}\sqrt{\mathfrak{s}_i}$ (from the previous lemma) we get

$$\begin{aligned}
&|\boldsymbol{\mathcal{A}}^g(\mathbf{I}, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_1), \cdots, \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_{k-1}), \boldsymbol{\omega}(t_k) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_k), \boldsymbol{\omega}(t_{k+1}), \cdots, \boldsymbol{\omega}(t_p))_i| \\
&\leq \sum_{j_1,\cdots,j_p=1}^{n} \left| \boldsymbol{\mathcal{A}}^g_{i,j_1,\cdots,j_p} \right| \left(\prod_{r=1}^{k-1} |(\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_r))_{j_r}| \right) |(\boldsymbol{\omega}(t_k) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t_k))_{j_k}| \left(\prod_{r=k+1}^{p} |\boldsymbol{\omega}(t_r)_{j_r}| \right) \\
&\leq \sum_{j_1,\cdots,j_p=1}^{n} \frac{\sqrt{\mathfrak{s}_{j_k}}}{\sqrt{\mathfrak{s}_i \mathfrak{s}_{j_1} \cdots \mathfrak{s}_{j_{k-1}} \mathfrak{s}_{j_{k+1}} \cdots \mathfrak{s}_{j_p}}} \left(\prod_{r=1}^{k-1} n^{|t_r|-1} \sqrt{\mathfrak{s}_{j_r}} \right) n^{P(|t_k|-1)} \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_{j_k}}} \left(\prod_{r=k+1}^{p} \sqrt{\mathfrak{s}_{j_r}} \right) \\
&= \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}} n^p \left(\prod_{r=1}^{k-1} n^{|t_r|-1} \right) n^{P(|t_k|-1)} \\
&\leq \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}} n^{P|t_k|} \left(\prod_{r=1}^{k-1} n^{|t_r|-1} \right).
\end{aligned}$$

It follows that

$$|(\boldsymbol{\omega}(t) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))_i| \leq \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}} \sum_{k=1}^{p} n^{P|t_k|} \left(\prod_{r=1}^{k-1} n^{|t_r|-1} \right).$$

To conclude the proof, let $K = \arg\max_{k\in[p]} P|t_k| + \sum_{r=1}^{k-1}(|t_r|-1)$ and observe that

$$\sum_{k=1}^{p} n^{P|t_k|} \left(\prod_{r=1}^{k-1} n^{|t_r|-1} \right) \leq \sum_{j=0}^{K} n^j = \frac{n^{K+1}-1}{n-1} \leq n^{P(|t_1|+\cdots+|t_p|)} = n^{P(|t|-1)}.$$

□

We can now use the two lemmas to show Theorem 14.

Theorem. *Let $P = \max_{g\in\Sigma} \sharp g$ be the maximum arity of symbols in $\mathcal{F} = (\Sigma, \sharp)$. Let $f : \mathfrak{T} \to \mathbb{R}$ be a function computed by an SVTA with n states and let $\hat{f}$ be the function computed by its truncation to $\hat{n}$ states. Then, for any tree $t \in \mathfrak{T}$ we have $|f(t) - \hat{f}(t)| \leq n^{P|t|}\mathfrak{s}_{\hat{n}+1}$.*

Consequently, for any $\varepsilon > 0$ and any tree $t \in \mathfrak{T}$:

$$\text{if } |t| \leq \frac{\log\varepsilon - \log\mathfrak{s}_{\hat{n}+1}}{P\log n} \quad \text{then} \quad |f(t) - \hat{f}(t)| \leq \varepsilon.$$

Proof. For any tree t we have

$$\begin{aligned} |f(t) - \hat{f}(t)| &= |\boldsymbol{\alpha}^\top(\boldsymbol{\omega}(t) - \boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))| \\ &\leq \sum_{i=1}^{n} |\boldsymbol{\alpha}_i| \ |\boldsymbol{\omega}(t)_i - (\boldsymbol{\Pi}\hat{\boldsymbol{\omega}}(t))_i| \\ &\leq \sum_{i=1}^{n} \sqrt{\mathfrak{s}_i} \ n^{P(|t|-1)} \frac{\mathfrak{s}_{\hat{n}+1}}{\sqrt{\mathfrak{s}_i}} \\ &= n^{P|t|}\mathfrak{s}_{\hat{n}+1}. \end{aligned}$$

It is then easy to check that if $|t| \leq \frac{\log\varepsilon - \log\mathfrak{s}_{\hat{n}+1}}{P\log n}$ then $n^{P|t|}\mathfrak{s}_{\hat{n}+1} \leq \varepsilon$. □

4 Low-Rank Regression with Tensor Structured Outputs

Contents

4.1 Introduction

In this chapter, we leave the world of weighted automata and we consider a *supervised learning task* of regression with tensor structured outputs. While in the two previous chapters tensors arose as parameters of a model (graph weighted models and weighted tree automata), in this chapter the connection with tensors comes from the structured nature of the data. Data with a natural tensor structure is encountered in many scientific areas including neuroimaging (Zhou, L. Li, and H. Zhu, 2013), signal processing (Cichocki et al., 2009a), spatio-temporal analysis (Bahadori, Yu, and Liu, 2014) and computer vision (Lu, Plataniotis, and Venetsanopoulos, 2013). Consequently, there has recently been an increasing interest in adapting machine learning and statistical methods to tensors. Extending multivariate regression methods to tensors is one of the challenging tasks in this area. Most existing works extend linear models to the multilinear setting and focus on the tensor structure of the input data (see e.g. Signoretto et al., 2014; Guhaniyogi, Qamar, and Dunson, 2015; Guo, Kotsia, and Patras, 2012; Hou and Chaib-draa, 2015; Zhou, L. Li, and H. Zhu, 2013). Little has been done however to investigate learning methods for *tensor-structured output data*. We will now recall the

classical settings of regression and multivariate regression before motivating the problem of *low rank regression for tensor structured outputs*.

In a classical *regression* task, the learner is given a set of N input/output samples $\{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\} \subset \mathbb{R}^d \times \mathbb{R}$ from which it tries to infer a real valued function $f : \mathbb{R}^d \to \mathbb{R}$. One example of such a task consists in predicting the maximum temperature y in a city on some day given a feature vector $\mathbf{x}$ as input. The input vector could gather measurements of the proportions of certain gazes in the air, or could simply be the values of the maximum temperatures in this city in the preceding days. One way to infer the function f from the training data is to assume a linear dependency between inputs and outputs — that is $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$ for some regression vector $\mathbf{w}$ — and to minimize the squared error loss on the training data $\sum_{n=1}^{N} (\langle \mathbf{w}, \mathbf{x}^{(n)} \rangle - y^{(n)})^2$. In a *multivariate regression* task, the outputs of the function to be learned are vectors instead of scalars, i.e. the learner has to infer a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^p$ from a training sample $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\} \subset \mathbb{R}^d \times \mathbb{R}^p$. The components of the output vector could be for example the maximum temperature, minimum temperature and rain level in a city. In that setting, one could simply infer the vector-valued function f by independently learning a linear model for each component of the output vector: learn $f_1 : \mathbb{R}^d \to \mathbb{R}$ that predicts the maximum temperature, $f_2 : \mathbb{R}^d \to \mathbb{R}$ that predicts the minimum temperature, etc. However, in a multivariate regression task it is very likely that the different components of the output are correlated (e.g. the minimum and maximum temperatures from our previous example are clearly correlated), in which case independently solving the regression task for each component of the outputs seems sub-optimal as this approach does not try to take profit of the correlations between the outputs. Instead, one would want to jointly learn the different scalar-valued functions associated with the output's components to try to share as much knowledge as possible between the different prediction tasks. The *reduced rank regression* approach that we present in the next section achieves this goal by enforcing the model to be of *low rank*.

In this chapter, we consider a regression task where the outputs of the target function f are *higher order tensors*. To motivate this problem let us go back to our meteorological prediction example and suppose now that we want to predict the three same variables (min. and max. temperatures and rain level) in three different cities: Marseille, Paris and Lille. We could simply consider the problem as a multivariate regression task where the outputs have dimension 9 (3 variables times 3 cities), however it is easy to see that in doing so we loose some of the structure of the outputs: representing the outputs as 3×3 matrices, where the rows correspond to cities and the columns to variables, seems more representative of the structure of the data than a 9-dimensional vector. Such a problem setting has been previously considered in e.g. (Romera-Paredes et al., 2013) under the name *multilinear multitask learning* and in the context of spatio-temporal forecasting (Babari and Droste, 2015; Yu, Cheng, and Liu, 2015). Other applications of regression models for tensor structured outputs could be found in the fields of image processing (if for example the outputs of the function f are images) or neuro-sciences where data with a tensor structure is commonly encountered.

In order to leverage the tensor structure of the output data, we formulate the problem as the minimization of a least squares criterion subject to a *multilinear rank* constraint on the regression tensor. The rank constraint enforces the model to capture the low-rank structure of the outputs and to explain the dependencies between inputs and outputs in

a low-dimensional multilinear subspace. Unlike previous work (e.g. Romera-Paredes et al., 2013; Signoretto et al., 2014; Wimalawarne, Sugiyama, and Tomioka, 2014) we do not rely on a convex relaxation of this difficult non-convex optimization problem. Instead we show that it is equivalent to a multilinear subspace identification problem for which we design a fast and efficient approximation algorithm (HOLRR) along with a kernelized version which extends our approach to the nonlinear setting (Section 4.2). The techniques we use to design HOLRR are inspired from the HOSVD algorithm (De Lathauwer, 1997; De Lathauwer, De Moor, and Vandewalle, 2000) that we presented in Chapter 1 (page 18). Our theoretical analysis shows that HOLRR is a quasi-optimal algorithm, that it directly generalizes the reduced rank regression method and that it is statistically consistent. Moreover, we provide a generalization bound for the class of tensor-valued regression functions with bounded multilinear rank that motivates the use of multilinear rank regularization (Section 4.2.3).

Related work. The problem we consider is a generalization of the reduced-rank regression problem (Section 4.1.1) to tensor structured responses. Reduced-rank regression has its roots in statistics (Izenman, 1975) but it has also been investigated by the neural network community (Baldi and Hornik, 1989); non-parametric extensions of this method have been proposed in (Mukherjee and J. Zhu, 2011) and (Foygel et al., 2012). In the context of multi-task learning, a linear model using a tensor-rank penalization of a least squares criterion has been proposed in (Romera-Paredes et al., 2013) to take into account the multi-modal interactions between tasks. Their approach relies on a convex relaxation of the multilinear rank constraint using the trace norms of the matricizations. They also propose a non-convex approach but it is computationally very expensive. An extension of the partial least squares method to the multilinear setting has been proposed in (Zhao et al., 2013). Nonparametric low-rank estimation strategies in reproducing kernel Hilbert spaces (RKHS) based on a multilinear spectral regularization have been proposed in (Signoretto, De Lathauwer, and Suykens, 2013; Signoretto et al., 2014). Their method is based on estimating the regression function in the tensor product of RKHSs and is naturally adapted for tensor covariates. A greedy algorithm to solve a low-rank tensor learning problem has been proposed in (Bahadori, Yu, and Liu, 2014) in the context of multivariate spatio-temporal data analysis. The linear model they assume is different from the one we propose and is specifically designed for spatio-temporal data. The generalization bound we provide is inspired from works on matrix and tensor completion (Srebro, Alon, and Jaakkola, 2004; Nickel and Tresp, 2013).

Summary of the contributions. We propose an *efficient approximation algorithm* for minimizing a least square criterion subject to a low multilinear rank constraint (HOLRR), along with a *kernelized version* that extends our approach to the non linear setting. We show several theoretical properties of HOLRR: it *generalizes the reduced rank regression method*, it is *statistically consistent*, and we derive a *generalization bound* for the class of tensor-valued regression functions with bounded multilinear rank. We compare HOLRR with other multivariate and tensor methods on synthetic and real world data sets where we show that it obtains *better predictive accuracy* while being *computationally very competitive*.

An earlier version of this work where we directly tackled the regression task in the

non linear setting using operator-valued kernels has been presented in the french signal processing conference *GRETSI* in September 2015 (poster) and in a *workshop on tensors and covariance matrix estimation* held in Marseille in November 2015 (30mn talk). The current version of this work has been presented in the french machine learning conference *CAp* in July 2016 (15mn talk) and in the *Mathematics and Statistics Seminar* at Lancaster University in May 2016 (45mn talk), and will be presented at the international conference NIPS 2016 (Rabusseau and Kadri, 2016) (poster). We implemented the HOLRR algorithm in Python using the scikit-tensor package developed by Maximilian Nickel (available at `https://github.com/mnick/scikit-tensor`) and we will release the code under an open source license in the near future.

4.1.1 Low-Rank Regression

Multivariate regression is the task of recovering a function $f : \mathbb{R}^d \to \mathbb{R}^p$ from a set of input-output pairs $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ sampled from the model with an additive noise $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is the error term. To solve this problem, the *ordinary least squares* (OLS) approach assumes a linear dependence between input and output data (i.e. $f : \mathbf{x} \mapsto \mathbf{W}^\top \mathbf{x}$) and boils down to finding a matrix $\mathbf{W} \in \mathbb{R}^{d \times p}$ that minimizes the squared error $\|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2$, where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{Y} \in \mathbb{R}^{N \times p}$ denote the input and the output matrices respectively. To prevent overfitting and to avoid numerical instabilities, a ridge regularization term (i.e. $\gamma\|\mathbf{W}\|_F^2$) is often added to the objective function, leading to the *regularized least squares* (RLS) method. It is easy to see that the OLS/RLS approach in the multivariate setting is equivalent to performing p linear regressions for each scalar output $\{\mathbf{y}_j\}_{j=1}^p$ independently. Thus it performs poorly when the outputs are correlated and the true dimension of the response is less than p. *Low-rank regression* (or reduced-rank regression) addresses this issue by solving the rank penalized problem

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \gamma\|\mathbf{W}\|_F^2 \text{ s.t. } \operatorname{rank}(\mathbf{W}) \leq R \tag{4.1}$$

for a given integer R. The rank constraint was first proposed in (Anderson, 1951), whereas the term *reduced-rank regression* was introduced in (Izenman, 1975). This constraint allows the model to take linear dependencies between the components of the output into account. Adding a ridge regularization was proposed in (Mukherjee and J. Zhu, 2011). In the remaining of this chapter, we will refer to this approach as low-rank regression (LRR).

The constraint $rank(\mathbf{W}) = R$ implies linear restrictions on $\mathbf{W}$, i.e. there must exist $k = \min(d, p) - R$ linearly independent vectors $\mathbf{v}_1, \cdots, \mathbf{v}_k \in \mathbb{R}^p$ such that $\mathbf{W}\mathbf{v}_i = \mathbf{0}$, which, in turn, implies that

$$\langle \mathbf{y}, \mathbf{v}_i \rangle = \langle f(\mathbf{x}) + \boldsymbol{\varepsilon}, \mathbf{v}_i \rangle = \langle \mathbf{W}^\top \mathbf{x} + \boldsymbol{\varepsilon}, \mathbf{v}_i \rangle = \langle \boldsymbol{\varepsilon}, \mathbf{v}_i \rangle$$

for each $i \in [k]$. Intuitively, this means that the whole linear subspace generated by the $\mathbf{v}_i$'s only contains noise. Another way to see this is to write $\mathbf{W} = \mathbf{A}\mathbf{B}$ with $\mathbf{A} \in \mathbb{R}^{d \times R}$ and $\mathbf{B} \in \mathbb{R}^{R \times p}$, which implies that the relation between $\mathbf{x}$ and $\mathbf{y}$ is explained in an R-dimensional latent space.

Even though minimization problem (4.1) is non-convex due to the rank constraint, it admits an analytic solution which is given in the following proposition. This result was

proved in (Izenman, 1975) for $\gamma = 0$ (i.e. without ridge regularization) and in (Mukherjee and J. Zhu, 2011) for the general case where $\gamma \geq 0$.

Proposition 14. *Let* $\mathbf{W}_{RLS} = (\mathbf{X}^\top\mathbf{X}+\gamma\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}$ *be the solution of the regularized least squares, let* $\mathbf{Y}_{RLS} = \mathbf{X}\mathbf{W}_{RLS}$ *and let* $\mathbf{P}$ *be the matrix of the orthogonal projection onto the space spanned by the top* R *eigenvectors of the matrix* $\mathbf{Y}_{RLS}^\top\mathbf{Y}_{RLS} + \gamma\mathbf{W}_{RLS}^\top\mathbf{W}_{RLS}$.

Then, $\mathbf{W}_{LRR} = \mathbf{W}_{RLS}\mathbf{P}$ *is a solution of problem* (4.1).

Proof. This proof summarizes the one given in (Mukherjee and J. Zhu, 2011). First check that the objective function in (4.1) is equal to $\|\tilde{\mathbf{X}}\mathbf{W} - \tilde{\mathbf{Y}}\|_F^2$ where $\tilde{\mathbf{X}} \in \mathbb{R}^{(N+d)\times d}$ and $\tilde{\mathbf{Y}} \in \mathbb{R}^{(N+d)\times p}$ are the block matrices defined by

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\gamma}\,\mathbf{I}_d \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix}.$$

Let $\tilde{\mathbf{Y}}_{RLS} = \tilde{\mathbf{X}}\mathbf{W}_{RLS}$. One can show that

$$\|\tilde{\mathbf{X}}\mathbf{W} - \tilde{\mathbf{Y}}\|^2 = \|\tilde{\mathbf{X}}\mathbf{W} - \tilde{\mathbf{Y}}_{RLS}\|_F^2 + \|\tilde{\mathbf{Y}}_{RLS} - \tilde{\mathbf{Y}}\|_F^2$$

which follows from the orthogonal projection properties of the least squares solution. Thus problem (4.1) is equivalent to

$$\min_{\mathbf{W}:\mathrm{rank}(\mathbf{W})\leq R} \|\tilde{\mathbf{X}}\mathbf{W} - \tilde{\mathbf{Y}}_{RLS}\|_F^2.$$

Let $\mathbf{P}$ be the matrix of the orthogonal projection onto the space spanned by the top R eigenvectors of

$$\tilde{\mathbf{Y}}_{RLS}^\top\tilde{\mathbf{Y}}_{RLS} = \mathbf{Y}_{RLS}^\top\mathbf{Y}_{RLS} + \gamma\mathbf{W}_{RLS}^\top\mathbf{W}_{RLS}.$$

It follows from the Eckart-Young theorem that $\tilde{\mathbf{Y}}_{RLS}\mathbf{P}$ is the best rank R approximation of $\tilde{\mathbf{Y}}_{RLS}$, hence observing that $\tilde{\mathbf{X}}\mathbf{W}_{LRR} = \tilde{\mathbf{X}}\mathbf{W}_{RLS}\mathbf{P} = \tilde{\mathbf{Y}}_{RLS}\mathbf{P}$ concludes the proof. □

For more description and discussion of reduced-rank regression, we refer the reader to the books of Velu and Reinsel (2013) and Izenman (2008).

4.2 Low-Rank Regression for Tensor-Valued Functions

4.2.1 Problem Formulation

We consider a multivariate regression task where the input is a vector and the response has a tensor structure. Let $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1\times d_2\times\cdots\times d_p}$ be the function we want to learn from a sample of input-output data $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^N$ drawn from the model $\mathcal{Y} = f(\mathbf{x}) + \mathcal{E}$, where $\mathcal{E}$ is an error term. We assume that the function f is linear, that is $f(\mathbf{x}) = \mathcal{W} \bullet_1 \mathbf{x}$ for some regression tensor $\mathcal{W} \in \mathbb{R}^{d_0\times d_1\times\cdots\times d_p}$. The vectorization of this relation leads to $\mathrm{vec}(f(\mathbf{x})) = \mathbf{W}_{(1)}^\top\mathbf{x}$ showing that this model is equivalent to the standard multivariate linear model.

Vectorization of the outputs. One way to tackle this linear regression task for tensor responses would be to vectorize each output sample and to perform a standard multivariate low-rank regression on the data $\{(\mathbf{x}^{(n)}, \mathrm{vec}(\mathcal{Y}^{(n)}))\}_{n=1}^N \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_1\cdots d_p}$. A

major drawback of this approach is that the tensor structure of the output is lost in the vectorization step. The low-rank model tries to capture linear dependencies between components of the output but it ignores *higher level dependencies* that could be present in a tensor-structured output. For illustration, suppose the output is a matrix encoding the samples of d_1 continuous variables at d_2 different time steps, one could expect structural relations between the d_1 time series, for example linear dependencies between the rows of the output matrix.

Low-rank regression for tensor responses. To overcome the limitation described above we propose an extension of the low-rank regression method for tensor-structured responses by enforcing low multilinear rank of the regression tensor $\mathcal{W}$. Recall that the multilinear rank of a tensor $\mathcal{T} \in \mathbb{R}^{m_1 \times \cdots \times m_k}$ is a tuple $(R_1, \cdots, R_k)$ where each R_i is equal to the rank of the matricization $\mathbf{T}_{(i)}$. Furthermore, we have $\text{rank}_{ml}(\mathcal{T}) = (R_1, \cdots, R_k)$ if and only if there exist k matrices $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ and a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots R_k}$ such that $\mathcal{T} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_k \mathbf{U}_k$ and $\mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}$ for each $i \in [k]$ (see Section 1.2).

Let $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^N \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_p}$ be a training sample of input/output data drawn from the model $f(\mathbf{x}) = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$ where $\mathcal{W}$ is assumed of low multilinear rank. Considering the framework of empirical risk minimization, we want to find a regression tensor $\mathcal{W}$ of low multilinear rank that minimizes the loss on the training data. To avoid numerical instabilities and to prevent overfitting we add a ridge regularization to the objective function, leading to the following minimization problem

$$\min_{\mathcal{W} \in \mathbb{R}^{d_0 \times \cdots \times d_p}} \sum_{n=1}^{N} \mathcal{L}(\mathcal{W} \bullet_1 \mathbf{x}^{(n)}, \mathcal{Y}^{(n)}) + \gamma \|\mathcal{W}\|_F^2 \quad \text{s.t. } \text{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, \cdots, R_p), \tag{4.2}$$

for some given integers $R_0, R_1, \cdots, R_p$ and where $\mathcal{L}$ is a loss function. In this chapter, we consider the squared error loss between tensors defined by $\mathcal{L}(\mathcal{T}, \hat{\mathcal{T}}) = \|\mathcal{T} - \hat{\mathcal{T}}\|_F^2$. Using this loss we can rewrite problem (4.2) as

$$\min_{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times \cdots \times d_p}} \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 + \gamma \|\mathcal{W}\|_F^2 \quad \text{s.t. } \text{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, \cdots, R_p), \tag{4.3}$$

where the input matrix $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ and the output tensor $\mathcal{Y} \in \mathbb{R}^{N \times d_1 \times \cdots \times d_p}$ are defined by $\mathbf{X}_{n,:} = (\mathbf{x}^{(n)})^\top$ and $\mathcal{Y}_{n,:,\cdots,:} = \mathcal{Y}^{(n)}$ for $n = 1, \cdots, N$ ($\mathcal{Y}$ is the tensor obtained by stacking the output tensors $\mathcal{Y}^{(1)}, \cdots, \mathcal{Y}^{(N)}$ of the training sample along the first mode).

Low-rank regression function. Let $\mathcal{W}^*$ be a solution of problem (4.3), it follows from the multilinear rank constraint that $\mathcal{W}^* = \mathcal{G} \times_1 \mathbf{U}_0 \times_2 \cdots \times_{p+1} \mathbf{U}_p$ for some core tensor $\mathcal{G} \in \mathbb{R}^{R_0 \times \cdots \times R_p}$ and column-wise orthogonal matrices $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ for $0 \leq i \leq p$. The regression function $f^* : \mathbf{x} \mapsto \mathcal{W}^* \bullet_1 \mathbf{x}$ can thus be written as

$$f^* : \mathbf{x} \mapsto \mathcal{G} \times_1 \mathbf{x}^\top \mathbf{U}_0 \times_2 \mathbf{U}_1 \times_3 \cdots \times_{p+1} \mathbf{U}_p.$$

This implies several interesting properties. First, for any $\mathbf{x} \in \mathbb{R}^{d_0}$ we have $f^*(\mathbf{x}) = \mathcal{T}_\mathbf{x} \times_1 \mathbf{U}_1 \times_2 \cdots \times_p \mathbf{U}_p$ with $\mathcal{T}_\mathbf{x} = \mathcal{G} \bullet_1 \mathbf{U}_0^\top \mathbf{x}$, which implies $\text{rank}(f^*(\mathbf{x})) \leq (R_1, \cdots, R_p)$,

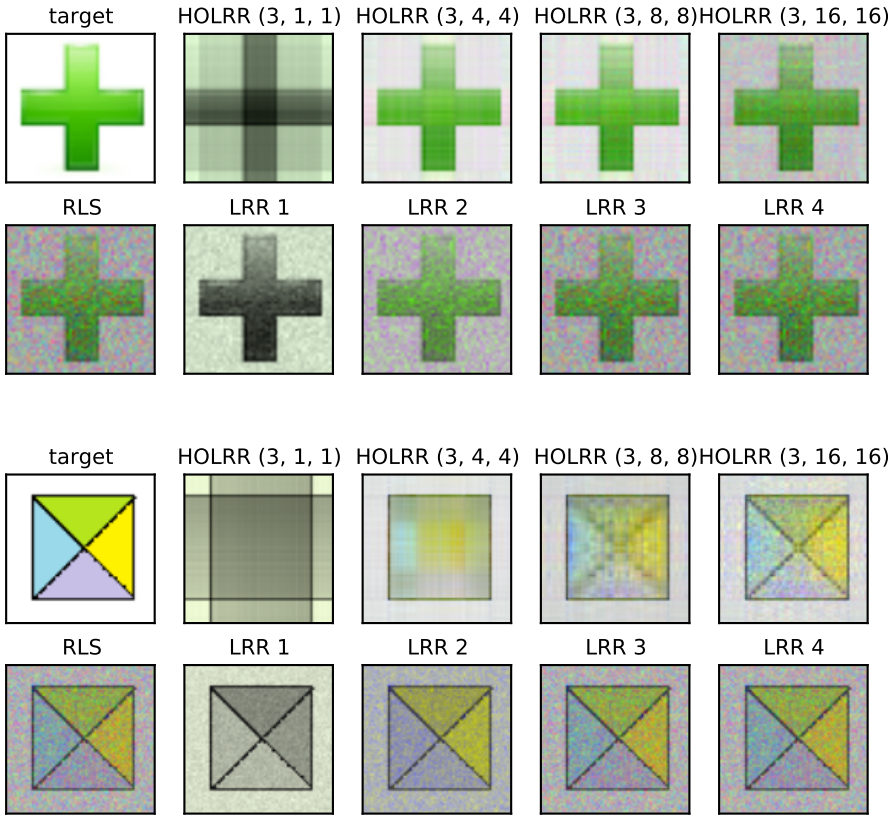


Figure 4.1: Image reconstruction from noisy measurements: $\mathcal{Y} = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$ where $\mathcal{W} \in \mathbb{R}^{3 \times width \times height}$ is a color image (RGB). Since $\mathcal{W}$ is an image we can visualize the estimators returned by the different algorithms. Each image is labeled with the name of the algorithm followed by the rank parameter used to estimate the regression tensor $\mathcal{W}$.

that is the image of f^* is a set of tensors with low multilinear rank. Second, the relation between $\mathbf{x}$ and $\mathcal{Y} = f^*(\mathbf{x})$ is explained in a low dimensional subspace of size $R_0 \times R_1 \times \cdots \times R_p$. Indeed one can decompose the mapping f^* into the following steps: (i) project $\mathbf{x}$ in $\mathbb{R}^{R_0}$ as $\bar{\mathbf{x}} = \mathbf{U}_0^\top \mathbf{x}$, (ii) perform a low-dimensional mapping $\bar{\mathcal{Y}} = \mathcal{G} \bullet_1 \bar{\mathbf{x}}$, (iii) project back into the output space to get $\mathcal{Y} = \bar{\mathcal{Y}} \times_1 \mathbf{U}_1 \times_2 \cdots \times_p \mathbf{U}_p$.

To give an illustrative intuition on the differences between matrix and multilinear rank regularization we present a simple experiment[a] in Figure 4.1. We generate data from the model $\mathcal{Y} = \mathcal{W}^* \bullet_1 \mathbf{x} + \mathcal{E}$ where the tensor $\mathcal{W}^* \in \mathbb{R}^{3 \times m \times n}$ is a color image of size $m \times n$ encoded with three color channels RGB. The components of both $\mathbf{x}$ and $\mathcal{E}$ are independently drawn from $\mathcal{N}(0, 1)$. This experiment allows us to visualize the regression tensors $\mathcal{W}$ returned by RLS, LRR and our method HOLRR that enforces low multilinear rank of the regression function. First, this shows that vectorizing the outputs and performing LRR does not enforce any low-rank structure on the output modes of

[a] An extended version of this experiment is presented in Section 4.3.2.

$\mathcal{W}$ (indeed, the matrix returned by LRR is of size $3 \times mn$ thus of rank at most 3). This is well illustrated in (Figure 4.1) where the regression tensors returned by HOLRR-(3,1,1) are clearly of low rank on the modes corresponding to the width and height of the image while the ones returned by LRR-1 are not. This is to be expected: the rank constraint $(3, 1, 1)$ of HOLRR enforces every rows of the three color channel matrices $\mathcal{W}_{1,:,:}$, $\mathcal{W}_{2,:,:}$ and $\mathcal{W}_{3,:,:}$ to be colinear (and similarly for the columns), in contrast the rank one constraint in LRR only enforces the three color channel matrices to be colinear but no other structure is enforced on each of these matrices. This also shows that taking into account the low-rank structure of the model allows one to better eliminate the noise when the true regression tensor is of low rank (Figure 4.1, top). However if the ground truth model does not have a low-rank structure, enforcing low multilinear rank leads to underfitting for low values of the rank parameter (Figure 4.1, bottom).

4.2.2 Higher-Order Low-Rank Regression and its Kernel Extension

In this section, we propose an efficient algorithm to tackle problem (4.3). We first show that the ridge regularization term in (4.3) can be incorporated in the data fitting term. Let $\tilde{\mathbf{X}} \in \mathbb{R}^{(N+d_0)\times d_0}$ and $\tilde{\mathcal{Y}} \in \mathbb{R}^{(N+d_0)\times d_1 \times \cdots \times d_p}$ be defined by

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\gamma}\,\mathbf{I} \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{Y}}_{(1)} = \begin{bmatrix} \mathbf{Y}_{(1)} \\ \mathbf{0} \end{bmatrix}.$$

It is easy to check that the objective function in (4.3) is equal to $\|\mathcal{W} \times_1 \tilde{\mathbf{X}} - \tilde{\mathcal{Y}}\|_F^2$. Expressing the multilinear rank constraint in terms of a Tucker decomposition of the regression tensor $\mathcal{W}$, minimization problem (4.3) can then be rewritten as

$$\min_{\substack{\mathcal{G}\in\mathbb{R}^{R_0\times R_1\times\cdots\times R_p}, \\ \mathbf{U}_i\in\mathbb{R}^{d_i\times R_i} \text{ for } 0\le i\le p}} \|\mathcal{W} \times_1 \tilde{\mathbf{X}} - \tilde{\mathcal{Y}}\|_F^2 \quad \text{s.t. } \mathcal{W} = \mathcal{G} \times_1 \mathbf{U}_0 \cdots \times_{p+1} \mathbf{U}_p, \mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I} \text{ for all } i. \tag{4.4}$$

We now show that this minimization problem can be reduced to finding $p+1$ projection matrices onto subspaces of dimension $R_0, R_1, \cdots, R_p$. We start by showing that the core tensor $\mathcal{G}$ solution of (4.4) is determined by the factor matrices $\mathbf{U}_0, \cdots, \mathbf{U}_p$.

Theorem 15. *For given column-wise orthogonal matrices $\mathbf{U}_0, \cdots, \mathbf{U}_p$ the tensor $\mathcal{G}$ that minimizes (4.4) is given by*

$$\mathcal{G} = \tilde{\mathcal{Y}} \times_1 (\mathbf{U}_0^\top \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \tilde{\mathbf{X}}^\top \times_2 \mathbf{U}_1^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p^\top.$$

Proof. Since the Frobenius norm of a tensor is equal to the one of its vectorization we can use Eq. (1.4) to write the objective function in (4.4) as

$$\|(\mathbf{U}_p \otimes \mathbf{U}_{p-1} \otimes \cdots \otimes \mathbf{U}_1 \otimes \tilde{\mathbf{X}}\mathbf{U}_0)\mathrm{vec}(\mathcal{G}) - \mathrm{vec}(\tilde{\mathcal{Y}})\|_F^2.$$

Let $\mathbf{M} = \mathbf{U}_p \otimes \mathbf{U}_{p-1} \otimes \cdots \otimes \mathbf{U}_1 \otimes \tilde{\mathbf{X}}\mathbf{U}_0$. The solution with respect to $\mathrm{vec}(\mathcal{G})$ of this classical linear least squares problem is given by $(\mathbf{M}^\top\mathbf{M})^{-1}\mathbf{M}^\top$. Using the mixed-product and inverse properties of the Kronecker product and the column-wise orthogonality of

$\mathbf{U}_1, \cdots, \mathbf{U}_p$ we obtain

$$\operatorname{vec}(\boldsymbol{\mathcal{G}}) = \left(\mathbf{U}_p \otimes \cdots \otimes \mathbf{U}_1 \otimes (\mathbf{U}_0^\top \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \tilde{\mathbf{X}}^\top\right) \operatorname{vec}(\tilde{\boldsymbol{\mathcal{Y}}})$$

which is equivalent to the result of the theorem (using Eq. (1.4) again). □

It follows from Theorem 15 that problem (4.3) is equivalent to

$$\min_{\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}, 0 \leq i \leq p} \|\tilde{\boldsymbol{\mathcal{Y}}} \times_1 \boldsymbol{\Pi}_0 \times_2 \cdots \times_{p+1} \boldsymbol{\Pi}_p - \tilde{\boldsymbol{\mathcal{Y}}}\|_F^2 \tag{4.5}$$

subject to $\mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}$ for all i, $\boldsymbol{\Pi}_0 = \tilde{\mathbf{X}}\mathbf{U}_0 \left(\mathbf{U}_0^\top \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \mathbf{U}_0\right)^{-1} \mathbf{U}_0^\top \tilde{\mathbf{X}}^\top$, and $\boldsymbol{\Pi}_i = \mathbf{U}_i \mathbf{U}_i^\top$ for $i \geq 1$. Note that $\boldsymbol{\Pi}_0$ is the orthogonal projection onto the space spanned by the columns of $\tilde{\mathbf{X}}\mathbf{U}_0$ and $\boldsymbol{\Pi}_i$ is the orthogonal projection onto the column space of $\mathbf{U}_i$ for $i \geq 1$. Hence solving problem (4.3) is equivalent to finding $p+1$ low-dimensional subspaces $U_0, \cdots, U_p$ such that projecting $\tilde{\boldsymbol{\mathcal{Y}}}$ onto the spaces $\tilde{\mathbf{X}}U_0, U_1, \cdots, U_p$ along the corresponding modes is close to $\tilde{\boldsymbol{\mathcal{Y}}}$.

HOLRR algorithm. Since solving problem (4.5) for the $p+1$ projections simultaneously is a difficult non-convex optimization problem we propose to solve it independently for each projection. Note that this is not an unusual strategy, this is the one used for example by the HOSVD algorithm (presented in Chapter 1) to find a low multilinear rank approximation of a tensor. This approach has the benefits of both being computationally efficient and providing good theoretical approximation guarantees (see Theorem 17). The following proposition gives the analytic solutions of (4.5) when each projection is considered independently.

Proposition 15. *For $0 \leq i \leq p$, using the definition of $\boldsymbol{\Pi}_i$ in problem (4.5), the optimal solution of*

$$\min_{\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}} \|\tilde{\boldsymbol{\mathcal{Y}}} \times_{i+1} \boldsymbol{\Pi}_i - \tilde{\boldsymbol{\mathcal{Y}}}\|_F^2 \ \text{s.t.}\ \mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}$$

is given by the eigenvectors of

$$\begin{cases} (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}_{(1)} \tilde{\mathbf{Y}}_{(1)}^\top \tilde{\mathbf{X}} & \text{if } i = 0 \\ \tilde{\mathbf{Y}}_{(i+1)} \tilde{\mathbf{Y}}_{(i+1)}^\top & \text{otherwise} \end{cases}$$

that corresponds to the R_i largest eigenvalues.

Proof. For any $0 \leq i \leq p$, since $\boldsymbol{\Pi}_i$ is a projection we have

$$\langle \tilde{\boldsymbol{\mathcal{Y}}} \times_{i+1} \boldsymbol{\Pi}_i, \tilde{\boldsymbol{\mathcal{Y}}} \rangle = \langle \boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}, \tilde{\mathbf{Y}}_{(i+1)} \rangle = \|\boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}\|_F^2,$$

thus minimizing $\|\tilde{\boldsymbol{\mathcal{Y}}} \times_{i+1} \boldsymbol{\Pi}_i - \tilde{\boldsymbol{\mathcal{Y}}}\|_F^2$ is equivalent to minimizing

$$\|\boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}\|_F^2 - 2\langle \boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}, \tilde{\mathbf{Y}}_{(i+1)} \rangle = -\|\boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}\|_F^2.$$

It turns out that the maximization of $\|\boldsymbol{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}\|_F^2$ is a simple eigenvalue/eigenvector problem when $i \geq 1$ and a generalized eigenvalue/eigenvector problem when $i = 0$.

Indeed, for $i \geq 1$ we have $\|\mathbf{\Pi}_i \tilde{\mathbf{Y}}_{(i+1)}\|_F^2 = \mathrm{Tr}(\mathbf{U}_i^\top \tilde{\mathbf{Y}}_{(i+1)} \tilde{\mathbf{Y}}_{(i+1)}^\top \mathbf{U}_i)$ which is maximized by letting the columns of $\mathbf{U}_i$ be the top R_i eigenvectors of the matrix $\tilde{\mathbf{Y}}_{(i+1)} \tilde{\mathbf{Y}}_{(i+1)}^\top$. For $i = 0$, we have

$$\|\mathbf{\Pi}_0 \tilde{\mathbf{Y}}_{(1)}\|_F^2 = \mathrm{Tr}(\mathbf{\Pi}_0 \tilde{\mathbf{Y}}_{(1)} \tilde{\mathbf{Y}}_{(1)}^\top \mathbf{\Pi}_0^\top) = \mathrm{Tr}\left((\mathbf{U}_0^\top \mathbf{A} \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \mathbf{B} \mathbf{U}_0\right)$$

with $\mathbf{A} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ and $\mathbf{B} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}_{(1)} \tilde{\mathbf{Y}}_{(1)}^\top \tilde{\mathbf{X}}$, which is maximized by letting the columns of $\mathbf{U}_0$ be the top R_0 eigenvectors of $\mathbf{A}^{-1}\mathbf{B}$. □

The results from Theorem 15 and Proposition 15 can be rewritten using the original input matrix $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ and output tensor $\mathcal{Y} \in \mathbb{R}^{N \times d_1 \times \cdots \times d_p}$. Since $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} = \mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}$ and $\tilde{\mathcal{Y}} \times_1 \tilde{\mathbf{X}}^\top = \mathcal{Y} \times_1 \mathbf{X}^\top$, we can rewrite the solution of Theorem 15 as

$$\mathcal{G} = \mathcal{Y} \times_1 (\mathbf{U}_0^\top (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}) \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \mathbf{X}^\top \times_2 \mathbf{U}_1^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p^\top.$$

Similarly for Proposition 15 we have

$$(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}_{(1)} \tilde{\mathbf{Y}}_{(1)}^\top \tilde{\mathbf{X}} = (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{X}$$

and one can check that $\tilde{\mathbf{Y}}_{(i)} \tilde{\mathbf{Y}}_{(i)}^\top = \mathbf{Y}_{(i)} \mathbf{Y}_{(i)}^\top$ for any $i > 1$.

The overall Higher-Order Low-Rank Regression procedure (HOLRR) is summarized in Algorithm 4. Note that similarly to HOSVD, the approximate solution returned by HOLRR could be used as a good initialization point for an iterative method (e.g. Newton method on the manifold of tensors with low multilinear rank or simply an alternating least squares scheme). Studying the theoretical and experimental properties of this approach is left for future work.

A recurrent question about the HOLRR algorithm concerns the choice of the rank parameter $(R_0, \cdots, R_p)$. This parameter can be estimated using cross-validation on the training set (which can be done in a parallel fashion), which is the method that we use in the experiment section. We plan to study alternative approaches in the future. One promising direction would be to design an algorithm able to determine the rank constraint in a greedy fashion: starting with a rank constraint of $(1, \cdots, 1)$, at each step we choose the mode i for which relaxing the rank constraint from R_i to R_i+1 leads to the greater decrease of the loss function (similarly to the algorithm proposed by Bahadori, Yu, and Liu, 2014). Another direction would be to derive a statistical estimator for the multilinear rank (such as the one proposed for the reduced rank regression estimator in Velu and Reinsel, 2013, Section 2.6).

HOLRR Kernel Extension We now design a kernelized version of the HOLRR algorithm by analyzing how HOLRR would be instantiated in a feature space. We show that all the steps involved can be performed using the Gram matrix of the input data without having to explicitly compute the feature map. Let $\phi : \mathbb{R}^{d_0} \to \mathbb{R}^L$ be a feature map and let $\mathbf{\Phi} \in \mathbb{R}^{N \times L}$ be the matrix with rows $\phi(\mathbf{x}^{(n)})^\top$ for $n \in [N]$. The higher-order low-rank

Algorithm 4 HOLRR

Input: Input matrix $\mathbf{X} \in \mathbb{R}^{N \times d_0}$, output tensor $\mathcal{Y} \in \mathbb{R}^{N \times d_1 \times \cdots \times d_p}$, rank parameter $(R_0, R_1, \cdots, R_p)$ and regularization parameter γ.

Output: Regression tensor $\mathcal{W}$ of multilinear rank $(R_0, \cdots, R_p)$ defining the function $f : \mathbf{x} \mapsto \mathcal{W} \bullet_1 \mathbf{x}$.

1: $\mathbf{U}_0 \leftarrow$ top R_0 eigenvectors of $(\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{X}$
2: **for** $i = 1$ **to** p **do**
3: $\quad \mathbf{U}_i \leftarrow$ top R_i eigenvectors of $\mathbf{Y}_{(i+1)} \mathbf{Y}_{(i+1)}^\top$
4: **end for**
5: $\mathbf{M} = \left(\mathbf{U}_0^\top (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}) \mathbf{U}_0 \right)^{-1} \mathbf{U}_0^\top \mathbf{X}^\top$
6: $\mathcal{G} \leftarrow \mathcal{Y} \times_1 \mathbf{M} \times_2 \mathbf{U}_1^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p^\top$
7: **return** $\mathcal{G} \times_1 \mathbf{U}_0 \times_2 \cdots \times_{p+1} \mathbf{U}_p$

regression problem in the feature space boils down to the minimization problem

$$\min_{\mathcal{W} \in \mathbb{R}^{L \times d_1 \times \cdots \times d_p}} \|\mathcal{W} \times_1 \mathbf{\Phi} - \mathcal{Y}\|_F^2 + \gamma \|\mathcal{W}\|_F^2 \quad \text{s.t. } \operatorname{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, \cdots, R_p). \tag{4.6}$$

Following the HOLRR algorithm, one needs to compute the top R_0 eigenvectors of the $L \times L$ matrix $(\mathbf{\Phi}^\top \mathbf{\Phi} + \gamma \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi}$. The following proposition shows that this can be done using the Gram matrix $\mathbf{K} = \mathbf{\Phi} \mathbf{\Phi}^\top$ without explicitly knowing the feature map ϕ.

Proposition 16. *If $\boldsymbol{\alpha} \in \mathbb{R}^N$ is an eigenvector with eigenvalue λ of the $N \times N$ matrix*

$$(\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K},$$

then $\mathbf{v} = \mathbf{\Phi}^\top \boldsymbol{\alpha} \in \mathbb{R}^L$ is an eigenvector with eigenvalue λ of the $L \times L$ matrix

$$(\mathbf{\Phi}^\top \mathbf{\Phi} + \gamma \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi}.$$

Proof. Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ be an eigenvector of $(\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K}$ with eigenvalue λ. We have

$$\begin{aligned} \lambda \mathbf{v} &= \mathbf{\Phi}^\top (\lambda \boldsymbol{\alpha}) = \mathbf{\Phi}^\top \left((\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K} \right) \boldsymbol{\alpha} \\ &= \mathbf{\Phi}^\top (\mathbf{\Phi} \mathbf{\Phi}^\top + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi} \mathbf{\Phi}^\top \boldsymbol{\alpha} \\ &= \left((\mathbf{\Phi}^\top \mathbf{\Phi} + \gamma \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi} \right) \mathbf{v}. \end{aligned}$$

□

Let $\mathbf{A}$ be the top R_0 eigenvectors of the matrix $(\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K}$. When working with the feature map ϕ, it follows from the previous proposition that line 1 in Algorithm 4 is equivalent to choosing $\mathbf{U}_0 = \mathbf{\Phi}^\top \mathbf{A} \in \mathbb{R}^{L \times R_0}$, while the updates in line 3 stay the same. The regression tensor $\mathcal{W} \in \mathbb{R}^{L \times d_1 \times \cdots \times d_p}$ returned by this algorithm is then equal

to $\mathcal{W} = \mathcal{Y} \times_1 \mathbf{P} \times_2 \mathbf{U}_1\mathbf{U}_1^\top \times_2 \cdots \times_{p+1} \mathbf{U}_p\mathbf{U}_p^\top$, where

$$\begin{aligned}\mathbf{P} &= \mathbf{\Phi}^\top\mathbf{A}\left(\mathbf{A}^\top\mathbf{\Phi}(\mathbf{\Phi}^\top\mathbf{\Phi}+\gamma\mathbf{I})\mathbf{\Phi}^\top\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{\Phi}\mathbf{\Phi}^\top \\ &= \mathbf{\Phi}^\top\mathbf{A}\left(\mathbf{A}^\top\mathbf{\Phi}\mathbf{\Phi}^\top(\mathbf{\Phi}\mathbf{\Phi}^\top+\gamma\mathbf{I})\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{\Phi}\mathbf{\Phi}^\top \\ &= \mathbf{\Phi}^\top\mathbf{A}\left(\mathbf{A}^\top\mathbf{K}(\mathbf{K}+\gamma\mathbf{I})\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{K}.\end{aligned}$$

Suppose now that the feature map ϕ is induced by a kernel function $k : \mathbb{R}^{d_0} \times \mathbb{R}^{d_0} \to \mathbb{R}$. The prediction for an input vector $\mathbf{x}$ is then given by $\mathcal{W} \bullet_1 \mathbf{x} = \mathcal{C} \bullet_1 \mathbf{k}_\mathbf{x}$ where the nth component of $\mathbf{k}_\mathbf{x} \in \mathbb{R}^N$ is $\langle\phi(\mathbf{x}^{(n)}), \phi(\mathbf{x})\rangle = k(\mathbf{x}^{(n)}, \mathbf{x})$ and the tensor $\mathcal{C} \in \mathbb{R}^{N\times d_1\times\cdots\times d_p}$ is defined by

$$\mathcal{C} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{U}_1 \times_2 \cdots \times_{p+1} \mathbf{U}_p$$

where

$$\mathcal{G} = \mathcal{Y} \times_1 \left(\mathbf{A}^\top\mathbf{K}(\mathbf{K}+\gamma\mathbf{I})\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{K} \times_2 \mathbf{U}_2^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p^\top \in \mathbb{R}^{R_0\times\cdots\times R_p}.$$

Observe that $\mathcal{C}$ has multilinear rank at most $(R_0, \cdots, R_p)$, hence the low multilinear rank constraint on $\mathcal{W}$ in the feature space translates into the low rank structure of the coefficient tensor $\mathcal{C}$.

Let $\mathcal{H}$ be the reproducing kernel Hilbert space associated with the kernel k. The overall procedure for kernelized HOLRR is summarized in Algorithm 5. This algorithm returns the tensor $\mathcal{C} \in \mathbb{R}^{N\times d_1\times\cdots\times d_p}$ defining the regression function

$$f : \mathbf{x} \mapsto \mathcal{C} \bullet_1 \mathbf{k}_\mathbf{x} = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}^{(n)})\mathcal{C}^{(n)},$$

where $\mathcal{C}^{(n)} = \mathcal{C}_{n:\cdots:} \in \mathbb{R}^{d_1\times\cdots\times d_p}$.

Algorithm 5 `Kernelized HOLRR`

Input: Gram matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$, output tensor $\mathcal{Y} \in \mathbb{R}^{N\times d_1\times\cdots\times d_p}$, rank parameter $(R_0, R_1, \cdots, R_p)$ and regularization parameter γ.

Output: Coefficient tensor $\mathbf{C}$ defining the function $\mathbf{x} \mapsto \mathbf{C} \bullet_1 \mathbf{k}_\mathbf{x}$ where $\mathbf{k}_\mathbf{x} \in \mathbb{R}^N$ is defined by $(\mathbf{k}_\mathbf{x})_n = k(\mathbf{x}^{(n)}, \mathbf{x})$.

$\mathbf{A} \leftarrow$ top R_0 eigenvectors of $(\mathbf{K}+\gamma\mathbf{I})^{-1}\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^\top\mathbf{K}$
for $i = 1$ **to** p **do**
  $\mathbf{U}_i \leftarrow$ top R_i eigenvectors of $\mathbf{Y}_{(i+1)}\mathbf{Y}_{(i+1)}^\top$
end for
$\mathbf{M} \leftarrow \left(\mathbf{A}^\top\mathbf{K}(\mathbf{K}+\gamma\mathbf{I})\mathbf{A}\right)^{-1}\mathbf{A}^\top\mathbf{K}$
$\mathcal{G} \leftarrow \mathcal{Y} \times_1 \mathbf{M} \times_2 \mathbf{U}_1^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p^\top$
return $\mathcal{C} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{U}_1 \times_3 \cdots \times_{p+1} \mathbf{U}_p$

The kernelized version of HOLRR we developed in this section relies on the so-called *kernel trick* which consists in expressing all the operations used in an algorithm in terms of inner product between the input samples $\langle\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\rangle$, and replacing this inner product by the inner product in some feature space where the inputs have been embedded. In

particular, one can use the inner product in the reproducing kernel Hilbert space (RKHS) induced by some kernel function. Since we are dealing with tensor-valued functions, an alternative natural non-linear extension of HOLRR could be designed in the *operator-valued kernels* framework. These kernels were first proposed in (Micchelli and Pontil, 2005) to extend the classical learning theory of scalar-valued functions in an RKHS to vector-valued functions, which led to several interesting applications (see e.g. Alvarez, Rosasco, and Lawrence, 2012; Kadri et al., 2015). From this perspective, the approach we proposed only considers the case of scalar-valued kernel functions, and it would be interesting to consider operator-valued kernels from a multilinear perspective and to extend K-HOLRR to this framework. The analysis provided in (Signoretto, De Lathauwer, and Suykens, 2013) could prove useful in this regard. We already started pursuing this line of research during my PhD (Rabusseau, Kadri, and Denis, 2015) using the notion of CP rank instead of the multilinear rank, which made it more difficult to obtain approximation guarantees. Another obstacle that we encountered was the computational cost of using operator-valued kernels when the output dimensions are large. Nonetheless, we think that this line of research is worth pursuing and we will do so in the future.

4.2.3 Theoretical Analysis

In this section we study theoretical properties of HOLRR. After analyzing its running time complexity and showing that it generalizes the low rank regression approach, we show that it is a quasi-optimal algorithm and that it is statistically consistent. We will conclude this section by giving a generalization bound for the class of tensor-valued regression functions with low multilinear rank.

Complexity analysis. We first compare the computational complexity of LRR and HOLRR. The LRR algorithm first needs to compute the regularized least square estimator which can be done in $\mathcal{O}\left((d_0)^3 + N((d_0)^2 + d_0 d_1 \cdots d_p)\right)$, and then to compute the top R eigenvectors of a matrix of the same size as $\mathbf{Y}_{(1)}^\top \mathbf{Y}_{(1)}$ which can be done in $\mathcal{O}\left((N+R)(d_1 \cdots d_p)^2\right)$ (including the cost for matrix multiplication), which leads to an overall complexity of

$$\mathcal{O}\left((d_0)^3 + N((d_0)^2 + d_0 d_1 \cdots d_p) + (N+R)(d_1 \cdots d_p)^2\right).$$

The HOLRR algorithm also needs to compute the regularized least square estimator $\mathbf{W}_{RLS}$ and to compute the R_0 dominant eigenvectors of the matrix $\mathbf{W}_{RLS}\mathbf{Y}_{(1)}^\top \mathbf{X}$ (line 1) which can all be done in $\mathcal{O}\left((d_0)^3 + (R_0+N)(d_0)^2 + N d_0 \cdots d_p\right)$. Each computation of the R_i top eigenvectors of $\mathbf{Y}_{(i)}\mathbf{Y}_{(i)}^\top$ in line 3 can be done in $\mathcal{O}\left(R_i (d_i)^2 + d_i N d_1 \cdots d_p\right)$. Observing that the computations in line 5 are cheaper than the one in line 1 since $R_0 \leq d_0$, we get an overall complexity of

$$\mathcal{O}\left((d_0)^3 + N((d_0)^2 + d_0 d_1 \cdots d_p) + \max_{i \geq 0} R_i (d_i)^2 + N d_1 \cdots d_p \max_{i \geq 1} d_i\right).$$

Since the complexity of HOLRR only have a linear dependence on the product of the output dimensions instead of a quadratic one for LRR, we can conclude that HOLRR will be more efficient than LRR when the output dimensions $d_1, \cdots, d_p$ are large. It is

worth mentioning that the method proposed in (Romera-Paredes et al., 2013) to solve a convex relaxation of problem 4.4 is an iterative algorithm (relying on the alternating direction method of multipliers) that needs to compute SVDs of matrices of size $d_i \times d_1 \cdots d_{i-1} d_{i+1} \cdots d_p$ for each $0 \leq i \leq p$ *at each iteration*, it is thus computationally more expensive than HOLRR. Moreover, since HOLRR only relies on simple linear algebra tools, readily available methods could be used to further improve the speed of the algorithm, e.g. randomized-SVD (Halko, Martinsson, and Tropp, 2011) and random feature approximation of the kernel function (Kar and Karnick, 2012; Rahimi and Recht, 2007).

Relation between HOLRR and LRR. The following theorem shows that when the outputs of the regression task are vectors (i.e. 1st order tensors), HOLRR will return the same estimator as LRR for appropriate values of the rank constraint.

Theorem 16. *Let $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^N \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_1}$ be a training sample of input/output pairs. Let $1 \leq R \leq d_0$ and let $\gamma \geq 0$.*

Then, the estimator $\mathbf{W}_{HOLRR} \in \mathbb{R}^{d_0 \times d_1}$ returned by the HOLRR algorithm with the rank constraint set to $(R_0, R_1) = (R, d_1)$ and with ridge regularization parameter γ is equal to the estimator $\mathbf{W}_{LRR}$ returned by the LRR algorithm with a rank constraint set to R and with the same ridge regularization parameter γ.

Proof. Let $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ and $\mathbf{Y} \in \mathbb{R}^{N \times d_1}$ denote the input and output matrices respectively. First recall that the estimator returned by LRR is

$$\mathbf{W}_{LRR} = \mathbf{W}_{RLS} \mathbf{V} \mathbf{V}^\top$$

where $\mathbf{W}_{RLS} = (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}$ is the regularized least square estimator and $\mathbf{V} \in \mathbb{R}^{d_1 \times R}$ is the matrix having for columns the top R eigenvectors of

$$\mathbf{A} \triangleq \mathbf{W}_{RLS}^\top (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}) \mathbf{W}_{RLS}$$

(see Proposition 14 or Mukherjee and J. Zhu, 2011). Using the notations from Algorithm 4, since $R_1 = d_1$ we have $\mathbf{U}_1 \mathbf{U}_1^\top = \mathbf{I}$ and it follows that

$$\mathbf{W}_{HOLRR} = \mathbf{U}_0 (\mathbf{U}_0^\top (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}) \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \mathbf{X}^\top \mathbf{Y}$$

where $\mathbf{U}_0 \in \mathbb{R}^{d_0 \times R}$ is the matrix having for columns the top R eigenvectors of

$$\mathbf{B} \triangleq (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y} \mathbf{Y}^\top \mathbf{X}.$$

It is easy to check that $\mathbf{A} = \mathbf{Y}^\top \mathbf{X} \mathbf{W}_{RLS}$ and $\mathbf{B} = \mathbf{W}_{RLS} \mathbf{Y}^\top \mathbf{X}$, which implies from the definition of eigenvectors that $\mathbf{U}_0 = \mathbf{W}_{RLS} \mathbf{V}$ (indeed if $\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$ then $\lambda(\mathbf{W}_{RLS}\mathbf{v}) = \mathbf{W}_{RLS}\mathbf{A}\mathbf{v} = \mathbf{B}(\mathbf{W}_{RLS}\mathbf{v})$). We obtain

$$\begin{aligned}
\mathbf{W}_{HOLRR} &= \mathbf{W}_{RLS} \mathbf{V} \Big((\mathbf{W}_{RLS} \mathbf{V})^\top (\mathbf{X}^\top \mathbf{X} + \gamma \mathbf{I}) \mathbf{W}_{RLS} \mathbf{V} \Big)^{-1} (\mathbf{W}_{RLS} \mathbf{V})^\top \mathbf{X}^\top \mathbf{Y} \\
&= \mathbf{W}_{RLS} \mathbf{V} (\mathbf{V}^\top \mathbf{A} \mathbf{V})^{-1} \mathbf{V}^\top \mathbf{A}.
\end{aligned}$$

Let $\mathbf{A} = \tilde{\mathbf{V}}\mathbf{D}\tilde{\mathbf{V}}^\top$ be the eigen-decomposition of the matrix $\mathbf{A}$ so that $\mathbf{V} = \tilde{\mathbf{V}}\mathbf{P}$ where $\mathbf{P} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}^\top \in \mathbb{R}^{d_0 \times R}$. We get

$$\begin{aligned}\mathbf{W}_{HOLRR} &= \mathbf{W}_{RLS}\tilde{\mathbf{V}}\mathbf{P}(\mathbf{P}^\top\tilde{\mathbf{V}}^\top\tilde{\mathbf{V}}\mathbf{D}\tilde{\mathbf{V}}^\top\tilde{\mathbf{V}}\mathbf{P})^{-1}\mathbf{P}^\top\tilde{\mathbf{V}}^\top\tilde{\mathbf{V}}\mathbf{D}\tilde{\mathbf{V}}^\top \\ &= \mathbf{W}_{RLS}\tilde{\mathbf{V}}\mathbf{P}\mathbf{P}^\top\mathbf{D}^{-1}\mathbf{P}\mathbf{P}^\top\mathbf{D}\tilde{\mathbf{V}}^\top \\ &= \mathbf{W}_{RLS}\tilde{\mathbf{V}}\mathbf{P}\mathbf{P}^\top\tilde{\mathbf{V}}^\top \\ &= \mathbf{W}_{RLS}\mathbf{V}\mathbf{V}^\top = \mathbf{W}_{LRR}.\end{aligned}$$ □

One can easily check that the LRR estimator is equal to the regularized least square estimator when the rank constraint is set to either the input or the output dimension of the regression task. Thus the previous theorem implies that HOLRR also generalizes the regularized least square method.

Approximation guarantees. It is easy to check that problem (4.3) is NP-hard since it generalizes the problem of fitting a Tucker decomposition (Hillar and Lim, 2013). The following theorem shows that HOLRR is a $(p+1)$-approximation algorithm for this problem, i.e. HOLRR is a *quasi-optimal* algorithm. This result generalizes the approximation guarantees provided by the HOSVD algorithm for the problem of finding the best low multilinear rank approximation of an arbitrary tensor (see Chapter 1).

Theorem 17. *Let $\mathcal{W}^*$ be a solution of problem (4.3) and let $\mathcal{W}$ be the regression tensor returned by Algorithm 4. If $\mathcal{L} : \mathbb{R}^{d_0 \times \cdots \times d_p} \to \mathbb{R}$ denotes the objective function of (4.3) with respect to $\mathcal{W}$ then*

$$\mathcal{L}(\mathcal{W}) \leq (p+1)\mathcal{L}(\mathcal{W}^*).$$

Proof. The proof follows the one given in Chapter 1 to obtain the approximation guarantees for the HOSVD algorithm. Let $\mathbf{U}_0, \cdots, \mathbf{U}_p$ be the matrices defined in Algorithm 4 and let $\mathbf{\Pi}_0, \cdots, \mathbf{\Pi}_p$ be the orthogonal projection matrices defined in problem (4.5). The regression tensor $\mathcal{W}$ returned by HOLRR satisfies

$$\mathcal{W} \times_1 \tilde{\mathbf{X}} = \tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p.$$

Similarly, it follows from Theorem 15 that a solution $\mathcal{W}^*$ of problem (4.3) satisfies

$$\mathcal{W}^* \times_1 \tilde{\mathbf{X}} = \tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0^* \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p^*$$

for some orthogonal projection matrices $\mathbf{\Pi}_i^*$ for $0 \leq i \leq p$.

Using successive applications of Lemma 1 (page 18) we obtain

$$\mathcal{L}(\mathcal{W}) = \|\mathcal{W} \times_1 \tilde{\mathbf{X}} - \tilde{\mathcal{Y}}\|_F^2 = \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p - \tilde{\mathcal{Y}}\|_F^2 \leq \sum_{i=0}^{p} \|\tilde{\mathcal{Y}} \times_{i+1} \mathbf{\Pi}_i - \tilde{\mathcal{Y}}\|_F^2.$$

By Proposition 15, each summand in this upper bound is minimal with respect to $\mathbf{\Pi}_i$, hence $\|\tilde{\mathcal{Y}} \times_{i+1} \mathbf{\Pi}_i - \tilde{\mathcal{Y}}\|_F^2 \leq \|\tilde{\mathcal{Y}} \times_{i+1} \mathbf{\Pi}_i^* - \tilde{\mathcal{Y}}\|_F^2$ for any $i \in [p]$. It remains to show that

$$\|\tilde{\mathcal{Y}} \times_{i+1} \mathbf{\Pi}_i^* - \tilde{\mathcal{Y}}\|_F^2 \leq \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0^* \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p^* - \tilde{\mathcal{Y}}\|_F^2 = \mathcal{L}(\mathcal{W}^*)$$

for all $i \in [p]$. Indeed, using the fact that the Frobenius norm of a tensor is equal to the one of its matricization and Eq. (1.3) for the matricization of a Tucker decomposition, we obtain for the case $i = 0$

$$\begin{aligned}
\|\tilde{\boldsymbol{\mathcal{Y}}} \times_1 \boldsymbol{\Pi}_0^* \times_2 \cdots \times_{p+1} \boldsymbol{\Pi}_p^* - \tilde{\boldsymbol{\mathcal{Y}}}\|^2 &= \|\boldsymbol{\Pi}_0^* \tilde{\mathbf{Y}}_{(1)} (\boldsymbol{\Pi}_p^* \otimes \cdots \otimes \boldsymbol{\Pi}_1^*)^\top - \tilde{\mathbf{Y}}_{(1)}\|_F^2 \\
&= \|(\boldsymbol{\Pi}_0^* - \mathbf{I}_{d_0})\tilde{\mathbf{Y}}_{(1)} + \boldsymbol{\Pi}_0^* \tilde{\mathbf{Y}}_{(1)} (\boldsymbol{\Pi}_p^* \otimes \cdots \otimes \boldsymbol{\Pi}_1^* - \mathbf{I}_{d_1 d_2 \cdots d_p})^\top\|_F^2 \\
&= \|(\boldsymbol{\Pi}_0^* - \mathbf{I}_{d_0})\tilde{\mathbf{Y}}_{(1)}\|_F^2 + \|\boldsymbol{\Pi}_0^* \tilde{\mathbf{Y}}_{(1)} (\boldsymbol{\Pi}_p^* \otimes \cdots \otimes \boldsymbol{\Pi}_1^* - \mathbf{I}_{d_1 d_2 \cdots d_p})^\top\|_F^2 \\
&\geq \|(\boldsymbol{\Pi}_0^* - \mathbf{I}_{d_0})\tilde{\mathbf{Y}}_{(1)}\|_F^2 \\
&= \|\tilde{\boldsymbol{\mathcal{Y}}} \times_1 \boldsymbol{\Pi}_0^* - \tilde{\boldsymbol{\mathcal{Y}}}\|_F^2
\end{aligned}$$

where we used the orthogonality of $\boldsymbol{\Pi}_0^*$ and $\boldsymbol{\Pi}_0^* - \mathbf{I}_{d_0}$. The proofs for other values of i are similar. □

Consistency of HOLRR. In the following theorem we show that the estimator returned by HOLRR is statistically consistent. This means that as the size of the training sample grows to infinity, the estimator returned by HOLRR converges in probability to the true regression tensor $\boldsymbol{\mathcal{W}}^*$ used to generate the sampling data. Of course this result relies on assumptions on the distributions used to generate the training sample.

Theorem 18. *Let $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}$ be independent and identically distributed (i.i.d.) random variables taking their values in $\mathbb{R}^{d_0}$ and following a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Let $\boldsymbol{\xi}^{(1)}, \cdots, \boldsymbol{\xi}^{(N)}$ be i.i.d. random variables taking their values in $\mathbb{R}^{d_1 \times \cdots \times d_p}$ such that for any $n \in [N]$ each component of $\boldsymbol{\xi}^{(n)}$ follows a normal distribution $\mathcal{N}(0, \sigma^2)$. Finally, let $\boldsymbol{\mathcal{W}}^* \in \mathbb{R}^{d_0 \times d_1 \times \cdots \times d_p}$ be a regression tensor with multilinear rank $(R_0, R_1, \cdots, R_p)$ and let $\boldsymbol{\mathcal{Y}}^{(n)} = \boldsymbol{\mathcal{W}}^* \bullet_1 \mathbf{x}^{(n)} + \boldsymbol{\xi}^{(n)}$ for all $n \in [N]$.*

Let $\boldsymbol{\mathcal{W}}_N$ be the estimator returned by HOLRR with the training sample $\{(\mathbf{x}^{(n)}, \boldsymbol{\mathcal{Y}}^{(n)})\}_{n=1}^N$ as input, with rank parameter $(R_0, R_1, \cdots, R_p)$ and regularization parameter $\gamma = 0$. Then, for any $\varepsilon > 0$ we have

$$\lim_{N \to \infty} \mathbb{P}[\|\boldsymbol{\mathcal{W}}^* - \boldsymbol{\mathcal{W}}_N\|_F > \varepsilon] = 0.$$

Proof. The proof is given in Appendix 4.A. The techniques we use are close to the ones used to show that the ordinary least square estimator is consistent. The key technical difficulty resides in the fact that the HOLRR estimator depends on computations of eigenvectors and that the notion of probability in the limit of an eigenvector is tedious (think for example of the case where this eigenvector belongs to an eigenspace of dimension greater than 1). We bypass this difficulty by reasoning on the orthogonal projection matrices onto the space spanned by the eigenvectors rather than on the eigenvectors directly. □

Note that the previous theorem still holds if we use a rank parameter $(\hat{R}_0, \hat{R}_1, \cdots, \hat{R}_p)$ for HOLRR that overestimates the multilinear rank of $\boldsymbol{\mathcal{W}}^*$ (i.e. $\hat{R}_i \geq R_i$ for all i). This is somehow intuitive since the ordinary least square estimator, which does not enforce any rank constraint, is consistent. However, a simple simulation study suggests that the convergence rate of the HOLRR estimator is slower when the multilinear rank is

overestimated (see Section 4.3.1). We plan to derive sample complexity bounds that would theoretically confirm this behavior in the near future.

Generalization Bound. We conclude this section by deriving a generalization bound for the regression problem with tensor outputs, to the best of our knowledge this is the first result on generalization ability of low rank tensor regression methods. More precisely, the following theorem gives an upper bound on the excess-risk for the function class

$$\mathcal{F} = \{\mathbf{x} \mapsto \boldsymbol{\mathcal{W}} \bullet_1 \mathbf{x} \ : \ \text{rank}_{ml}(\boldsymbol{\mathcal{W}}) \leq (R_0, \cdots, R_p)\}$$

of tensor-valued regression functions with bounded multilinear rank. Recall that the expected loss of an hypothesis $h \in \mathcal{F}$ with respect to the target function f^* is defined by $R(h) = \mathbb{E}_{\mathbf{x}}[\mathcal{L}(h(\mathbf{x}), f^*(\mathbf{x}))]$ and its empirical loss by $\hat{R}(h) = \sum_{n=1}^{N} \frac{1}{N} \mathcal{L}(h(\mathbf{x}^{(n)}), f^*(\mathbf{x}^n))$.

Theorem 19. *Let $\mathcal{L} : \mathbb{R}^{d_1 \times \cdots \times d_p} \to \mathbb{R}^+$ be a loss function satisfying*

$$\mathcal{L}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) = \frac{1}{d_1 \cdots d_p} \sum_{i_1, \cdots, i_p} \ell(\boldsymbol{\mathcal{A}}_{i_1, \cdots, i_p}, \boldsymbol{\mathcal{B}}_{i_1, \cdots, i_p})$$

for some loss function $\ell : \mathbb{R} \to \mathbb{R}^+$ bounded by M. Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample of size N, the following inequality holds for all $h \in \mathcal{F}$:

$$R(h) \leq \hat{R}(h) + M\sqrt{2D \log\left(\frac{4e(p+2)d_0 d_1 \cdots d_p}{d_0 + d_1 + \cdots + d_p}\right) \frac{\log N}{N}} + M\sqrt{\frac{\log(1/\delta)}{2N}},$$

where $D = R_0 R_1 \cdots R_p + \sum_{i=0}^{p} R_i d_i$.

Proof. (Sketch) The complete proof is given in Appendix 4.B. It relies on bounding the pseudo-dimension of the class of real-valued functions with domain $\mathbb{R}^{d_0} \times [d_1] \times \cdots \times [d_p]$

$$\tilde{\mathcal{F}} = \left\{(\mathbf{x}, i_1, \cdots, i_p) \mapsto (\boldsymbol{\mathcal{W}} \bullet_1 \mathbf{x})_{i_1, \cdots, i_p} \ : \ \text{rank}_{ml}(\boldsymbol{\mathcal{W}}) = (R_0, \cdots, R_p)\right\}.$$

We show in the appendix that the pseudo-dimension of $\tilde{\mathcal{F}}$ is upper bounded by

$$(R_0 R_1 \cdots R_p + \sum_{i=0}^{p} R_i d_i) \log\left(\frac{4e(p+2)d_0 d_1 \cdots d_p}{d_0 + d_1 + \cdots + d_p}\right).$$

This is done by leveraging the following result originally due to (Warren, 1968): the number of sign patterns of r polynomials, each of degree at most d, over q variables is at most $(4edr/q)^q$ for all $r > q > 2$ (Srebro, 2004, Theorem 34, 35). The remaining of the proof consists in showing that the risk (resp. empirical risk) of hypothesis in $\mathcal{F}$ and $\tilde{\mathcal{F}}$ are closely related and invoking standard error generalization bounds in terms of the pseudo-dimension (Mohri, Rostamizadeh, and Talwalkar, 2012, Theorem 10.6). □

A standard generalization bound based on the pseudo-dimension for multivariate regression without low-rank constraint would involve a term in $\mathcal{O}\left(\sqrt{d_0 d_1 \cdots d_p}\right)$. In contrast, the bound from the previous theorem only depends on the product of the output dimensions in a term bounded by $\mathcal{O}\left(\sqrt{\log(d_1 \cdots d_p)}\right)$. In some sense, taking

Table 4.1: Average running times in seconds for some of the experiments. We did not run MLMT-NC on the real world data sets because it is computationally very expensive. The implementation of the Greedy algorithm is limited to 2nd order output tensors, this is why we did not run it on the synthetic and Meteo UK data sets. Finally, the synthetic non linear data was generated using a polynomial relation which is why the RBF kernel was not used on this data set.

Data set	MLMTL-NC	ADMM	Greedy	HOPLS	HOLRR	K-HOLRR (poly)	K-HOLRR (rbf)
Synthetic	945.79	12.92	–	0.12	0.04	0.53	–
CCDS	–	235.73	75.47	121.28	100.94	0.46	0.61
Foursquare	–	33.83	37.70	22.3	14.41	19.20	19.67
Meteo UK	–	40.23	–	2.12	1.67	1.57	1.66

into account the low multilinear rank of the hypothesis allows us to significantly reduce the dependence of the bound on the output dimensions from $\mathcal{O}\left(\sqrt{d_0 \cdots d_p}\right)$ to $\mathcal{O}\left(\sqrt{(R_0 \cdots R_p + \sum_i R_i d_i)(\sum_i \log d_i)}\right)$.

4.3 Experiments

In this section, we evaluate HOLRR on both synthetic and real-world data sets. Our experimental results are for tensor-structured output regression problems on which we report root mean-squared errors (RMSE) averaged across all the outputs. We compare HOLLR with the following methods: regularized least squares **RLS**, low-rank regression **LRR** described in Section 4.1.1, a multilinear approach based on tensor trace norm regularization **ADMM** (Gandy, Recht, and Yamada, 2011; Romera-Paredes et al., 2013), a nonconvex multilinear multitask learning approach **MLMT-NC** (Romera-Paredes et al., 2013), the greedy tensor approach for multivariate spatio-temporal analysis **Greedy** (Bahadori, Yu, and Liu, 2014), and the partial least squares higher order extension **HOPLS** proposed in (Zhao et al., 2013).

For experiments with kernel algorithms we use the readily available kernelized RLS and the LRR kernel extension proposed in (Mukherjee and J. Zhu, 2011). Note that ADMM, MLMT-NC and Greedy only consider a linear dependency between inputs and outputs. The greedy tensor algorithm proposed in (Bahadori, Yu, and Liu, 2014) is developed specially for spatio-temporal data and the implementation provided by the authors is restricted to the case where outputs are matrices. Although MLMLT-NC is perhaps the closest algorithm to ours, we applied it only to simulated data. This is because MLMLT-NC is computationally very expensive and becomes intractable for large data sets. Average running times for some of the experiments are reported in Table 4.1.

4.3.1 Synthetic Data

We generate both linear and nonlinear data. Linear data is drawn from the model $\mathcal{Y} = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$ where $\mathcal{W} \in \mathbb{R}^{10\times10\times10\times10}$ is a tensor of multilinear rank $(6, 4, 4, 8)$ drawn

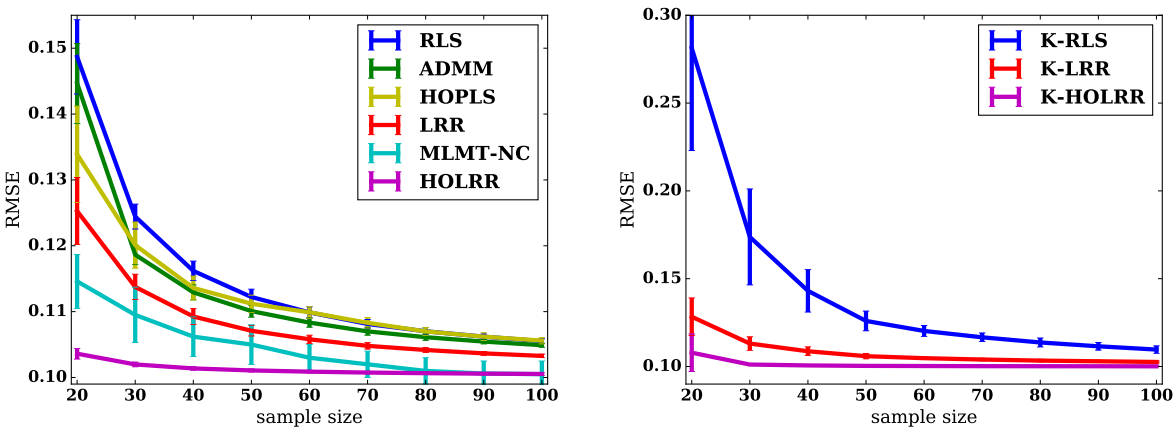


Figure 4.2: Average RMSE as a function of the training set size: (left) linear data, (right) nonlinear data. The linear data is generated using the relation $\mathcal{Y} = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$. The non linear data is generated using the relation $\mathcal{Y} = \mathcal{W} \bullet_1 (\mathbf{x} \otimes \mathbf{x}) + \mathcal{E}$. Since the non linear data is generated from a polynomial model we used an homogeneous polynomial kernel of degree 2 for all algorithms. In both cases the variance of the noise is equal to 0.1, thus the RMSE obtained on the test set by the true regression function is close to 0.1.

at random, $\mathbf{x} \in \mathbb{R}^{10}$ is drawn from $\mathcal{N}(0, \mathbf{I})$, and each component of the error tensor $\mathcal{E}$ is drawn from $\mathcal{N}(0, 0.1)$. Nonlinear data is drawn from the model $\mathcal{Y} = \mathcal{W} \bullet_1 (\mathbf{x} \otimes \mathbf{x}) + \mathcal{E}$ where $\mathcal{W} \in \mathbb{R}^{25 \times 10 \times 10 \times 10}$ is of rank $(5, 6, 4, 2)$ and $\mathbf{x} \in \mathbb{R}^5$ and $\mathcal{E}$ are generated as above. Hyper-parameters for all algorithms are selected using 3-fold cross-validation on the training data.

These experiments have been carried out for different sizes of the training data set, 20 trials have been executed for each size. The average RMSEs on a test set of size 100 for the 20 trials are reported in Figure 4.2. We see that HOLRR algorithm clearly outperforms the other methods on the linear data. MLMT-NC achieves the second best performance, it is however much more computationally expensive (see Table 4.1). On the nonlinear data LRR achieves good performances but HOLRR is still significantly more accurate, especially with small training data sets.

To see how sensitive HOLLR is with respect to the choice of the multilinear rank, we have carried out a similar experiment where we compare HOLLR performances for different values of the rank parameter. In this experiment, the rank of the tensor $\mathcal{W}$ used to generate the data is $(2, 2, 2, 2)$ while the input and output dimensions and the noise level are the same as above. The results are reported in Figure 4.3 where we see that overestimating the multilinear rank leads to a slower convergence rate (the results for cases where the multilinear rank is underestimated are very poor and would not fit in the figure).

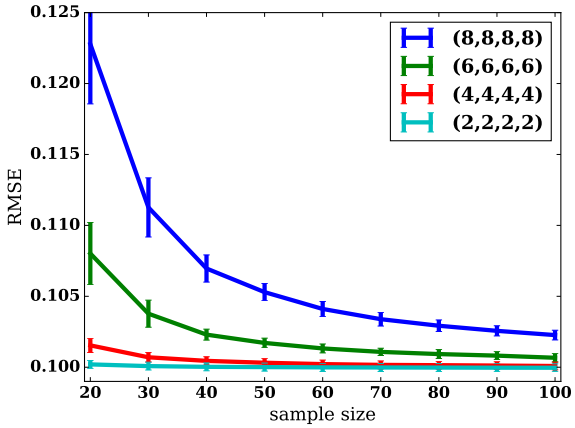


Figure 4.3: Effect of overestimating rank on synthetic linear data for the HOLRR estimator. Average RMSE as a function of the training set size. The multilinear rank of the tensor used to generate the data is $(2, 2, 2, 2)$.

4.3.2 Image Reconstruction from Noisy Measurements

To give an illustrative intuition on the differences between matrix and multilinear rank regularization we generate data from the model $\mathcal{Y} = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$ where the tensor $\mathcal{W}$ is a color image of size $m \times n$ encoded with three color channels RGB. We consider two different tasks depending on the input dimension: (i) $\mathcal{W} \in \mathbb{R}^{3 \times m \times n}$, $\mathbf{x} \in \mathbb{R}^3$ and (ii) $\mathcal{W} \in \mathbb{R}^{n \times m \times 3}$, $\mathbf{x} \in \mathbb{R}^n$. In both tasks the components of both $\mathbf{x}$ and $\mathcal{E}$ are drawn from $\mathcal{N}(0, 1)$ and the regression tensor $\mathcal{W}$ is learned from a training set of size 200.

This experiment allows us to visualize the tensors returned by the RLS, LRR and HOLRR algorithms. The results are shown in Figure 4.4 for three images: a green cross (of size 50×50), a thumbnail of a Rothko painting (44×70) and a square made of triangles (70×70), note that the first two images have a low rank structure which is not the case for the third one.

We first see that HOLRR clearly outperforms LRR on the task where the input dimension is small (task (i)). This is to be expected since the rank of the matrix $\mathbf{W}_{(1)}$ is at most 3 and LRR is unable to enforce a low-rank structure on the output modes of $\mathcal{W}$. When the rank constraint is set to 1 for LRR and $(3, 1, 1)$ for HOLRR, we clearly see that (unlike HOLRR) the LRR approach does not enforce any low-rank structure on the regression tensor along the output modes. On task (ii) the difference is more subtle, but we can see that setting a rank constraint of 2 for the LRR algorithm prevents the model from capturing the white border around the green cross and creates the vertical lines artifact in the Rothko painting. For higher values of the rank the model starts to learn the noise. The tensor returned by HOLRR with rank $(2, 2, 3)$ for the cross image and $(4, 4, 3)$ for the Rothko painting do not exhibit these behaviors and give better results on these two images. On the square image which does not have a low-rank structure both algorithms exhibit underfitting for low values of the rank parameter. Overall, we see that capturing the multilinear low-rank structure of the output data allows HOLRR to separate the noise from the true signal better than RLS and LRR.

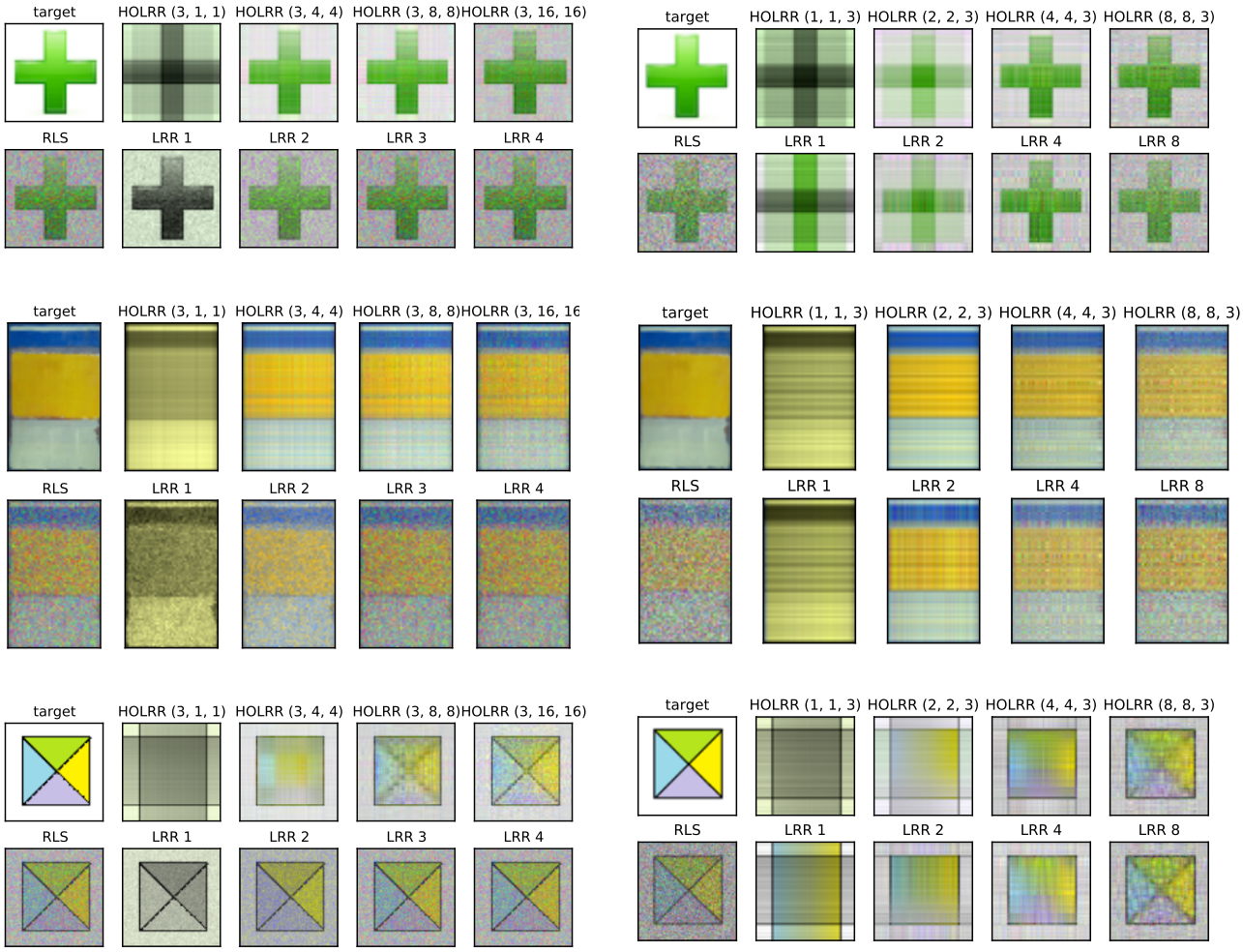


Figure 4.4: Image reconstruction from noisy measurements: $\mathcal{Y} = \mathcal{W} \bullet_1 \mathbf{x} + \mathcal{E}$ where $\mathcal{W}$ is a color image (RGB). The images are the estimators of the regression tensor $\mathcal{W}$ returned by the different algorithms. Each image is labeled with the name of the algorithm followed by the value used for the rank constraint. Images on the left correspond to task (i) and on the right to task (ii).

(left): the input dimension is the number of channels: $\mathcal{W} \in \mathbb{R}^{3 \times width \times height}$ and each input sample $\mathbf{x}^{(n)}$ is of dimension 3.

(right): the input dimension is the height of the image: $\mathcal{W} \in \mathbb{R}^{height \times width \times 3}$ and each input sample $\mathbf{x}^{(n)}$ is a vector in $\mathbb{R}^{height}$.

Table 4.2: Average RMSE (over 10 runs) of the different algorithms on spatio-temporal forecasting tasks.

Data set	ADMM	Greedy	HOPLS	HOLRR	K-HOLRR (poly)	K-HOLRR (rbf)
CCDS	0.8448	0.8325	0.8147	0.8096	0.8275	**0.7913**
Foursquare	0.1407	**0.1223**	**0.1224**	0.1227	**0.1223**	0.1226
Meteo-UK	0.6140	–	0.625	0.5971	0.6107	**0.5886**

4.3.3 Real Data

We evaluate our algorithm on a forecasting task on the following real-world data sets:

CCDS: the comprehensive climate data set is a collection of climate records of North America from (Lozano et al., 2009). The data set contains monthly observations of 17 variables such as Carbon dioxide and temperature spanning from 1990 to 2001 across 125 observation locations.

Foursquare: the Foursquare data set (Long, Jin, and Joshi, 2012) contains users' check-in records in Pittsburgh area categorized by different venue types such as Art & Entertainment, College & University, and Food. It records the number of check-ins by 121 users in each of the 15 category of venues over 1200 time intervals.

Meteo-UK: The data set is collected from the meteorological office of the UK[b]. It contains monthly measurements of 5 variables in 16 stations across the UK from 1960 to 2000.

The forecasting task consists in predicting all variables at times $t+1,\ldots,t+k$ from their values at times $t-2$, $t-1$ and t. The first two real data sets were used in (Bahadori, Yu, and Liu, 2014) with $k=1$ (i.e. outputs are matrices). We consider here the same setting for these two data sets. For the third data set we consider higher-order output tensors by setting $k=5$. The output tensors are thus of size respectively 17×125, 15×121 and $16\times 5\times 5$ for the three data sets.

For all the experiments, we use 90% of the available data for training and 10% for testing. All hyper-parameters are chosen by cross-validation. The average test RMSE over 10 runs are reported in Table 4.2 (running times are reported in Table 4.1). We see that HOLRR and K-HOLRR outperform the other methods on the CCDS data set while being orders of magnitude faster for the kernelized version (0.61s vs. 75.47s for Greedy and 235.73s for ADMM in average). On the Foursquare data set HOLRR performs as well as Greedy and HOPLS, and on the Meteo-UK data set K-HOLRR gets the best results with the rbf kernel while being much faster than ADMM (1.66s vs. 40.23s in average). It is surprising to observe that even though HOLRR does not take the tensor structure of the input into account, it performs better or as well as Greedy which takes the input structure into account and is specifically designed for the spatio-temporal forecasting task.

[b] http://www.metoffice.gov.uk/public/weather/climate-historic/

4.4 Conclusion

In this chapter, we tackled the problem of learning a regression function that maps vectors to tensors from a set of input/output samples. We showed on a simple synthetic example that enforcing low-rank structure of the regression tensor can improve performances when the ground truth model has a low multilinear rank structure. We thus proposed to tackle the regression task with tensor-structured output data by minimizing a least squares criterion subject to a multilinear rank constraint. We proposed a fast and efficient algorithm (HOLRR) to compute an approximate solution of this difficult non-convex minimization problem. We also proposed a kernelized version of HOLRR that extends our method to the non-linear setting. We provided a theoretical analysis of the approximation guarantees and statistical properties of HOLRR and a generalization bound for the class of regression function with bounded multilinear rank. On both synthetic and real-world data sets, HOLRR performed better or competitively with state of the art methods, while being computationally very efficient.

In the future, we plan to extend this work to other loss functions and to the setting where both inputs and outputs are tensors. Note that the strategy we used to derive HOLRR cannot be directly applied to these settings: indeed, HOLRR relies on the fact that the minimization problem can be reduced to a multilinear subspace identification problem which is not the case anymore when the inputs are tensors (i.e. the minimization problem cannot be decomposed into independent subspace identification problems for the input modes). Using the tools proposed in (Signoretto, De Lathauwer, and Suykens, 2013) is a promising direction to leverage the tensor structure of the input data. Considering the operator-valued kernels framework from a multilinear perspective and trying to derive an alternative non linear extension of HOLRR in this framework is also an interesting direction. We also plan to investigate how non-linearity could be introduced in low multilinear rank models using the neural networks framework: one can show that low multilinear rank regression functions such as the ones we considered in this chapter can be computed by neural networks with two hidden layers (without activation functions) where some Kronecker structure is enforced on the weight matrices of the network. Training such a network would be equivalent to tackle problem (4.2) using stochastic gradient descent, and introducing non-linear activation functions would lead to an alternative non-linear extension of the low multilinear rank model we considered in this chapter.

Appendix

4.A Proof of Theorem 18

Theorem. *Let $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}$ be independent and identically distributed (i.i.d.) random variables taking their values in $\mathbb{R}^{d_0}$ and following a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Let $\boldsymbol{\xi}^{(1)}, \cdots, \boldsymbol{\xi}^{(N)}$ be i.i.d. random variables taking their values in $\mathbb{R}^{d_1 \times \cdots \times d_p}$ such that for any $n \in [N]$ each component of $\boldsymbol{\xi}^{(n)}$ follows a normal distribution $\mathcal{N}(0, \sigma^2)$. Finally, let $\boldsymbol{\mathcal{W}}^* \in \mathbb{R}^{d_0 \times d_1 \times \cdots \times d_p}$ be a regression tensor with multilinear rank $(R_0, R_1, \cdots, R_p)$ and let $\boldsymbol{\mathcal{Y}}^{(n)} = \boldsymbol{\mathcal{W}}^* \bullet_1 \mathbf{x}^{(n)} + \boldsymbol{\xi}^{(n)}$ for all $n \in [N]$.*

Let $\boldsymbol{\mathcal{W}}_N$ be the estimator returned by HOLRR with the training sample $\{(\mathbf{x}^{(n)}, \boldsymbol{\mathcal{Y}}^{(n)})\}_{n=1}^N$ as input, with rank parameter $(R_0, R_1, \cdots, R_p)$ and regularization parameter $\gamma = 0$. Then, for any $\varepsilon > 0$ we have

$$\lim_{N \to \infty} \mathbb{P}[\|\boldsymbol{\mathcal{W}}^* - \boldsymbol{\mathcal{W}}_N\|_F > \varepsilon] = 0.$$

We first define the notion of convergence in probability. We say that a sequence of tensor-valued random variables $\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2, \cdots, \boldsymbol{\mathcal{T}}_N$ converges in probability to a tensor $\boldsymbol{\mathcal{T}}$ if for any $\varepsilon > 0$

$$\lim_{N \to \infty} \mathbb{P}[\|\boldsymbol{\mathcal{T}}_N - \boldsymbol{\mathcal{T}}\|_F > \varepsilon] = 0$$

which we will denote by $plim(\boldsymbol{\mathcal{T}}_N) = \boldsymbol{\mathcal{T}}$. If $(\mathbf{A}_N)$ and $(\mathbf{B}_N)$ are two sequences of matrix valued random variables such that $plim(\mathbf{A}_N) = \mathbf{A}$ and $plim(\mathbf{B}_N) = \mathbf{B}$ then the following properties follow directly from Slutsky's theorem (see e.g. Theorem 7.19 in DasGupta, 2011)

$$plim(\mathbf{A}_N \mathbf{B}_N) = \mathbf{A}\mathbf{B}$$
$$plim((\mathbf{A}_N)^{-1}) = \mathbf{A}^{-1}$$

(assuming that $\mathbf{A}$ is invertible).

Using the notations and hypothesis from the theorem, let $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ denote the input matrix, let $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{N \times d_1 \times \cdots \times d_p}$ denote the output tensor, and let $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{N \times d_1 \times \cdots \times d_p}$ denote the noise tensor (i.e. the tensor obtained by stacking the tensors $\boldsymbol{\xi}^{(1)}, \cdots, \boldsymbol{\xi}^{(N)}$ along the first mode). It is easy to check that $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{W}} \times_1 \mathbf{X} + \boldsymbol{\mathcal{E}}$.

First observe that it follows from the definitions of the random variables $\mathbf{x}^{(n)}$ and $\boldsymbol{\xi}^{(n)}$ and from the law of large numbers that the following hold:

- $plim(\frac{1}{N}\mathbf{X}^\top \mathbf{X}) = \lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} {\mathbf{x}^{(n)}}^\top = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{I}$,
- $plim(\frac{1}{N}\mathbf{X}^\top \mathbf{E}_{(1)}) = \lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)} {\boldsymbol{\xi}^{(n)}}^\top = \mathbb{E}[\mathbf{x}]\,\mathbb{E}[\boldsymbol{\xi}^\top] = \mathbf{0}$.

The estimator returned by HOLRR with the parameters given in the theorem satisfies

$$\boldsymbol{\mathcal{W}}_N = \boldsymbol{\mathcal{Y}} \times_1 \mathbf{U}_0 (\mathbf{U}_0^\top \mathbf{X}^\top \mathbf{X} \mathbf{U}_0)^{-1} \mathbf{U}_0^\top \mathbf{X}^\top \times_2 \mathbf{U}_1 \mathbf{U}_1^\top \times_3 \cdots \times_{p+1} \mathbf{U}_p \mathbf{U}_p^\top$$

where $\mathbf{U}_0 \in \mathbb{R}^{d_0 \times R_0}$ is the matrix having the top R_0 eigenvectors of the matrix

$$\mathbf{M} \triangleq (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{X} \tag{4.7}$$

for columns, and for any $i \geq 1$, $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ is the matrix having the top R_i eigenvectors of $\mathbf{Y}_{(i+1)}\mathbf{Y}_{(i+1)}^\top$ for columns. We will show the following facts:

(i) $plim(\boldsymbol{\mathcal{Y}} \times_1 \mathbf{U}_0(\mathbf{U}_0^\top \mathbf{X}^\top \mathbf{X}\mathbf{U}_0)^{-1}\mathbf{U}_0^\top \mathbf{X}^\top) = \boldsymbol{\mathcal{W}}^*$

(ii) $plim(\boldsymbol{\mathcal{W}}^* \times_{i+1} \mathbf{U}_i\mathbf{U}_i^\top) = \boldsymbol{\mathcal{W}}^*$ for any $i \geq 1$

which will entail the result. Indeed, assuming for sake of simplicity that $p = 2$ it is easy to check that

$$\begin{aligned} plim(\boldsymbol{\mathcal{W}}_N) &= plim(\boldsymbol{\mathcal{Y}} \times_1 \mathbf{U}_0(\mathbf{U}_0^\top \mathbf{X}^\top \mathbf{X}\mathbf{U}_0)^{-1}\mathbf{U}_0^\top \mathbf{X}^\top \times_2 \mathbf{U}_1\mathbf{U}_1^\top \times_3 \mathbf{U}_2\mathbf{U}_2^\top) \\ &= plim\Big(plim\Big(plim\Big(plim(\boldsymbol{\mathcal{Y}} \times_1 \mathbf{U}_0(\mathbf{U}_0^\top \mathbf{X}^\top \mathbf{X}\mathbf{U}_0)^{-1}\mathbf{U}_0^\top \mathbf{X}^\top) \times_2 \mathbf{U}_1\mathbf{U}_1^\top\Big) \times_3 \mathbf{U}_2\mathbf{U}_2^\top\Big). \end{aligned}$$

One of the main technical difficulty in this proof comes from the fact that for any $i \in [p]$ the limit $plim(\mathbf{U}_i)$ may not exist. Indeed, $\mathbf{U}_i$ is defined as the matrix having for columns the top R_i eigenvectors of some matrix $\mathbf{A}$, but how can we choose these eigenvectors in such a way that the matrix $\mathbf{U}_i$ converges as N grows to infinity? Think for example of the case where $\mathbf{A}$ has some eigenvalues of multiplicity greater than one. We will bypass this difficulty by reasoning on the orthogonal projection matrix $\mathbf{U}_i\mathbf{U}_i^\top$ which is itself unique (once a basis has been chosen).

We start by showing that

$$plim\left(\mathbf{U}_0\left(\mathbf{U}_0^\top \frac{1}{N}\mathbf{X}^\top \mathbf{X}\mathbf{U}_0\right)^{-1}\mathbf{U}_0\right) = plim(\mathbf{U}_0\mathbf{U}_0^\top). \tag{4.8}$$

Since $plim(\frac{1}{N}\mathbf{X}^\top\mathbf{X}) = \mathbf{I}$, for any $\varepsilon > 0$ and with probability one there exists some $d_0 \times d_0$ matrix $\boldsymbol{\Delta}$ such that

$$\|\boldsymbol{\Delta}\|_F \leq \varepsilon \;\text{ and }\; \frac{1}{N}\mathbf{X}^\top\mathbf{X} = \mathbf{I} - \boldsymbol{\Delta}.$$

Moreover, for any $\varepsilon' > 0$ we can choose ε in such a way that $\sum_{k\geq 1}(d_0\|\boldsymbol{\Delta}\|_F)^k \leq \varepsilon'$. We then have

$$\begin{aligned} \mathbf{U}_0\left(\mathbf{U}_0^\top \frac{1}{N}\mathbf{X}^\top \mathbf{X}\mathbf{U}_0\right)^{-1}\mathbf{U}_0^\top &= \mathbf{U}_0\left(\mathbf{U}_0^\top(\mathbf{I} - \boldsymbol{\Delta})\mathbf{U}_0\right)^{-1}\mathbf{U}_0^\top \\ &= \mathbf{U}_0\left(\mathbf{I} - \mathbf{U}_0^\top\boldsymbol{\Delta}\mathbf{U}_0\right)^{-1}\mathbf{U}_0^\top \\ &= \mathbf{U}_0\mathbf{U}_0^\top + \sum_{k\geq 1}(\mathbf{U}_0\mathbf{U}_0^\top\boldsymbol{\Delta}\mathbf{U}_0\mathbf{U}_0^\top)^k \end{aligned}$$

where we used the identity $(\mathbf{I} - \mathbf{A})^{-1} = \sum_{k\geq 0}\mathbf{A}^k$, and it follows that

$$\begin{aligned} \|\mathbf{U}_0\left(\mathbf{U}_0^\top \frac{1}{N}\mathbf{X}^\top \mathbf{X}\mathbf{U}_0\right)^{-1}\mathbf{U}_0^\top - \mathbf{U}_0\mathbf{U}_0^\top\|_F &= \|\sum_{k\geq 1}(\mathbf{U}_0\mathbf{U}_0^\top\boldsymbol{\Delta}\mathbf{U}_0\mathbf{U}_0^\top)^k\|_F \\ \leq \sum_{k\geq 1}\|(\mathbf{U}_0\mathbf{U}_0^\top\boldsymbol{\Delta}\mathbf{U}_0\mathbf{U}_0^\top)^k\|_F &\leq \sum_{k\geq 1}(\|\mathbf{U}_0\mathbf{U}_0^\top\|_F^2\|\boldsymbol{\Delta}\|_F)^k = \sum_{k\geq 1}(d_0\|\boldsymbol{\Delta}\|_F)^k \leq \varepsilon' \end{aligned}$$

where we used the fact that the Frobenius norm is sub-multiplicative. Since this inequality holds with probability one for any $\varepsilon' > 0$ this shows that Eq. (4.8) holds.

We can now show that fact (i) holds. First observe that

$$plim\left(\frac{1}{N}\mathbf{Y}_{(1)}^{\top}\mathbf{X}\right) = plim\left(\mathbf{W}^{*\top}_{(1)}\frac{1}{N}\mathbf{X}^{\top}\mathbf{X}\right) + plim\left(\frac{1}{N}\mathbf{E}_{(1)}^{\top}\mathbf{X}\right) = \mathbf{W}^{*\top}_{(1)}.$$

Now since the ordinary least squares estimator $\mathbf{W}_{OLS}$ is consistent and since the matrix $\mathbf{M}$ defined in Eq. (4.7) satisfies $\mathbf{M} = \mathbf{W}_{OLS}\mathbf{Y}_{(1)}^{\top}\mathbf{X}$ we have $plim(\frac{1}{N}\mathbf{M}) = \mathbf{W}^{*}_{(1)}\mathbf{W}^{*\top}_{(1)}$. Furthermore, since $\mathbf{U}_0\mathbf{U}_0^{\top}$ is the (unique) matrix of the orthogonal projection onto the space spanned by the top R_0 eigenvectors of $\mathbf{M}$ (which are equal to the top R_0 eigenvectors of $\frac{1}{N}\mathbf{M}$) it follows that $plim(\mathbf{U}_0\mathbf{U}_0^{\top})$ is the matrix of the orthogonal projection onto the space spanned by the R_0 left singular vectors of $\mathbf{W}^{*}_{(1)}$, thus $plim(\mathbf{U}_0\mathbf{U}_0^{\top}\mathbf{W}^{*}_{(1)}) = \mathbf{W}^{*}_{(1)}$. To conclude, we obtain the following equation which is equivalent to fact (i) using matricization:

$$\begin{aligned} plim(\mathbf{U}_0(\mathbf{U}_0^{\top}\mathbf{X}^{\top}\mathbf{X}\mathbf{U}_0)^{-1}\mathbf{U}_0^{\top}\mathbf{X}^{\top}\mathbf{Y}_{(1)}) &= plim\left(\mathbf{U}_0(\mathbf{U}_0^{\top}\frac{1}{N}\mathbf{X}^{\top}\mathbf{X}\mathbf{U}_0)^{-1}\mathbf{U}_0^{\top}\frac{1}{N}\mathbf{X}^{\top}\mathbf{Y}_{(1)}\right) \\ &= plim(\mathbf{U}_0\mathbf{U}_0^{\top}\mathbf{W}^{*}_{(1)}) = \mathbf{W}^{*}_{(1)}. \end{aligned}$$

We now show fact (ii): $plim(\boldsymbol{\mathcal{W}}^{*} \times_{i+1} \mathbf{U}_i\mathbf{U}_i^{\top}) = \boldsymbol{\mathcal{W}}^{*}$ for any $1 \leq i \leq p$. We will show this result for $i = 1$, the proof for other values of i is similar. Recall that $\mathbf{U}_1$ is the matrix having for columns the top R_1 eigenvectors of $\mathbf{Y}_{(2)}\mathbf{Y}_{(2)}^{\top}$. Since fact (ii) is equivalent to $plim(\mathbf{U}_1\mathbf{U}_1^{\top}\mathbf{W}^{*}_{(2)}) = \mathbf{W}^{*}_{(2)}$ we just need to show that $plim(\mathbf{U}_1\mathbf{U}_1^{\top})$ is the matrix of the orthogonal projection onto the space spanned by the R_1 left singular vectors of $\mathbf{W}^{*}_{(2)}$.

First observe that $\mathbf{Y}_{(2)} = \mathbf{W}^{*}_{(2)}(\mathbf{X}^{\top} \otimes \mathbf{I}) + \mathbf{E}_{(2)}$, hence $\mathbf{Y}_{(2)}\mathbf{Y}_{(2)}^{\top}$ is equal to

$$\mathbf{W}^{*}_{(2)}((\mathbf{X}^{\top}\mathbf{X}) \otimes \mathbf{I})\mathbf{W}^{*\top}_{(2)} + \mathbf{E}_{(2)}\mathbf{E}_{(2)}^{\top} + \mathbf{W}^{*}_{(2)}(\mathbf{X}^{\top} \otimes \mathbf{I})\mathbf{E}_{(2)}^{\top} + \mathbf{E}_{(2)}(\mathbf{X}^{\top} \otimes \mathbf{I})\mathbf{W}^{*\top}_{(2)}.$$

Since $(\mathbf{X}^{\top} \otimes \mathbf{I})\mathbf{E}_{(2)}^{\top}$ is obtained by permuting the components of the matrix $\mathbf{X}^{\top}\mathbf{E}_{(1)}$ we have

$$plim\left(\frac{1}{N}\mathbf{W}^{*}_{(2)}(\mathbf{X}^{\top} \otimes \mathbf{I})\mathbf{E}_{(2)}^{\top}\right) = plim\left(\frac{1}{N}\mathbf{E}_{(2)}(\mathbf{X}^{\top} \otimes \mathbf{I})\mathbf{W}^{*\top}_{(2)}\right) = \mathbf{0}.$$

Furthermore, since

$$(\mathbf{E}_{(2)}\mathbf{E}_{(2)}^{\top})_{ij} = \sum_{n=1}^{N}\sum_{k_2=1}^{d_2}\cdots\sum_{k_p=1}^{d_p}\boldsymbol{\xi}^{(n)}_{i,k_2,\cdots,k_p}\boldsymbol{\xi}^{(n)}_{j,k_2,\cdots,k_p}$$

we have $\lim_{N\to\infty}\frac{1}{N}(\mathbf{E}_{(2)}\mathbf{E}_{(2)}^{\top})_{ij} = d_2\cdots d_p\,\mathbb{E}[\xi_i\xi_j]$ where each $\xi_i \sim \mathcal{N}(0,\sigma^2)$, and it follows that

$$plim\left(\frac{1}{N}\mathbf{E}_{(2)}\mathbf{E}_{(2)}^{\top}\right) = d_2\cdots d_p\sigma^2\mathbf{I}_{d_1}.$$

We then have

$$plim\left(\frac{1}{N}\mathbf{Y}_{(2)}\mathbf{Y}_{(2)}^{\top}\right) = \mathbf{W}^{*}_{(2)}\mathbf{W}^{*\top}_{(2)} + d_2\cdots d_p\sigma^2\mathbf{I}_{d_1}.$$

Observing that the matrix $\mathbf{A} + \alpha\mathbf{I}$ has the same eigenvectors as $\mathbf{A}$ for any matrix $\mathbf{A}$ and scalar α, and since $\mathbf{U}_1\mathbf{U}_1^{\top}$ is the matrix of the orthogonal projection onto the space spanned by the top R_1 left singular vectors of $\mathbf{Y}_{(2)}$ we can conclude that $plim(\mathbf{U}_1\mathbf{U}_1^{\top})$ is

the matrix of the orthogonal projection onto the space spanned by the R_1 left singular vectors of $\mathbf{W}^*{}_{(2)}$.

4.B Proof of Theorem 19

We start by bounding the pseudo-dimension of the class of real-valued functions with domain $\mathbb{R}^{d_0} \times [d_1] \times \cdots \times [d_p]$

$$\tilde{\mathcal{F}} = \left\{(\mathbf{x}, i_1, \cdots, i_p) \mapsto (\boldsymbol{\mathcal{W}} \bullet_1 \mathbf{x})_{i_1,\cdots,i_p} \ : \ \text{rank}_{ml}(\boldsymbol{\mathcal{W}}) = (R_0, \cdots, R_p)\right\}.$$

We first recall the definition of the pseudo-dimension of a class of real-valued functions.

Definition 6. *A class $\mathcal{F}$ of real-valued functions pseudo-shatters the points $x_1, \cdots, x_m$ with thresholds $t_1, \cdots, t_m$ if for every binary labeling of the points $(s_1, \cdots, s_m) \in \{-, +\}^m$ there exists $f \in \mathcal{F}$ s.t. $f(x_i) < t_i$ iff $s_i = -$. The pseudo-dimension of a class F is the supremum over m for which there exist m points that are pseudo-shattered by $\mathcal{F}$ (with some thresholds).*

We will say that a set of polynomials $p_1, p_2, \cdots, p_k$ has at least m sign patterns if there exist $x_1, \cdots, x_m$ such that such that the sign vectors $\mathbf{v}_i = [sign(p_1(x_i)), \cdots, sign(p_k(x_i))]^\top$ are pairwise distinct. Following (Warren, 1968), the following theorem bounds the number of sign patterns for a set of polynomials.

Theorem. *(Srebro, 2004, Theorem 34, 35) The number of sign patterns of r polynomials, each of degree at most d, over q variables is at most $\left(\frac{4edr}{q}\right)^q$ for all $r > q > 2$.*

The following lemma gives an upper bound on the pseudo-dimension of $\tilde{\mathcal{F}}$ using the previous theorem.

Lemma 6. *The pseudo-dimension of the real-valued function class $\tilde{\mathcal{F}}$ is upper bounded by $(R_0R_1 \cdots R_p + \sum_{i=0}^{p} R_i d_i) \log\left(\frac{4e(p+2)d_0d_1\cdots d_p}{d_0+d_1+\cdots+d_p}\right)$.*

Proof. It is well known that the pseudo-dimension of a vector space of real-valued functions is equal to its dimension (Mohri, Rostamizadeh, and Talwalkar, 2012, Theorem 10.5). Since $\tilde{\mathcal{F}}$ is a (non-linear) subspace of the $d_0d_1 \cdots d_p$-dimensional vector space

$$\left\{(\mathbf{x}, i_1, \cdots, i_p) \mapsto (\boldsymbol{\mathcal{W}} \bullet_1 \mathbf{x})_{i_1,\cdots,i_p} \ : \ \boldsymbol{\mathcal{W}} \in \mathbb{R}^{d_0 \times \cdots \times d_p}\right\}$$

of real-valued functions with domain $\mathbb{R}^{d_0} \times [d_1] \times \cdots \times [d_p]$, the pseudo-dimension of $\tilde{\mathcal{F}}$ is bounded by $d_0d_1 \cdots d_p$.

Now, let $m \leq d_0 \cdots d_p$ and let $\{(\mathbf{x}^k, i_1^k, \cdots, i_p^k)\}_{k=1}^m$ be a set of points that are pseudo-shattered by $\tilde{\mathcal{F}}$ with thresholds $t_1, \cdots, t_m \in \mathbb{R}$. Then for each sign pattern $(s_1, \cdots, s_m) \in \{-, +\}^m$, there exists $\tilde{f} \in \tilde{\mathcal{F}}$ such that $sign(\tilde{f}(\mathbf{x}^k, i_1^k, \cdots, i_p^k) - t_k) = s_k$. Any function $\tilde{f} \in \tilde{\mathcal{F}}$ can be written as

$$(\mathbf{x}, j_1, \cdots, j_p) \mapsto \left(\boldsymbol{\mathcal{G}} \times_1 \mathbf{x}^\top \mathbf{U}_0 \times_2 \mathbf{U}_1 \cdots \times_{p+1} \mathbf{U}_p\right)_{j_1,\cdots,j_p}$$

for some $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_0 \times \cdots \times R_p}$, $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ for $0 \leq i \leq p$. Thus, considering the entries of $\boldsymbol{\mathcal{G}}, \mathbf{U}_0, \cdots, \mathbf{U}_p$ as variables, the set $\{\tilde{f}(\mathbf{x}^k, i_1^k, \cdots, i_p^k) - t_k\}_{k=1}^m$ can be seen as a set of

m polynomials of degree at most $p+2$ over these $D = R_0 \cdots R_p + \sum_{i=0}^{p} d_i R_i$ variables. It then follows from the previous theorem that $2^m \leq \left(\frac{4e(p+2)m}{D}\right)^D$. The result follows using $m \leq d_0 \cdots d_p$ and $D \geq \sum_{i=0}^{p} d_i$.

□

Once the pseudo-dimension of the function class $\tilde{\mathcal{F}}$ is bounded, one can invoke standard error generalization bounds in terms of the pseudo-dimension (Mohri, Rostamizadeh, and Talwalkar, 2012, Theorem 10.6) to obtain the following theorem that gives an upper bound on the excess risk for the class of function

$$\mathcal{F} = \{\mathbf{x} \mapsto \boldsymbol{\mathcal{W}} \bullet_1 \mathbf{x} \ : \ \text{rank}_{ml}(\boldsymbol{\mathcal{W}}) = (R_0, \cdots, R_p)\}.$$

Theorem. *Let $\mathcal{L} : \mathbb{R}^{d_1 \times \cdots \times d_p} \to \mathbb{R}$ be a loss function satisfying*

$$\mathcal{L}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) = \frac{1}{d_1 \cdots d_p} \sum_{i_1, \cdots, i_p} \ell(\boldsymbol{\mathcal{A}}_{i_1, \cdots, i_p}, \boldsymbol{\mathcal{B}}_{i_1, \cdots, i_p})$$

for some loss function $\ell : \mathbb{R} \to \mathbb{R}^+$ bounded by M. Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample of size N, the following inequality holds for all $h \in \mathcal{F}$:

$$R(h) \leq \hat{R}(h) + M\sqrt{\frac{2D \log\left(\frac{4e(p+2)d_0 d_1 \cdots d_p}{d_0 + d_1 + \cdots + d_p}\right) \log N}{N}} + M\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2N}}$$

where $D = R_0 R_1 \cdots R_p + \sum_{i=0}^{p} R_i d_i$.

Proof. For any $h : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times \cdots \times d_p}$ we define $\tilde{h} : \mathbb{R}^{d_0} \times [d_1] \times \cdots \times [d_p] \to \mathbb{R}$ by $\tilde{h}(\mathbf{x}, i_1, \cdots, i_p) = h(\mathbf{x})_{i_1 \cdots i_p}$. Let $\mathcal{D}$ denote the distribution of the input data. We have

$$\begin{aligned} R(h) &= \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}}[\mathcal{L}(f(\mathbf{x}), h(\mathbf{x}))] = \frac{1}{d_1 \cdots d_p} \sum_{i_1, \cdots, i_p} \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}}[\ell(f(\mathbf{x})_{i_1 \cdots i_p}, h(\mathbf{x})_{i_1 \cdots i_p})] \\ &= \mathop{\mathbb{E}}_{\substack{\mathbf{x} \sim \mathcal{D} \\ i_k \sim \mathcal{U}(d_k), k \in [p]}} [\ell(\tilde{f}(\mathbf{x}, i_1, \cdots, i_p), \tilde{h}(\mathbf{x}, i_1, \cdots, i_p))] \end{aligned}$$

where $\mathcal{U}(k)$ denotes the discrete uniform distribution on $[k]$ for any integer $k \geq 1$. It follows that $R(h) = R(\tilde{h})$. Similarly, one can show that $\hat{R}(h) = \hat{R}(\tilde{h})$. The result then directly follows using Theorem 10.6 in (Mohri, Rostamizadeh, and Talwalkar, 2012) (see below) to bound $R(\tilde{h}) - \hat{R}(\tilde{h})$. □

Theorem (Theorem 10.6 in Mohri, Rostamizadeh, and Talwalkar, 2012)**.** *Let H be a family of real-valued functions and let $G = \{x \mapsto L(h(x), f(x)) : h \in H\}$ be the family of loss functions associated to H. Assume that the pseudo-dimension of G is bounded by d and that the loss function L is bounded by M. Then, for any $\delta > 0$, with probability at least δ over the choice of a sample of size m, the following inequality holds for all $h \in H$:*

$$R(h) \leq \hat{R}(h) + M\sqrt{\frac{2d \log\left(\frac{em}{d}\right)}{m}} + M\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}}.$$

5 Learning Algebraic Mixtures with the Tensor Method of Moments

Contents

5.1 Introduction

In this last chapter, we stay in the world of machine learning but we leave the supervised learning setting and consider an *unsupervised learning* task. In an unsupervised setting, the learner has to reveal hidden structure in a sample of unlabeled data. One example of an unsupervised learning task is *clustering*, where from a sample of N unlabeled examples the learner has to identify k groups of examples such that examples in a same group are more similar than examples in different groups. Another example is the *density estimation* task, where the learner has to identify the underlying distribution used to generate the training data. More precisely, the learner is given a sample $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^d$ independently and identically drawn from a distribution $\mathcal{D}$, and tries to infer the distribution $\mathcal{D}$ from this data. In this chapter, we consider a density estimation task for *algebraic mixtures*, which are a generalization of *mixture models*.

Mixture models, such as Gaussian mixtures or hidden Markov models (HMM), are widely used in statistics and machine learning (McLachlan and Peel, 2004). Roughly

speaking, a mixture model assumes that the distribution of interest $\mathcal{D}$ is the mixture of several other distributions $\mathcal{D}_1, \cdots, \mathcal{D}_k$, thus one can think of a sample $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ drawn from $\mathcal{D}$ as the reunion of k samples (not necessarily of the same sizes) drawn from the k distributions of the mixture. Let us illustrate this model with a mixture of univariate Gaussians. In this model, the probability density function (PDF) of the mixture is given by $f = \sum_{i=1}^{k} p_i \mathcal{N}(\mu_i, \sigma_i^2)$ where $0 \leq p_i \leq 1$, $\sum_{i=1}^{k} p_i = 1$ and $\mathcal{N}(\mu_i, \sigma_i^2)$ denotes the PDF of a Gaussian distribution with mean μ_i and variance σ_i^2. The coefficients of the mixture p_i are called *weights*. A real-world example where this model naturally appears is the distribution of height within a population of men and women: one can assume that the height of men is distributed from an univariate Gaussian with mean μ_1 while the height of women is distributed from a Gaussian with mean $\mu_2 < \mu_1$. The heights of the whole population would then be distributed from a mixture of these two univariate Gaussians, and the weights p_1 and p_2 of the mixture would be defined by the proportion of men/women in the population.

Mixture models can be interpreted as *latent variable models* in the following way: given a mixture $f = p_1 f_1 + \cdots + p_k f_k$ (where $f, f_1, \cdots, f_k$ are PDFs), sampling an example from the distribution $\mathcal{D}_f$ (the probability distribution associated with the PDF f) can be interpreted as first randomly choosing a component $i \in [k]$ according to the probabilities $p_1, \cdots, p_k$, and then drawing a sample from the distribution $\mathcal{D}_{f_i}$. We can thus introduce a discrete *latent variable* h taking its values in $[k]$ and following the distribution defined by $\mathbb{P}[h = i] = p_i$. Then, the conditional PDF satisfies $f(x|h = i) = f_i$.

Given an affine combination of PDFs $f = w_1 f_1 + \cdots + w_k f_k$ (i.e. the weights w_i sum to one but may be negative), the function f may remain positive even if some of the weights are negative. In that case, the function f is a valid PDF and thus defines a probability distribution $\mathcal{D}_f$. Such mixtures with negative weights have been studied in a few specific papers (R. Jiang, Zuo, and H.-X. Li, 1999; B. Zhang and C. Zhang, 2005; Müller et al., 2012). We call such distributions *algebraic mixtures* and they will be the subject of this chapter. We will show that algebraic mixtures naturally appear in a fundamental relation between probabilistic and weighted automata. This uncanny relation was the starting point of our interest for this unusual model of distribution.

While the components of an algebraic mixture with positive weights can still be interpreted as in a classical mixture, the components with negative weights can be thought of as repulsive components: it will be less likely to observe data in the regions of the space where the negative components have a high density. We will give more intuition on this in Section 5.2.2 by introducing the model of *distorted distributions* and showing that this model is equivalent to algebraic mixtures. Distorted distributions are distributions defined by a specific sampling process: a sample is drawn by first drawing a point x from an underlying *true distribution* f, and then rejecting this point with probability $\phi(x)$ where ϕ is the *mask component* of the distorted distribution.. Thus a sample drawn from a distorted distribution can be seen as a sample drawn from f where some data is missing *not at random* (since the distribution of the missing data is not uniform but depends on the mask component ϕ). We will see that such distributions could be relevant to model particular learning settings such as *section bias* (Heckman, 1977; Berk, 1983) and *inference from presence-only data* (Phillips et al., 2009; Pearce and Boyce, 2006; Keating and Cherry, 2004).

In contrast with classical mixture models, algebraic mixtures cannot be seen as latent

variable models, which prevents us from applying the Expectation-Maximization algorithm (Bailey and Elkan, 1994) that is commonly used to learn classical mixture models. However, the tensor method of moments proposed in (Anandkumar et al., 2014) to learn mixture models does not fundamentally rely on the fact that mixture models are latent variable models. In this paper, the authors show that the parameters of a number of latent variable models, including spherical Gaussian mixture models and HMMs, can easily be estimated from tensor decomposition of low-order moments that can be estimated on the training data. We will show in this chapter that this method can be extended to algebraic mixtures when the parameters of the distribution can be expressed in terms of the low order moments, and in particular that algebraic mixtures of spherical Gaussian distributions can be learned using this extension. The tensor method of moments can be seen as an algebraic learning method since it boils down to using tensor decomposition techniques to solve a system of polynomial equations that relates the moments and the parameters of the model. The learnability of distributions whose moments are polynomials of the parameters has been studied using tools from algebraic geometry in (Belkin and Sinha, 2010), and the tensor method of moments has been extended to the non parametric setting for learning multi-view latent variable models using kernel mean embeddings in (Song et al., 2014).

Summary of the contributions. In this chapter, we consider the problem of estimating the density function of an algebraic mixture from learning data. We *introduce the notion of algebraic mixture models* that generalizes the classical mixture models to the case where some of the weights are negative (Section 5.2). We show that algebraic mixtures can modelize a specific missing data scenario by introducing the notion of *distorted distributions* (Section 5.2.2). We also show how algebraic mixtures *naturally appear in a fundamental relation between weighted and probabilistic automata* (Section 5.2.3). We then propose an *extension of the tensor method of moments* proposed in (Anandkumar et al., 2014) to algebraic mixtures (Section 5.3). Focussing on Gaussian mixtures, we show how the low-order moments of an algebraic mixture of spherical Gaussians are related to the parameters of the mixture (Section 5.3.1) and we propose a *learning algorithm for algebraic mixtures of spherical Gaussians*. We then assess the performance of this algorithm in a simulation study (Section 5.4).

The works presented in this chapter have been presented in the french machine learning conference CAp in July 2014 (45mn talk) where we received the best student paper award, and at the *ICML Workshop on Method of Moments and Spectral Learning* in June 2014 (poster).

5.1.1 Method of Moments and Tensor Decomposition

We first recall the classical method of moments used for density estimation of parametric distributions. We then present a special case of this method proposed in (Anandkumar et al., 2014) that relies on tensor decomposition techniques.

Method of moments. The method of moments is a general density estimation method used to fit the parameters of a distribution from sample data. Let $\{\mathcal{D}_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^k\}$ be a parameterized family of distributions characterized by a set of parameters $\theta_1, \theta_2, \cdots \theta_k$

— think for example of the family of univariate Gaussian distributions that are parameterized by their mean μ and their variance σ^2. The method of moments consists in deriving equations that relate the parameters of the distribution to the first k moments:

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\boldsymbol{\theta}}}[x] = g_1(\theta_1,\cdots,\theta_k),\ \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\boldsymbol{\theta}}}[x^2] = g_2(\theta_1,\cdots,\theta_k),\cdots,\ \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\boldsymbol{\theta}}}[x^k] = g_k(\theta_1,\cdots,\theta_k).$$

For univariate Gaussians we have $\mathbb{E}[x] = \mu$ and $\mathbb{E}[x^2] = \mu^2 + \sigma^2$. Given a sample $\{x_n\}_{n=1}^N$ drawn from the distribution $\mathcal{D}_{\boldsymbol{\theta}}$, the population moments are defined by $\hat{M}_i = \frac{1}{N}\sum_{n=1}^N x_n^i$ for $i \in [k]$, and the method of moments estimators $\hat{\theta}_1,\cdots,\hat{\theta}_k$ are given by the solution (if it exists) of the system of equations $\hat{M}_i = g_i(\hat{\theta}_1,\cdots,\hat{\theta}_k)$ for $i \in [k]$. It is easy to check that for univariate Gaussians, the method of moments leads to the same estimators as the maximum likelihood method.

Method of moments and tensor decomposition. Recall that $(\mathbb{F}^d)^{\otimes p}$ denotes the *p-th order tensor product* of the vector space $\mathbb{F}^d$, and that $\mathbf{v}^{\circ p} = \mathbf{v}\circ\cdots\circ\mathbf{v}$ denotes the pth tensor power of the vector $\mathbf{v}\in\mathbb{F}^d$. In particular, $\mathbf{v}\circ\mathbf{v}$ can be identified with the matrix $\mathbf{v}\mathbf{v}^\top$.

For a multivariate random variable $\mathbf{x}$ taking its values in $\mathbb{R}^d$, its moment of order i is defined as the tensor $\mathbb{E}[\mathbf{x}^{\circ i}] \in (\mathbb{R}^d)^{\otimes i}$. As an illustration, the moments of order 1 and 2 of a random variable $\mathbf{x}$ following the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ are given by

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \ \text{ and } \ \mathbb{E}[\mathbf{x}\circ\mathbf{x}] = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \boldsymbol{\mu}\boldsymbol{\mu}^\top + \boldsymbol{\Sigma}$$

and once again the method of moments leads to the maximum likelihood estimators (note that this is not true in general).

It has been shown in (Anandkumar et al., 2014) that for a wide class of latent variable models (including exchangeable single topics models, mixtures of spherical Gaussians and hidden Markov models), the parameters of the model can be expressed as a function of low-order (tensor) moments. Let us consider a latent variable model where the random variable of interest depends on a discrete random variable h taking its values in $[k]$ in the following way: an observation $\mathbf{x}$ in this model is generated by first drawing a value $i\in[k]$ according to the distribution of h and then drawing $\mathbf{x}$ from the parameterized distribution $\mathcal{D}_{\boldsymbol{\mu}_i}$. Then, Anandkumar et al. (2014) show that the low-order moments are related to the tensors

$$\mathbf{M}_2 = \sum_{i=1}^k p_i\boldsymbol{\mu}_i\circ\boldsymbol{\mu}_i \ \text{ and } \ \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^k p_i\boldsymbol{\mu}_i\circ\boldsymbol{\mu}_i\circ\boldsymbol{\mu}_i \tag{5.1}$$

where $p_i = \mathbb{P}[h=i]$ for $i\in[k]$. The problem of estimating the parameters $\boldsymbol{\mu}_i$ and p_i can thus be reduced to solving this system of polynomial equations, which can be done using the power tensor method (see below).

In this chapter, we will focus on the *spherical Gaussian mixture model*[a]. This model is specified as follows: let $k\geq 1$ be the number of components, and for $i\in[k]$, let $p_i>0$ be the probability of choosing the component $\mathcal{N}(\boldsymbol{\mu}_i,\sigma_i^2\mathbf{I})$ where $\boldsymbol{\mu_i}\in\mathbb{R}^d$, $\sigma_i^2>0$. The PDF of the random vector $\mathbf{x}\in\mathbb{R}^d$ is given by $f = \sum_{i=1}^k p_i\,\mathcal{N}(\boldsymbol{\mu}_i,\sigma_i^2\mathbf{I})$.

[a] To the best of our knowledge, the tensor method of moments can not be applied to mixtures of Gaussians that are not spherical.

Assuming that the component mean vectors $\boldsymbol{\mu}_i$ are linearly independent, the following relation between the low-order moments and the matrix and tensor $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ defined in Eq. (5.1) is proved in (Hsu and Kakade, 2013).

Theorem 20. *The average variance* $\bar{\sigma}^2 = \sum_{i=1}^k p_i \sigma_i^2$ *is the smallest eigenvalue of the covariance matrix* $\mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^\top]$. *Let* $\mathbf{v}$ *be any unit-norm eigenvector corresponding to* $\bar{\sigma}^2$ *and let*

$$\mathbf{m}_1 = \mathbb{E}[\mathbf{x}(\mathbf{v}^\top(\mathbf{x} - \mathbb{E}[\mathbf{x}]))^2], \qquad \mathbf{M}_2 = \mathbb{E}[\mathbf{x} \circ \mathbf{x}] - \bar{\sigma}^2 \mathbf{I}, \quad \textit{and}$$
$$\boldsymbol{\mathcal{M}}_3 = \mathbb{E}[\mathbf{x} \circ \mathbf{x} \circ \mathbf{x}] - \sum_{i=1}^d [\mathbf{m}_1 \circ \mathbf{e}_i \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{m}_1 \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{e}_i \circ \mathbf{m}_1]$$

where $\mathbf{e}_1, \cdots, \mathbf{e}_d$ *is the coordinate basis of* $\mathbb{R}^d$. *Then,*

$$\mathbf{m}_1 = \sum_{i=1}^k p_i \sigma_i^2 \boldsymbol{\mu}_i, \qquad \mathbf{M}_2 = \sum_{i=1}^k p_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \ , \ \textit{and} \qquad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^k p_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i.$$

Observe that if the weights p_i and means $\boldsymbol{\mu}_i$ are known, one can recover the variance parameters σ_i from the observable $\mathbf{m}_1$ by solving a linear system *on the condition* that the means $\boldsymbol{\mu}_i$ are linearly independent (in particular if $k > n$ this linear system is underdetermined).

Tensor power method. We now present the method proposed in (Anandkumar et al., 2014) to recover the unknowns p_i and $\boldsymbol{\mu}_i$ for $i \in [k]$ from the system (5.1). Recall that the CP *rank* of a tensor $\boldsymbol{\mathcal{T}} \in (\mathbb{F}^d)^{\otimes p}$ is the smallest integer R such that $\boldsymbol{\mathcal{T}}$ can be written as

$$\boldsymbol{\mathcal{T}} = \sum_{i=1}^R \lambda_i \mathbf{v}_i^{(1)} \circ \cdots \circ \mathbf{v}_i^{(p)}$$

with $\lambda_i \in \mathbb{F}$ and unit-norm vectors $\mathbf{v}_i^{(1)}, \cdots, \mathbf{v}_i^{(p)} \in \mathbb{F}^n$. The *symmetric rank* of a symmetric tensor $\boldsymbol{\mathcal{T}}$ is the smallest integer R such that $\boldsymbol{\mathcal{T}}$ can be written as $\boldsymbol{\mathcal{T}} = \sum_{i=1}^R \lambda_i \mathbf{v}_i^{\circ p}$ with $\lambda_i \in \mathbb{F}$ and unit-norm vectors $\mathbf{v}_i \in \mathbb{F}^d$ for $i \in [R]$. It is known that computing the rank of a tensor is NP-hard and it is conjectured that computing the symmetric rank is also NP-hard (Hillar and Lim, 2013). However, if a real-valued third-order tensor $\boldsymbol{\mathcal{T}}$ has a *symmetric orthonormal decomposition*, i.e. $\boldsymbol{\mathcal{T}} = \sum_{i=1}^R \lambda_i \mathbf{v}_i^{\circ 3}$ with $\lambda_i \in \mathbb{R}$, $\mathbf{v}_i \in \mathbb{R}^d$ and $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}$ for all $i, j \in [k]$, it has been shown in (Anandkumar et al., 2014) that this decomposition can be recovered using a *tensor power method* (see below). Moreover, they show that any *symmetric independent decomposition*

$$\boldsymbol{\mathcal{T}} = \sum_{i=1}^k \lambda_i \mathbf{v}_i \circ \mathbf{v}_i \circ \mathbf{v}_i$$

where the $\mathbf{v}_i$'s are linearly independent but not necessarily orthonormal, can be recovered if we have access to the second order tensor $\mathbf{M} = \sum_{i=1}^k \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$ (see Theorem 21 below). Note that the problem of finding a symmetric orthogonal decomposition generalizes the eigendecomposition problem to tensors of higher order: if $\mathbf{M} = \sum_{i=1}^k \lambda_i \mathbf{v}_i \circ \mathbf{v}_i$

and the $\mathbf{v}_i$'s are orthonormal, then the couples $(\lambda_i, \mathbf{v}_i)$ are eigenvalues/vectors of the matrix $\mathbf{M}$.

Theorem 21 (Anandkumar et al., 2014). *Let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be linearly independent vectors of $\mathbb{R}^d$, $\lambda_1, \ldots, \lambda_k$ be positive scalars, and let*

$$\mathbf{M}_2 = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i \circ \mathbf{v}_i \quad \text{and} \quad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^{k} \lambda_i \mathbf{v}_i \circ \mathbf{v}_i \circ \mathbf{v}_i.$$

Finally, let $\mathbf{W} \in \mathbb{R}^{d \times k}$ be a matrix such that $\mathbf{W}^\top \mathbf{M}_2 \mathbf{W} = \mathbf{I}_k$, and let $\boldsymbol{\nu}_i = \sqrt{\lambda_i} \mathbf{W}^\top \mathbf{v}_i$ and $\gamma_i = \lambda_i^{-1/2}$ for $i \in [k]$. Then,

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W}) = \sum_{i=1}^{k} \gamma_i \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i$$

is an orthonormal decomposition from which the parameters λ_i and $\mathbf{v}_i$ can be recovered.

Observe that the fact that the decomposition of the tensor $\boldsymbol{\mathcal{T}}$ given in the previous theorem is orthonormal follows from the fact that

$$\sum_{i=1}^{k} \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top = \mathbf{W}^\top \mathbf{M}_2 \mathbf{W} = \mathbf{I}_k.$$

One method that can be used to recover the parameters of the orthonormal decomposition $\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} \gamma_i \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i$ is the *tensor power method*. Similarly to the classical power method used to recover the dominant eigenvector of a matrix (Mises and Pollaczek-Geiringer, 1929), the tensor power method relies on iterating the map

$$\boldsymbol{\theta} \mapsto \frac{\boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\theta}, \boldsymbol{\theta})}{\|\boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\theta}, \boldsymbol{\theta})\|}$$

until convergence to a fixed point. It is shown in (Anandkumar et al., 2014) that the power method will converge to one of the vectors $\boldsymbol{\nu}_i$ from which the corresponding weight γ_i can easily be recovered. Indeed, once $\boldsymbol{\nu}_i$ is known it is easy to check that $\boldsymbol{\mathcal{T}}(\boldsymbol{\nu}_i, \boldsymbol{\nu}_i, \boldsymbol{\nu}_i) = \gamma_i$. The process is repeated using the standard deflation technique: apply the tensor power method on the residual $\boldsymbol{\mathcal{T}}' = \boldsymbol{\mathcal{T}} - \gamma_i \boldsymbol{\nu}_i^{\circ 3}$ to recover a second couple $(\gamma_j, \boldsymbol{\nu}_j)$, and so on.

The previous results induce a learning scheme for the spherical Gaussian mixture model: (i) estimate the tensors $\mathbf{m}_1$, $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ defined in Theorem 20 from the learning data; (ii) compute an orthonormal decomposition as in Theorem 21; (iii) use the tensor power method to compute the mean vectors $\boldsymbol{\mu}_i$ and the probabilities p_i and (iv) use $\mathbf{m}_1$ to recover the variance parameters σ_i^2 (by solving a linear system).

5.2 Algebraic Mixtures: Definition and Properties

We now describe the object of interest of this chapter: algebraic mixtures. Given a finite set of probability density functions (PDF) $f_1 \ldots, f_k$, it may happen that $w_1 f_1 + \ldots + w_k f_k$ defines a PDF even if some weights w_i are negative. For example, if f and g are two PDFs

satisfying $g \leq cf$ for some $c > 1$, then $\alpha f - (\alpha - 1)g$ is a PDF for any mixture parameter α satisfying $0 \leq \alpha - 1 \leq (c-1)^{-1}$, which follows from the fact that $g/f \leq c \leq \alpha/(\alpha - 1)$. We call *algebraic* such generalized mixtures.

It can easily be shown, by grouping the positive and negative weights respectively, that any algebraic mixture can be written as an algebraic mixture of at most two *non negative mixtures*, i.e. mixtures with only non negative weights:

$$\sum_{i=1}^{k} w_i f_i = w_P \sum_{i:w_i \geq 0} \frac{w_i}{w_P} f_i - w_N \sum_{i:w_i < 0} \frac{-w_i}{w_N} f_i \text{ where } w_P = \sum_{i:w_i \geq 0} w_i \text{ and } w_N = -\sum_{i:w_i < 0} w_i.$$

Thus, from a theoretical perspective, it will often be enough to consider algebraic mixtures of the form $\alpha f - (\alpha - 1)g$ for some mixing coefficient $\alpha > 1$. If f, g and α are known, and if we have access to a random generator for the distribution $\mathcal{D}_f$, then Algorithm 6 simulates the distribution $\mathcal{D}_{\alpha f - (\alpha-1)g}$ by rejection sampling.

Algorithm 6 Simulating an algebraic mixture

1: **repeat**
2: draw x according to $\mathcal{D}_f$
3: draw u uniformly in $[0, 1]$
4: **until** $u\alpha f(x) \geq (\alpha - 1)g(x)$
5: **return** x

5.2.1 Algebraic Mixtures of Gaussians

We illustrate the algebraic mixture model with a simple example of algebraic mixture of two Gaussian distributions. An example of such a mixture is shown in Figure 5.1. Let f and g be the PDFs of the two d-dimensional Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f)$ and $\mathcal{N}(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$. We show below on what conditions the function $\alpha f - (\alpha - 1)g$ defines a valid probability density function, i.e. an algebraic mixture of the two distributions $\mathcal{D}_f$ and $\mathcal{D}_g$.

For any real number $\alpha > 0$, we have

$$\alpha f(\mathbf{x}) - (\alpha - 1)g(\mathbf{x}) \geq 0 \tag{5.2}$$

if and only if

$$\exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_f)^\top \boldsymbol{\Sigma}_f^{-1}(\mathbf{x} - \boldsymbol{\mu}_f) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_g)^\top \boldsymbol{\Sigma}_g^{-1}(\mathbf{x} - \boldsymbol{\mu}_g)\right\} \geq \sqrt{\frac{|\boldsymbol{\Sigma}_f|}{|\boldsymbol{\Sigma}_g|}}(1 - 1/\alpha).$$

If $\alpha \leq 1$ then Eq. (5.2) is simply the PDF of a standard Gaussian mixture. If $\alpha > 1$ then (5.2) holds for any $\mathbf{x} \in \mathbb{R}^d$ if and only if

$$-(\mathbf{x} - \boldsymbol{\mu}_f)^\top \boldsymbol{\Sigma}_f^{-1}(\mathbf{x} - \boldsymbol{\mu}_f) + (\mathbf{x} - \boldsymbol{\mu}_g)^\top \boldsymbol{\Sigma}_g^{-1}(\mathbf{x} - \boldsymbol{\mu}_g) \tag{5.3}$$

has a finite lower bound, which holds if and only if $(\boldsymbol{\Sigma}_g^{-1} - \boldsymbol{\Sigma}_f^{-1})$ is positive definite. In

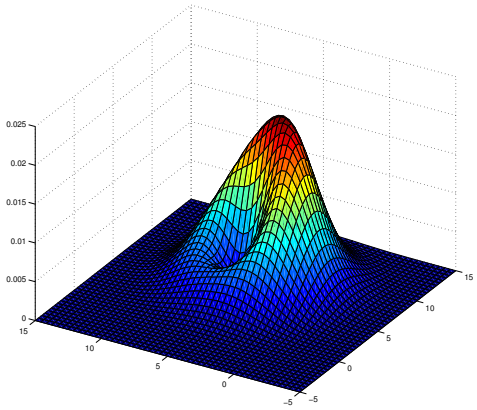


Figure 5.1: An algebraic mixture of two spherical Gaussians $m = 1.5f - 0.5g$ where $f = \mathcal{N}(\begin{bmatrix}11.4 & -3.4\end{bmatrix}^\top, 8\mathbf{I})$ and $g = \mathcal{N}(\begin{bmatrix}11.9 & -1.9\end{bmatrix}^\top, 4\mathbf{I})$.

that case, the minimum m of (5.3) is attained for

$$\boldsymbol{\mu}_0 = \boldsymbol{\Sigma}_0(\boldsymbol{\Sigma}_g^{-1}\boldsymbol{\mu}_g - \boldsymbol{\Sigma}_f^{-1}\boldsymbol{\mu}_f)$$

where $\boldsymbol{\Sigma}_0 = (\boldsymbol{\Sigma}_g^{-1} - \boldsymbol{\Sigma}_f^{-1})^{-1}$, and there exists a constant λ such that $\lambda g/f$ defines a Gaussian distribution of parameters $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$. It can be checked that

$$m = -(\boldsymbol{\mu}_f - \boldsymbol{\mu}_g)^\top \boldsymbol{\Sigma}_f^{-1}\boldsymbol{\Sigma}_0\boldsymbol{\Sigma}_g^{-1}(\boldsymbol{\mu}_f - \boldsymbol{\mu}_g).$$

Note that if the two distributions are distinct, then $\left(\frac{|\boldsymbol{\Sigma}_g|}{|\boldsymbol{\Sigma}_f|}\right)^{1/2} e^{m/2} - 1 < 0$. Otherwise, any positive α would be suitable and by dividing (5.2) by α, the density of the first distribution would be everywhere larger than the density of the second, which cannot happen. Hence every

$$\alpha \in \left]1, \left(1 - \sqrt{\frac{|\boldsymbol{\Sigma}_g|}{|\boldsymbol{\Sigma}_f|}} e^{m/2}\right)^{-1}\right]$$

defines a valid algebraic mixture of the two distributions.

For the special case of spherical Gaussians, i.e. $\boldsymbol{\Sigma}_f = \sigma_f^2\mathbf{I}$ and $\boldsymbol{\Sigma}_g = \sigma_g^2\mathbf{I}$, we obtain the following result.

Proposition 17. *Let $f = \mathcal{N}(\boldsymbol{\mu}_f, \sigma_f^2\mathbf{I})$ and $g = \mathcal{N}(\boldsymbol{\mu}_g, \sigma_g^2\mathbf{I})$ be two spherical Gaussian distributions. Then $\alpha f - (\alpha - 1)g$ defines an algebraic mixture if and only if*

$$\sigma_f > \sigma_g \text{ and } 1 < \alpha \leq \left(1 - \frac{\sigma_g^d}{\sigma_f^d} \exp\left\{-\frac{1}{2}\frac{||\boldsymbol{\mu}_f - \boldsymbol{\mu}_g||^2}{\sigma_f^2 - \sigma_g^2}\right\}\right)^{-1}.$$

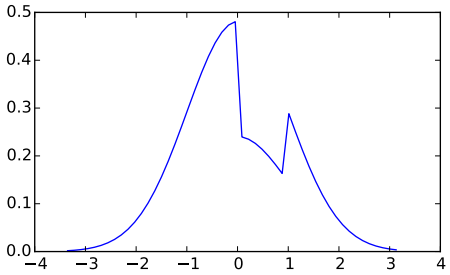


Figure 5.2: A distorted distribution $\mathcal{D}_{f,\phi}$ where f is the pdf of the univariate Gaussian $\mathcal{N}(0,1)$ and $\phi(x) = 0.5$ if $0 \leq x \leq 1$ and 0 otherwise.

5.2.2 Distorted Distributions

We now introduce a missing data model that is closely related to algebraic mixtures. Here we do not consider a missing data scenario where components of some vector are missing, but rather a scenario where data is missing from a training sample: let f be a PDF on $\mathcal{X}$ and let ϕ be a function defined on $\mathcal{X}$ and taking its values in $[0,1]$, so that $\phi(x)$ can be interpreted as a probability for any element x. We consider the following sampling process: an element x is drawn according to the distribution $\mathcal{D}_f$ and then, x is rejected with probability $\phi(x)$, or retained with probability $1-\phi(x)$. We denote by $\mathcal{D}_{f,\phi}$ the distribution of the observed data. Any sample $x_1, \ldots, x_n$ independently drawn according to $\mathcal{D}_{f,\phi}$ can be seen as a sample of $\mathcal{D}_f$ where some elements are missing. We call *distorted* such distributions and we will refer to the component f as the *underlying true distribution* and to ϕ as the *mask component*. An example of such a distribution is shown in Figure 5.2.

It turns out that distorted distributions are strictly equivalent to algebraic mixtures. Indeed, let $\beta = \int_{\mathcal{X}} f(x)\phi(x)dx$ be the probability to reject an element of $\mathcal{X}$ drawn according to $\mathcal{D}_f$ and let g be the density function defined by $g(x) = \beta^{-1} f(x)\phi(x)$. Observing that the density function m of $\mathcal{D}_{f,\phi}$ is proportional to $f(x)(1-\phi(x))$ it follows that

$$m(x) = \frac{f(x)}{1-\beta}(1-\phi(x)) = \frac{1}{1-\beta}(f(x) - \beta g(x)).$$

Hence by letting $\alpha = 1/(1-\beta)$, the distribution $\mathcal{D}_m$ is equal to the *algebraic mixture* $\alpha f - (\alpha-1)g$ of the distributions $\mathcal{D}_f$ and $\mathcal{D}_g$. Conversely, any algebraic mixture $\alpha f - (\alpha-1)g$ with $\alpha > 1$ is equivalent to the distorted distribution $\mathcal{D}_{f,\phi}$ where $\phi : x \mapsto \frac{\alpha-1}{\alpha}\frac{g(x)}{f(x)}$. Indeed, it can easily checked that ϕ takes its value in $[0,1]$ (which directly follows from the fact that $\alpha f - (\alpha-1)g \geq 0$ and $\alpha > 1$) and that the PDF of the distorted distribution $\mathcal{D}_{f,\phi}$ is proportional to $f - (\alpha-1)g/\alpha$.

This equivalence between algebraic mixtures and distorted distributions allow us to gain some intuition on the distributions modeled by algebraic mixtures: the positive components of the mixture can be interpreted similarly to classical mixture models, but the negative components of the mixture defines repulsive regions of the space where it is less likely to observe data. Let us consider a silly example. Going back to the distribution of heights in a population of men and women, suppose that on the day

where we measured the height of random people coming out from the train station, there was some very exclusive congress restricted to people (either men or women) who were between 1.5 and 1.7 meters tall, and that one half of the people that could go to this congress chose to do so rather than letting us measure them. The distribution of the data we gathered this day would be perfectly described by a distorted distribution: the underlying true distribution f would be the mixture of two Gaussians modeling the distribution of heights in the population, and the mask component ϕ would be defined by $\phi(x) = 0.5$ if $1.5 \leq x \leq 1.7$ and 0 otherwise. In this example, the mask component allows us to modelize the fact that it was less likely to observe people between 1.5 and 1.7 meters tall on that day.

Such a scenario where all the individuals are not equally likely to have been selected is known as the *sampling or selection bias* problem (Heckman, 1977; Berk, 1983). This phenomena is commonly encountered in e.g. medical studies: for instance most women that participate in breast cancer screening are middle-aged and have likely participated in another screening in the previous years, thus the sample obtained from the screening is not representative of a general population and will contain very few disease cases.

Another setting where algebraic mixture and distorted distributions could prove useful is the problem of drawing inferences from presence-only data (see e.g Keating and Cherry, 2004; Pearce and Boyce, 2006; Phillips et al., 2009). For example if we want to determine which habitat turtles prefer, we will only have access to data on where turtles were observed, but not on where turtles actually are[b]. If we did not observe any turtle in some specific habitat we cannot conclude that turtles do not live in it, maybe there was a turtle there but we did not see it. Such a setting could be modeled using a distorted distribution where the mask component would help us express this uncertainty on whether a turtle was actually there when we did not observe any. For example, areas where it is more likely to not see a turtle that is there, such as regions with tall sea grass, would correspond to an higher density of the mask component of the distorted distribution.

Lastly, we mention the notion of *contrastive learning* for mixture models that was introduced in (Zou et al., 2013). In constrastive learning, one wants to differentiate between background and foreground data by learning a latent variable model capturing relationships that appear in the foreground but not in the background. The method proposed by the authors is closely related to algebraic mixtures as they want to learn the difference between two mixture models, the first one modeling the foreground data and the second one the background data[c].

5.2.3 Algebraic Mixtures and Weighted Automata

Before showing how algebraic mixtures can be learned using the tensor method of moments we present a fundamental relation between probabilistic and weighted automata where algebraic mixtures naturally appear. A *probabilistic automaton* (PA) on strings is a weighted automaton $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ whose coefficients (i.e. the components of $\boldsymbol{\alpha}, \boldsymbol{\omega}$ and $\mathbf{A}^\sigma$) are all non negative and who satisfies the following syntactical

[b] This example is taken from a talk by Ioanna Manolopoulou that was given at the StatLearn workshop in April 2013.

[c] The method they propose is also an extension of the power method of moments and is actually equivalent to the one we propose (see Section 5.3.3).

conditions:

$$\boldsymbol{\alpha}^\top \mathbf{1} = 1, \;\; \mathbf{I} - \sum_{\sigma\in\Sigma} \mathbf{A}^\sigma \text{ is invertible, and } \sum_{\sigma\in\Sigma} \mathbf{A}^\sigma \mathbf{1} + \boldsymbol{\omega} = \mathbf{1}$$

where $\mathbf{1} = [1, 1, \cdots, 1]^\top$. In a PA, every weight of the automaton can be interpreted as a probability, and the conditions above imply that the components of the vector $\boldsymbol{\alpha}$ can be interpreted as probabilities of starting a computation in each state, and that the components of each $\mathbf{A}^\sigma$ and $\boldsymbol{\omega}$ can be interpreted as emission/transition and stopping probabilities. It is easy to check that these conditions imply that $\sum_{w\in\Sigma^*} f_A(w) = 1$ and that a PA computes a probability distribution on Σ^*.

It is shown in (Dupont, Denis, and Esposito, 2005) that hidden Markov models (HMM) and PAs are strictly equivalent (they define the same probability distributions), but there exist recognizable probability distributions on strings that cannot be computed by a PA or an HMM (see Appendix 5.A). However, we will now show that any distribution that can be computed by a PA (or an HMM) can be defined as an algebraic mixture of at most two distributions that can be computed by PAs. We start by showing that any WA can be expressed as an algebraic mixtures of two WAs with non negative coefficients.

Lemma 7. *Any recognizable function $f : \Sigma^* \to \mathbb{R}$ is the difference of two recognizable functions with non negative coefficients.*

Proof. For any real number a let $\phi^+(a) = \max\{x, 0\}$ and $\phi^-(a) = \max\{-x, 0\}$ (thus $a = \phi^+(a) - \phi^-(a)$). We extend these operators to vectors and matrices by applying them componentwise (i.e. to all their coefficients).

Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ be a WA and let $A^+ = (\mathbb{R}^{2n}, \tilde{\boldsymbol{\alpha}}^+, \{\tilde{\mathbf{A}}^\sigma\}_{\sigma\in\Sigma}, \tilde{\boldsymbol{\omega}})$ and $A^- = (\mathbb{R}^{2n}, \tilde{\boldsymbol{\alpha}}^-, \{\tilde{\mathbf{A}}^\sigma\}_{\sigma\in\Sigma}, \tilde{\boldsymbol{\omega}})$ be the WAs *with non negative coefficients* defined by

$$\tilde{\boldsymbol{\alpha}}^+ = \begin{bmatrix} \phi^+(\boldsymbol{\alpha}) \\ \phi^-(\boldsymbol{\alpha}) \end{bmatrix}, \;\; \tilde{\boldsymbol{\alpha}}^- = \begin{bmatrix} \phi^-(\boldsymbol{\alpha}) \\ \phi^+(\boldsymbol{\alpha}) \end{bmatrix}, \;\; \tilde{\boldsymbol{\omega}} = \begin{bmatrix} \phi^+(\boldsymbol{\omega}) \\ \phi^-(\boldsymbol{\omega}) \end{bmatrix} \;\; \text{and} \;\; \tilde{\mathbf{A}}^\sigma = \begin{bmatrix} \phi^+(\mathbf{A}^\sigma) & \phi^-(\mathbf{A}^\sigma) \\ \phi^-(\mathbf{A}^\sigma) & \phi^+(\mathbf{A}^\sigma) \end{bmatrix}$$

for each $\sigma \in \Sigma$. We claim that $f_A = f_{A^+} - f_{A^-}$. Indeed, first check by induction on the length of w that $\begin{bmatrix} \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \tilde{\mathbf{A}}^w = \begin{bmatrix} \mathbf{A}^w & -\mathbf{A}^w \end{bmatrix}$ where $\mathbf{A}^w = \mathbf{A}^{w_1} \cdots \mathbf{A}^{w_k}$ for any word $w = w_1 \cdots w_k \in \Sigma^*$ (and similarly for $\tilde{\mathbf{A}}^w$). Observing that $(\tilde{\boldsymbol{\alpha}}^+ - \tilde{\boldsymbol{\alpha}}^-)^\top = \boldsymbol{\alpha}^\top \begin{bmatrix} \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix}$ it follows that

$$\begin{aligned} f_{A^+}(w) - f_{A^-}(w) &= (\tilde{\boldsymbol{\alpha}}^+ - \tilde{\boldsymbol{\alpha}}^-)^\top \tilde{\mathbf{A}}^w \tilde{\boldsymbol{\omega}} \\ &= \boldsymbol{\alpha}^\top \begin{bmatrix} \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \tilde{\mathbf{A}}^w \tilde{\boldsymbol{\omega}} \\ &= \boldsymbol{\alpha}^\top \begin{bmatrix} \mathbf{A}^w & -\mathbf{A}^w \end{bmatrix} \begin{bmatrix} \phi^+(\boldsymbol{\omega}) \\ \phi^-(\boldsymbol{\omega}) \end{bmatrix} \\ &= \boldsymbol{\alpha}^\top \mathbf{A}^w (\phi^+(\boldsymbol{\omega}) - \phi^-(\boldsymbol{\omega})) \\ &= \boldsymbol{\alpha}^\top \mathbf{A}^w \boldsymbol{\omega} = f_A(w). \end{aligned}$$

□

The next step would consist in normalizing the (non negative) recognizable functions f_{A^+} and f_{A^-} obtained from the previous construction to express a recognizable probability distribution p as a difference of two distributions. However, even if the

recognizable function p is convergent, the recognizable functions f_{A^+} and f_{A^-} can be divergent (see an example in Appendix 5.A). Nonetheless, it has been shown in (Bailly and Denis, 2011) that if p is absolutely convergent (which is trivially the case for a probability distribution), then there always exists a WA $(\mathbb{R}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ computing p such that the WA $(\mathbb{R}^n, |\boldsymbol{\alpha}|, \{|\mathbf{A}^\sigma|\}_{\sigma\in\Sigma}, |\boldsymbol{\omega}|)$ defines a positive convergent function f'. In that case, both f_{A^+} and f_{A^-} are bounded by f' and are convergent. Now, let $\gamma^+ = \sum_{w\in\Sigma^*} f_{A^+}(w)$ and $\gamma^- = \sum_{w\in\Sigma^*} f_{A^-}(w)$. It is easy to check that $\gamma^+ - \gamma^- = 1$, that the functions $p^+ = \frac{f_{A^+}}{\gamma^+}$ and $p^- = \frac{f_{A^-}}{\gamma^-}$ are recognizable probability distributions, and thus that $p = \gamma^+ p^+ - (\gamma^+ - 1)p^-$ is an algebraic mixture. It just remains to show that p^+ and p^- can be computed by a probabilistic automaton, which is done in the following lemma.

Lemma 8. *Any probability distribution on Σ^* that can be computed by a WA with non negative coefficients can be computed by a PA.*

Proof. Let $A = (\mathbb{R}^n, \boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ be a minimal WA computing a probability distribution, let $\mathbf{v} = (\mathbf{I} - \sum_{\sigma\in\Sigma} \mathbf{A}^\sigma)^{-1}\boldsymbol{\omega}$ and let $\mathbf{D} = diag(\mathbf{v})$ be the diagonal matrix defined by $\mathbf{D}_{ii} = \mathbf{v}_i$. Then the conjugate WA $A^{\mathbf{D}} = (\mathbb{R}^n, \mathbf{D}\boldsymbol{\alpha}, \{\mathbf{D}^{-1}\mathbf{A}^\sigma\mathbf{D}\}_{\sigma\in\Sigma}, \mathbf{D}^{-1}\boldsymbol{\omega})$ is a PA that computes f_A. Indeed, since A is minimal with non negative coefficients $\mathbf{D}$ is invertible and non negative, and it is easy to check that $A^{\mathbf{D}}$ has non negative coefficients and that $f_{A^{\mathbf{D}}} = f_A$. Moreover, since $\boldsymbol{\alpha}^\top\mathbf{v} = \sum_{w\in\Sigma^*} f_A(w) = 1$ we have $(\mathbf{D}\boldsymbol{\alpha})^\top\mathbf{1} = \boldsymbol{\alpha}^\top\mathbf{v} = 1$, $\mathbf{I} - \sum_{\sigma\in\Sigma} \mathbf{D}^{-1}\mathbf{A}^\sigma\mathbf{D} = \mathbf{D}^{-1}(\mathbf{I} - \sum_{\sigma\in\Sigma}\mathbf{A}^\sigma)\mathbf{D}$ is invertible and $(\mathbf{I} - \mathbf{D}^{-1}\sum_{\sigma\in\Sigma}\mathbf{A}^\sigma\mathbf{D})^{-1}(\mathbf{D}\boldsymbol{\omega}) = \mathbf{D}^{-1}\mathbf{v} = \mathbf{1}$. □

Combining the previous lemmas we obtain the following theorem.

Theorem 22. *Any probability distribution on strings that can be computed by a weighted automaton is equal to an algebraic mixture of at most two probabilistic automata.*

This theorem shows that while WAs are strictly more expressive than PAs, the set of probability distributions that can be computed by WAs can be obtained by simply allowing algebraic mixtures of PAs. For recognizable probability distribution, allowing negative weights in a weighted automaton is thus somehow equivalent to extending the set of PAs by closure under algebraic mixtures. This also shows that algebraic mixtures can be considerably more expressive than traditional mixtures (it is easy to check that a classical mixture of two PAs can be computed by a PA).

5.3 Learning Algebraic Mixtures with Tensor Decomposition

We now propose an extension of the *tensor method of moments* described in Section 5.1.1 to algebraic mixtures. Similarly to the tensor method of moments for classical mixture models, the first step consists in relating the mixing weights and parameters of an algebraic mixture to observable low order moments by the system of polynomial equations

$$\mathbf{M}_2 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \quad \text{and} \quad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i$$

where the $\boldsymbol{\mu}_i$'s are the parameters of the components of the mixture and the w_i's are the mixing weights, that may now be negative. Then, we need to solve this system of equations to recover the parameters of the mixture. The fact that some of the w_i's may be negative implies that the matrix $\mathbf{M}_2$ is not positive semi-definite anymore, which prevents us from directly applying the power method proposed in (Anandkumar et al., 2014). We propose an extension of this method to the case where some of the mixing weights are negative. Our extension needs to use complex square roots of negative real numbers and to introduce non-hermitian quadratic forms. Considering the model of algebraic mixture of spherical Gaussians, we start by showing how the low-order moments of an algebraic mixture exhibit a similar relation with the parameters of the model to the one presented in (Hsu and Kakade, 2013).

5.3.1 Low-Order Moments of an Algebraic Mixture of Spherical Gaussians

Let $f(\mathbf{x}) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i^2 \mathbf{I})$ be the PDF of the random vector $\mathbf{x}$, where $\boldsymbol{\mu}_i \in \mathbb{R}^n$ are the component means, σ_i^2 the component variances, and $w_i \neq 0$ the coefficients ($\sum_{i=1}^{k} w_i = 1$). Assuming that the component means are linearly independent, the following theorem generalizes the results from Theorem 20 to algebraic mixtures of spherical Gaussians.

Theorem 23. *The average variance $\bar{\sigma}^2 = \sum_{i=1}^{k} w_i \sigma_i^2$ is an eigenvalue of the covariance matrix $\mathbb{E}[(\mathbf{x}-\mathbb{E}[\mathbf{x}])(\mathbf{x}-\mathbb{E}[\mathbf{x}])^\top]$. More precisely, if r is the number of negative eigenvalues of the matrix*

$$\mathbf{M} = \sum_{i=1}^{k} w_i (\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{x}]) \circ (\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{x}])$$

then $\bar{\sigma}^2$ is the $(r+1)$-th smallest eigenvalue of the covariance matrix. Furthermore, if l is the number of negative coefficients w_i, i.e. $l = |\{w_i : i \in [k],\ w_i < 0\}|$, then r is either equal to l or $l+1$.

Let $\mathbf{v}$ be any unit-norm eigenvector corresponding to $\bar{\sigma}^2$ and let

$$\mathbf{m}_1 = \mathbb{E}[\mathbf{x}(\mathbf{v}^\top(\mathbf{x} - \mathbb{E}[\mathbf{x}]))^2], \qquad \mathbf{M}_2 = \mathbb{E}[\mathbf{x} \circ \mathbf{x}] - \bar{\sigma}^2 \mathbf{I}, \quad \textit{and}$$

$$\boldsymbol{\mathcal{M}}_3 = \mathbb{E}[\mathbf{x} \circ \mathbf{x} \circ \mathbf{x}] - \sum_{i=1}^{d} [\mathbf{m}_1 \circ \mathbf{e}_i \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{m}_1 \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{e}_i \circ \mathbf{m}_1]$$

where $\mathbf{e}_1, \cdots, \mathbf{e}_d$ is the coordinate basis of $\mathbb{R}^d$. Then,

$$\mathbf{m}_1 = \sum_{i=1}^{k} w_i \sigma_i^2 \boldsymbol{\mu}_i, \qquad \mathbf{M}_2 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \ , \ \textit{and} \qquad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i.$$

Proof. The proof of this theorem is given in Appendix 5.B. □

The arguments used to prove the relations between the low-order moments and the tensors $\mathbf{m}_1$, $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ in the previous theorem are similar to the ones used in (Hsu and Kakade, 2013). The main difficulty in extending this result to algebraic mixtures is the identification of the position of the eigenvalue $\bar{\sigma}^2 = \sum_i w_i \sigma_i^2$ of the covariance

matrix. Indeed, $\bar{\sigma}^2$ is still an eigenvalue of the covariance matrix, but it is not the first one anymore, and the computation of the tensors $\mathbf{m}_1$, $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ depends on the identification of an eigenvector of the covariance matrix associated with this eigenvalue.

5.3.2 Tensor Power Method for Complex-Valued Tensors

It follows from the previous theorem that, similarly to the classical mixture case, the estimation of the components of the mixture can be reduced to solving a system of polynomial equations. We now propose an extension of the tensor power method to the case where the second order tensor $\mathbf{M}_2$ is not necessarily positive semi-definite. We consider systems of the form

$$\mathbf{M}_2 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \quad \text{and} \quad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \tag{5.4}$$

where the vectors $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ are linearly independent and $w_1, \cdots, w_k \in \mathbb{R}$ are non zero (but *may be negative*).

We will show how the parameters w_i and $\boldsymbol{\mu}_i$ can be recovered from $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ using a *power method for complex-valued tensors*. Recall that the power method, on which the tensor method of moments proposed in (Anandkumar et al., 2014) relies, first uses the matrix $\mathbf{M}_2$ to obtain a whitening transformation of the tensor $\boldsymbol{\mathcal{T}}$ admitting an orthonormal decomposition. When $\mathbf{M}_2$ is not positive semi-definite, this whitening process will result in a complex valued tensor admitting a *pseudo-orthonormal* decomposition (see below), hence the need for a power method for complex-valued tensors.

Pseudo-Orthonormalization. We call a set $\{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k\} \subset \mathbb{C}^d$ *pseudo-orthonormal* if $\boldsymbol{\nu}_i^\top \boldsymbol{\nu}_j = \delta_{ij}$ for all $i, j \in [k]$. Note that for any $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_d) \in \mathbb{C}^d$, $\boldsymbol{\nu}^\top \boldsymbol{\nu} = \nu_1^2 + \ldots + \nu_d^2 \in \mathbb{C}$ and in particular, $\boldsymbol{\nu}^\top \boldsymbol{\nu} \neq ||\boldsymbol{\nu}||_2^2 = |\nu_1|^2 + \ldots + |\nu_n|^2$. It can easily be checked that a pseudo-orthonormal set is linearly independent. We say that a tensor decomposition $\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} z_i \boldsymbol{\nu}_i^{\circ p}$ of a complex-valued tensor $\boldsymbol{\mathcal{T}} \in (\mathbb{C}^d)^{\otimes p}$ is *pseudo-orthonormal* if $\{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k\}$ is a pseudo-orthonormal set.

As in (Anandkumar et al., 2014), we build a whitening matrix $\mathbf{W}$ from $\mathbf{M}_2$, and we use $\mathbf{W}$ to obtain a transformation of the tensor $\boldsymbol{\mathcal{M}}_3$ admitting a pseudo-orthonormal decomposition.

Identifying $\mathbf{M}_2$ with the symmetric rank-k matrix $\sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$, let $\mathbf{U}\mathbf{D}\mathbf{U}^\top$ be the eigendecomposition of $\mathbf{M}_2$, where $\mathbf{D}$ is the $k \times k$ diagonal matrix whose diagonal elements are composed of the k non-zero eigenvalues of $\mathbf{M}_2$ and where $\mathbf{U}$ is a $d \times k$ matrix satisfying $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_k$ and $\mathbf{U}\mathbf{U}^\top \boldsymbol{\mu}_i = \boldsymbol{\mu}_i$ for any $i \in [k]$. Let $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{C}^{d \times k}$ and $\boldsymbol{\nu}_i = w_i^{\frac{1}{2}} \mathbf{W}^\top \boldsymbol{\mu}_i \in \mathbb{C}^k$ for $i \in [k]$ where we consider complex square roots of the negative components of $\mathbf{D}$ and w_i: $x^{1/2} = i|x|^{1/2}$ and $x^{-1/2} = (x^{1/2})^{-1} = -i|x|^{-1/2}$ if $x < 0$. We have

$$\sum_{i=1}^{k} \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top = \mathbf{W}^\top \left(\sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top \right) \mathbf{W} = \mathbf{W}^\top \mathbf{M}_2 \mathbf{W} = \mathbf{I}_k$$

hence $\boldsymbol{\nu}_i^\top \boldsymbol{\nu}_j = \delta_{ij}$ for all $i, j \in [k]$. Now let $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$. It is easy to check that

$\boldsymbol{\mathcal{T}}$ admits the pseudo-orthonormal decomposition

$$\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} w_i (\mathbf{W}^\top \boldsymbol{\mu}_i)^{\circ 3} = \sum_{i=1}^{k} z_i \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i$$

where $z_i = w_i^{-\frac{1}{2}}$ for all $i \in [k]$.

Tensor Power Method. We now present a tensor power method for complex-valued tensors admitting a pseudo-orthonormal decomposition. Note that the parameters of such a decomposition are not fully identifiable since $z\boldsymbol{\nu}^{\circ 3} = (-z)(-\boldsymbol{\nu})^{\circ 3}$. The following theorem extends Lemma 5.1 of (Anandkumar et al., 2014). This theorem shows that a couple $(z_i, \boldsymbol{\nu}_i)$ from a pseudo-orthonormal decomposition can be recovered using a tensor power method for complex-valued tensors (with a quadratic convergence rate).

Theorem 24. *Let $\boldsymbol{\mathcal{T}} \in (\mathbb{C}^d)^{\otimes 3}$ have a pseudo-orthonormal decomposition*

$$\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} z_i \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i \circ \boldsymbol{\nu}_i,$$

and let T be the mapping defined by $T(\boldsymbol{\theta}) = \boldsymbol{\mathcal{T}}(\mathbf{I}, \boldsymbol{\theta}, \boldsymbol{\theta})$ for any $\boldsymbol{\theta} \in \mathbb{C}^d$. Let $\boldsymbol{\theta}_0 \in \mathbb{C}^d$ and assume that $|z_1.\boldsymbol{\nu}_1^\top \boldsymbol{\theta}_0| > |z_2.\boldsymbol{\nu}_2^\top \boldsymbol{\theta}_0| \geq \cdots \geq |z_k.\boldsymbol{\nu}_k^\top \boldsymbol{\theta}_0| > 0$. For $t = 1, 2, \cdots$, define

$$\boldsymbol{\theta}_t = \frac{T(\boldsymbol{\theta}_{t-1})}{[T(\boldsymbol{\theta}_{t-1})^\top T(\boldsymbol{\theta}_{t-1})]^{\frac{1}{2}}} \quad \text{and} \quad \lambda_t = \boldsymbol{\mathcal{T}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t, \boldsymbol{\theta}_t) \tag{5.5}$$

where we assume that $\boldsymbol{\theta}_0$ is such that $T(\boldsymbol{\theta}_t)^\top T(\boldsymbol{\theta}_t) \neq 0$ for all t. Then, $\boldsymbol{\theta}_t \to \pm\boldsymbol{\nu}_1$ and $\lambda_t \to \pm z_1$ as $t \to \infty$.

More precisely, let $M = \max\left\{1, \frac{|z_1|^2}{|z_i|^2}, |z_1|\frac{\|\boldsymbol{\nu}_i\|}{|z_i|} : i \in [k]\right\}$ and $\varepsilon_t = kM\left|\frac{z_2.\boldsymbol{\nu}_2^\top \boldsymbol{\theta}_0}{z_1.\boldsymbol{\nu}_1^\top \boldsymbol{\theta}_0}\right|^{2^t}$. Then for all $t \geq 2$ such that $\varepsilon_t < \frac{1}{2}$, we have

$$|e_t f_t \lambda_t - z_1| \leq 7|z_1|\varepsilon_t \quad \text{and} \quad \|e_t f_t \boldsymbol{\theta}_t - \boldsymbol{\nu}_1\| \leq \varepsilon_t \left(||\boldsymbol{\nu}_1|| + \sqrt{2}\right),$$

where $(e_t)_t$ and $(f_t)_t$ are two sequences defined in the proof and taking their values in $\{-1, 1\}$.

Proof. The proof is given in Appendix 5.C. □

This theorem directly yields an algorithm to recover the parameters of a pseudo-orthonormal decomposition of a complex-valued tensor using the standard deflation technique.

It can be shown that if $\boldsymbol{\theta}_0$ is chosen at random in $\mathbb{C}^d$, the assumptions on $\boldsymbol{\theta}_t$ in the previous theorem are satisfied with probability one. We give a proof in the following lemma for the assumption $T(\boldsymbol{\theta}_t)^\top T(\boldsymbol{\theta}_t) \neq 0$, similar arguments can be used for the assumption $|z_1.\boldsymbol{\nu}_1^\top \boldsymbol{\theta}_0| > |z_2.\boldsymbol{\nu}_2^\top \boldsymbol{\theta}_0| \geq \cdots \geq |z_k.\boldsymbol{\nu}_k^\top \boldsymbol{\theta}_0| > 0$.

Lemma 9. *Using the definitions of Theorem 24, the set $S = \{\boldsymbol{\theta}_0 \in \mathbb{C}^d \mid \exists t \geq 0 : T(\boldsymbol{\theta}_t)^\top T(\boldsymbol{\theta}_t) = 0\}$ has Lebesgue measure zero in $\mathbb{C}^d$.*

Proof. We first define the sequence $(\tilde{\boldsymbol{\theta}}_t)_{t\geq 0}$ by $\tilde{\boldsymbol{\theta}}_0 = \boldsymbol{\theta}_0$ and $\tilde{\boldsymbol{\theta}}_t = T(\tilde{\boldsymbol{\theta}}_{t-1})$ for any $t \geq 1$ (i.e. the sequence generated by the power method without pseudo-normalization). We have $T(\boldsymbol{\theta}_{t-1})^\top T(\boldsymbol{\theta}_{t-1}) = \left(\widetilde{\boldsymbol{\theta}}_{t-1}^\top \widetilde{\boldsymbol{\theta}}_{t-1}\right)^{-2} \widetilde{\boldsymbol{\theta}}_t^\top \widetilde{\boldsymbol{\theta}}_t$ and one can check by induction on $t \geq 1$ that $\tilde{\boldsymbol{\theta}}_t = \sum_{i=1}^k z_i^{2^t-1} (\boldsymbol{\nu}_i^\top \boldsymbol{\theta}_0)^{2^t} \boldsymbol{\nu}_i$, hence $\widetilde{\boldsymbol{\theta}}_t^\top \widetilde{\boldsymbol{\theta}}_t = \sum_{i=1}^k (z_i^{2^t-1} (\boldsymbol{\nu}_i^\top \boldsymbol{\theta}_0)^{2^t})^2$ for all $t \geq 1$. For a given integer t, the set

$$S_t = \left\{ \boldsymbol{\theta} \in \mathbb{C}^d : P_t(\boldsymbol{\theta}) \triangleq \sum_{i=1}^k (z_i^{2^t-1} (\boldsymbol{\nu}_i^\top \boldsymbol{\theta})^{2^t})^2 = 0 \right\}$$

is the set of zeros of a multivariate polynomial. If P_t is non-trivial (i.e. different from zero), it is a proper algebraic subvariety of $\mathbb{C}^d$ of dimension less than n, thus of Lebesgue measure 0 (Federer, 2014, Section 2.6.5). Since $S = \cup_{t=0}^\infty S_t$, it is sufficient to show that P_t is non-trivial for any index t.

Without loss of generality, we assume that there exists at least one $i \in [k]$ such that the first component $(\boldsymbol{\nu}_i)_1$ of the vector $\boldsymbol{\nu}_i$ is not null. Suppose that P_t is null, then all of its monomials are null. In particular, the coefficient associated with $\theta_1^{2^{t+1}-1}\theta_j$, which is proportional to $\sum_{i=1}^k z_i^{2^{t+1}-2} (\boldsymbol{\nu}_i)_1^{2^{t+1}-1} (\boldsymbol{\nu}_i)_j$, is null for all $j \in [n]$. Let $\alpha_i = z_i^{2^{t+1}-2} (\boldsymbol{\nu}_i)_1^{2^{t+1}-1}$ for $i \in [k]$, and note that since $z_i \neq 0$ for all $i \in [k]$, we cannot have all the α_i equal to zero. Thus, we have $\sum_i \alpha_i (\boldsymbol{\nu}_i)_j = 0$ for all $j \in [n]$, i.e. $\sum_{i=1}^k \alpha_i \boldsymbol{\nu}_i = \mathbf{0}$ which is in contradiction with the linear independence of $\{\boldsymbol{\nu}_i\}_{i=1}^k$.

□

We can now state the following theorem, which summarizes the overall procedure to recover the parameters of a system of the form (5.4) using pseudo-orthonormalization and the complex tensor power method. Note that this procedure generalizes the one proposed in (Anandkumar et al., 2014): if all the weights $w_1, \cdots, w_k$ are positive, the method we propose boils down to theirs.

Theorem 25. *Let $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ be linearly independent vectors, let $w_1, \cdots, w_k \in \mathbb{R}$ be non-zero weights and let*

$$\mathbf{M}_2 = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \quad \textit{and} \quad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i.$$

Let $\mathbf{U}\mathbf{D}\mathbf{U}^\top$ be the eigendecomposition of $\mathbf{M}_2$, let $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{C}^{d\times k}$ and let $(\mathbf{W}^\top)^+ = \mathbf{U}\mathbf{D}^{\frac{1}{2}}$. Finally, let $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W}) \in \mathbb{C}^{k\times k\times k}$ and let $\boldsymbol{\theta}_0$ be drawn at random in $\mathbb{C}^k$.

Then, using the definitions of $\boldsymbol{\theta}_t$ and λ_t in Eq. (5.5), we have

$$\lim_t \frac{1}{\lambda_t^2} = w_j \quad \textit{and} \quad \lim_t \lambda_t (\mathbf{W}^\top)^+ \boldsymbol{\theta}_t = \boldsymbol{\mu}_j$$

with probability one, where $j = \arg\max_i \{ \left| \boldsymbol{\mu}_i^\top \mathbf{W} \boldsymbol{\theta}_0 \right| \}$.

Note that the indeterminacy on the sign of the coefficients in the pseudo-orthonormal decomposition $\boldsymbol{\mathcal{T}} = \sum_{i=1}^k w_i^{-\frac{1}{2}} \left(w_i^{\frac{1}{2}} \mathbf{W}^\top \boldsymbol{\mu}_i \right)^{\circ 3}$ vanishes when we recover the original pa-

Algorithm 7 Power Method for Algebraic Mixture Estimation

Input: $\hat{\mathbf{M}}_2 \in \mathbb{R}^{d\times d}$, $\hat{\boldsymbol{\mathcal{M}}}_3 \in \mathbb{R}^{d\times d\times d}$ and tolerance parameter ε
Output: $w_1, \cdots, w_k \in \mathbb{R}$, $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k \in \mathbb{R}^d$

Compute the eigen decomposition $\hat{\mathbf{M}}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$
$\mathbf{W} \leftarrow \mathbf{U}\mathbf{D}^{-\frac{1}{2}}$; $\boldsymbol{\mathcal{T}} \leftarrow \hat{\boldsymbol{\mathcal{M}}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$; $i \leftarrow 1$
repeat
  Draw $\boldsymbol{\theta}$ at random in $\mathbb{C}^d$
  repeat
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\mathcal{T}}(I, \boldsymbol{\theta}, \boldsymbol{\theta})$; $\boldsymbol{\theta} \leftarrow \frac{\boldsymbol{\theta}}{(\boldsymbol{\theta}^\top \boldsymbol{\theta})^{\frac{1}{2}}}$
  until convergence
  $\lambda \leftarrow \boldsymbol{\mathcal{T}}(\boldsymbol{\theta}, \boldsymbol{\theta}, \boldsymbol{\theta})$; $\boldsymbol{\mathcal{T}} \leftarrow \boldsymbol{\mathcal{T}} - \lambda.\boldsymbol{\theta}^{\circ 3}$
  $w_i \leftarrow 1/\lambda^2$; $\boldsymbol{\mu}_i \leftarrow \lambda(\mathbf{W}^\top)^+\boldsymbol{\theta}$
  $i \leftarrow i + 1$
until $\|\boldsymbol{\mathcal{T}}\| \leq \varepsilon$
return $(w_1, \boldsymbol{\mu}_1), \cdots, (w_k, \boldsymbol{\mu}_k)$

rameters w_i and $\boldsymbol{\mu}_i$. The overall procedure to recover the parameters of the system (5.4) from estimations of the tensors $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ is summarized in Algorithm 7.

The previous theorem, combined with Theorem 23, yields a procedure to estimate the parameters of an algebraic mixture of spherical Gaussians: (i) compute the sample covariance matrix $\mathbf{S}$, (ii) for each candidate eigenvalue of $\mathbf{S}$ for $\bar{\sigma}^2$, estimate the tensors $\mathbf{m}_1$, $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ on the data (see Theorem 23), (iii) compute estimations of the parameters using Algorithm 7, (iv) choose the model that maximizes the likelihood of the learning data.

5.3.3 Avoiding Complex-Valued Tensors

We developed the method described in the previous section during the Fall 2013. We became aware shortly after that an extension of the tensor power method to the case where the matrix $\mathbf{M}_2$ is not positive semi-definite had been independently proposed in (Zou et al., 2013) in the context of contrastive learning. The method they propose is equivalent to the one we developed with the benefit of not having to deal with complex tensors. It relies on iterating the map $\mathbf{v} \mapsto \boldsymbol{\mathcal{M}}_3(\mathbf{I}, \mathbf{M}_2^+\mathbf{v}, \mathbf{M}_2^+\mathbf{v})$ until convergence to recover the first couple $(w_i, \boldsymbol{\mu}_i)$, and uses deflation to recover the remaining components of the decomposition. Observe that the pseudo-inverse $\mathbf{M}_2^+$ has real coefficients. In comparison, our method relies on iterating the map $\mathbf{v} \mapsto \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}\mathbf{v}, \mathbf{W}\mathbf{v})$ where $\mathbf{W}$ is the whitening matrix described above. As shown in the following proposition, it turns out that these two methods are strictly equivalent, which comes from the fact that $\mathbf{M}_2^+ = \mathbf{W}\mathbf{W}^\top$.

Proposition 18. *Let $\mathbf{M}_2 \in \mathbb{R}^{d\times d}$ and $\boldsymbol{\mathcal{M}}_3 \in \mathbb{R}^{d\times d\times d}$ be the matrix and tensor defined in Eq. (5.4). Let $\mathbf{M}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ bet the eigendecomposition of $\mathbf{M}_2$ and let $\mathbf{W} = \mathbf{U}\mathbf{D}^{-1/2} \in \mathbb{C}^{d\times d}$.*

If $F : \mathbb{C}^d \to \mathbb{C}^d$ and $G : \mathbb{R}^d \to \mathbb{R}^d$ are the maps defined by

$$F : \mathbf{v} \mapsto \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}\mathbf{v}, \mathbf{W}\mathbf{v}) \quad \text{and} \quad G : \mathbf{v} \mapsto \boldsymbol{\mathcal{M}}_3(\mathbf{I}, \mathbf{M}_2^+\mathbf{v}, \mathbf{M}_2^+\mathbf{v}).$$

then, $F^k(\mathbf{W}^\top \boldsymbol{\theta}) = \mathbf{W}^\top G^k(\boldsymbol{\theta})$ *for any* $\boldsymbol{\theta} \in \mathbb{R}^d$ *and any integer* $k \geq 0$.

Proof. This can be proved easily by induction on k. If $k = 0$ we have $F^0(\mathbf{W}^\top \boldsymbol{\theta}) = \mathbf{W}^\top \boldsymbol{\theta} = \mathbf{W}^\top G^0(\boldsymbol{\theta})$. Suppose the result true for integers up to $k-1$, then using the fact that $\mathbf{W}\mathbf{W}^\top = \mathbf{M}_2^+$ we get

$$\begin{aligned} F^k(\mathbf{W}^\top \boldsymbol{\theta}) &= \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}F^{k-1}(\mathbf{W}^\top \boldsymbol{\theta}), \mathbf{W}F^{k-1}(\mathbf{W}^\top \boldsymbol{\theta})) \\ &= \boldsymbol{\mathcal{M}}_3(\mathbf{W}, \mathbf{W}\mathbf{W}^\top G^{k-1}(\boldsymbol{\theta}), \mathbf{W}\mathbf{W}^\top G^{k-1}(\boldsymbol{\theta})) \\ &= \mathbf{W}^\top \boldsymbol{\mathcal{M}}_3(\mathbf{I}, \mathbf{M}_2^+ G^{k-1}(\boldsymbol{\theta}), \mathbf{M}_2^+ G^{k-1}(\boldsymbol{\theta})) \\ &= \mathbf{W}^\top G^k(\boldsymbol{\theta}). \end{aligned}$$

□

We conclude by mentioning alternative approaches to the tensor power method that rely on simultaneous diagonalization/Schur decomposition methods (Kuleshov, Chaganty, and Liang, 2015; Colombo and Vlassis, 2016). Note that connections between tensor decomposition and simultaneous diagonalization had already been noticed in earlier works (see e.g. De Lathauwer, 2006). These methods rely on the simple observation that for any vector $\mathbf{v}$ the projection $\boldsymbol{\mathcal{M}}_3(\mathbf{v}, \mathbf{I}, \mathbf{I}) = \sum_{i=1}^k \langle \mathbf{v}, \boldsymbol{\mu}_i \rangle w_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$ preserves all the information about the parameters w_i and $\boldsymbol{\mu}_i$. It the $\boldsymbol{\mu}_i$'s are orthonormal, then the parameters of the decomposition can simply be recovered by performing an eigendecomposition of this matrix (Kolda, 2015). However, this method can be very sensitive to noise and better robustness can be achieved by using simultaneous diagonalization of random projections (Kuleshov, Chaganty, and Liang, 2015) or simultaneous Schur decomposition of deterministic projections (Colombo and Vlassis, 2016). We think that these methods could be applied to the case where some of the weights in (5.4) are negative without major modifications.

5.4 Experiments

In order to assess the performance of the tensor power method for algebraic mixtures, we run experiments on synthetic data generated from the algebraic mixture of spherical Gaussians presented in Figure 5.1.

First we run Algorithm 7 on the exact tensors $\mathbf{M}_2 = \sum_i w_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$ and $\boldsymbol{\mathcal{M}}_3 = \sum_i w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i$ with various initializations of the initial vector $\boldsymbol{\theta}_0$ to extract the first eigenvector/eigenvalue pair. The corresponding parameters w_i and $\boldsymbol{\mu}_i$ are always exactly recovered in less than 10 iterations (which confirms the quadratic convergence rate of the power method established in Theorem 24). The average over 500 initializations of the recovery precision as a function of the number of iterations is plotted in Figure 5.3. The recovery precision is evaluated by the norm $\|\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}}_i\|$ and the absolute difference $|w_i - \hat{w}_i|$ where $\hat{\boldsymbol{\mu}}_i$ and $\hat{w}_i$ are the estimators returned by Algorithm 7.

Then, we test our algorithm in a learning setting on a density estimation task. For various sizes (ranging from 500 to 2,000,000), we generate a training data set using Algorithm 6 and use the method described above to estimate the parameters of the algebraic mixture of spherical Gaussians. This experiment is repeated 500 times. We consider two settings for our algorithm (Algmix): in the first one we only estimate the component means and weights of the mixture and use the true values for σ_1^2 and σ_2^2, while in the second we estimate all the parameters of the model. We distinguish these

two cases to demonstrate that the estimation of the variance parameters is very sensitive to noise for small sizes of the training data set.

We compare our method with the maximum likelihood method (ML) and the (standard) Gaussian mixture model with 2 and 4 components (GMM-2, GMM-4). The average log-likelihood (over the 500 runs) on a test set of size 2,000 is plotted in Figure 5.4. For small data sets, it sometimes happens that the tensor power method does not converge, and that our method returns aberrant models. We used the following scheme to circumvent this issue: when the log-likelihood of ML on the training set was higher than the one our algorithm returns, we used the maximum likelihood estimator instead. Note that the number of times this workaround has to be used vanishes as the size of the training data set grows.

We see that our method quickly outperforms the maximum likelihood baseline but that it needs large sample sizes to outperform more elaborate baselines such as GMM-2 and GMM-4. We also observe that the estimation of the variance parameters by the tensor method of moments is very sensitive to noise for small sizes of the data sets. However, as the sample size grows, using the true variance parameters or their estimation does not lead to a significant difference.

5.5 Conclusion

In this chapter we introduced the notion of algebraic mixtures and the model of distorted distribution and we showed that these two models are equivalent. Seeing algebraic mixtures as distorted distributions allowed us to give some insight on the distributions defined by algebraic mixtures: negative components of an algebraic mixture define repulsive regions of the space from which samples drawn from the positive components are more likely to be rejected. We proposed a general learning framework for algebraic mixtures defined over families of distributions for which low order observable moments are polynomials of the parameters of the model that can be recovered using tensor decomposition techniques. We showed that the family of spherical Gaussian distributions is such a family. The learning method we proposed generalizes the tensor method of moments proposed in (Anandkumar et al., 2014), which is itself a special case of the method of moments.

Experiments on synthetic data generated from an algebraic mixture showed that our method performs well for large sample sizes but is sensitive to noise and thus performs poorly for small sample sizes. In order to try to explain these results, we first observe an artifact of the tensor method of moments and in particular of the tensor power method on which it relies. Given the observable tensors $\mathbf{M}_2 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \in \mathbb{R}^{d \times d \times d}$, the tensor power method requires that the vectors $\boldsymbol{\mu}_i$ are linearly independent. If we consider a mixture of two Gaussians, this implies that if the two component means are colinear, the method cannot be applied. However, we can always translate the data in such a way that the two component means are orthogonal (assuming the means are not equal). In that case, the method can be applied (moreover it will not need a whitening step which is particularly sensitive to noise). Furthermore, the assumption on linear independence suggests that the component means should not be too close to each other. While this seems reasonable for traditional mixtures of Gaussian distributions (intuitively a mixture of two Gaussians

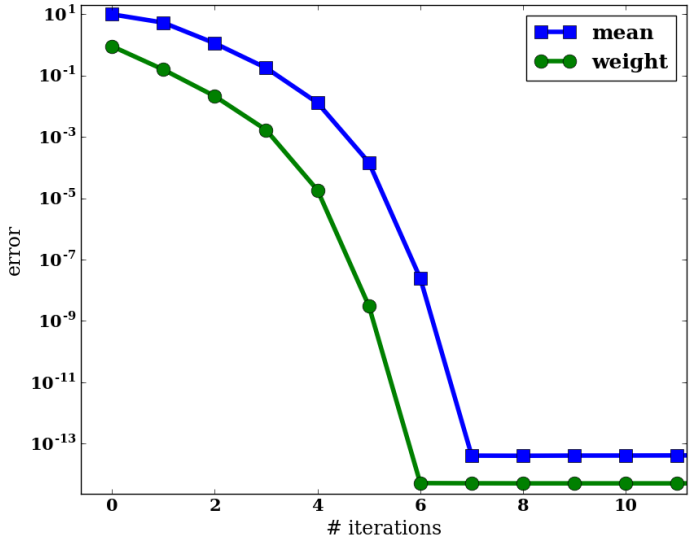


Figure 5.3: Recovery precision of the first couple $(w_i, \boldsymbol{\mu}_i)$ as a function of the number of iterations using Algorithm 7 on the exact tensors $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ (see Theorem 25). The y-axis corresponds to $|w_i - \hat{w}_i|$ for the error between weights and to $\|\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}}_i\|$ for the mean error.

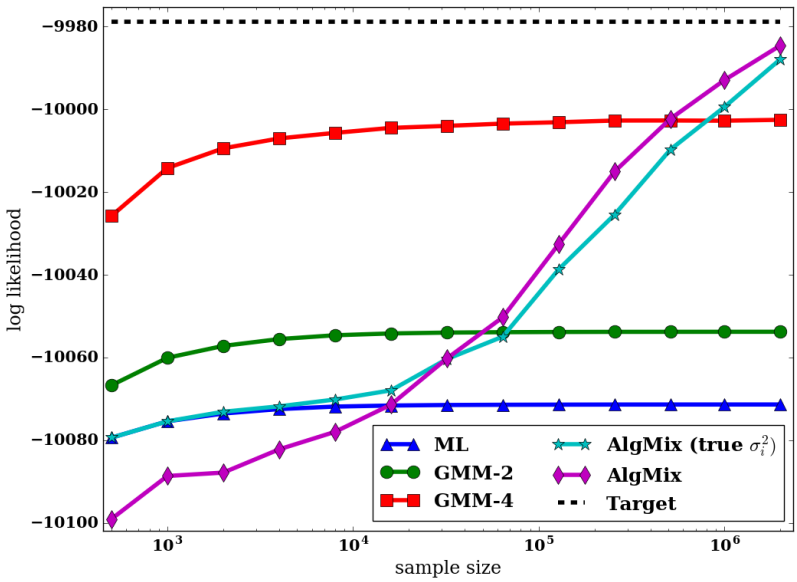


Figure 5.4: Log-likelihood of various estimator on a test set as a function of the training set size. The different estimator are: maximum likelihood (ML), Gaussian mixture model with 2 and 4 components (GMM-2 and GMM-4), the tensor method of moments using the true variance parameters and the same method using estimations of the variance parameters from the vector $\mathbf{m}_1$ defined in Theorem 23.

will be easier to learn if the component means are far from each other), it is the opposite for an algebraic mixture of two Gaussians where one of the weights is negative: when the component means are far away from each other, the variance of the negative component must be very large and/or the corresponding negative weight must be close to 0. This makes the algebraic mixture difficult to distinguish from its positive component. Thus the tensor method of moments may not be the more suitable learning method for algebraic mixtures.

In the future, we plan to investigate an alternative approach to learn algebraic mixtures relying on the notion of distorted distributions. Recall that a distorted distribution $\mathcal{D}_{f,\phi}$ is defined by the following sample process: first draw a point x from $\mathcal{D}_f$, and then reject this point with probability $\phi(x)$. Let S_0 be a sample drawn from $\mathcal{D}_{f,\phi}$. It follows that S_0 can be interpreted as a sample drawn from $\mathcal{D}_f$ from which some data is missing. Suppose now that we know that $\mathcal{D}_f$ comes from some class of distributions $\mathcal{C}$ (e.g. $\mathcal{D}_f$ is a Gaussian distribution) and that we have access to an algorithm $\mathcal{A}$ that can learn all distributions in this class in an optimal way (e.g. the maximum likelihood estimator for a Gaussian distribution). Applied to S_0, the algorithm $\mathcal{A}$ outputs a distribution f_0 that is no more optimal. Then, one could use mixture proportion estimation theory (Blanchard, Lee, and Scott, 2010; Sanderson and Scott, 2014; Scott, 2015) to estimate the proportion of missing data α_0. One could then estimate the distribution of the missing data in a non parametric fashion by looking at the empirical distribution $\alpha_0^{-1}(f_0 - (1-\alpha_0)\hat{m}_0)$ where $\hat{m}_0$ is the empirical distribution of S_0, and use this estimation to generate a new sample S_1 such that $S_0 \cup S_1$ is closer than S_0 to a "representative" sample of a distribution belonging to the class $\mathcal{C}$. This procedure could then be iterated until some stopping criterion. We think that this perspective on algebraic mixtures could lead to interesting applications in missing data scenario where the data is missing *not at random*. One example of such a scenario is the setting of *sample selection bias* (see e.g. Huang et al., 2006) where the sample available for training is not representative of the population of the test data because of a bias occurring during the sampling process. We think that the mask component of a distorted distribution could modelize such a bias. Another example is the problem of inference from presence-only data which is common in ecology. In this setting the data available consists only in locations where the species was observed, but the fact that the species was not observed in a location does not imply that it was not there. The mask component of a distorted distribution could prove useful to modelize this uncertainty on the presence of a species in a location where it was not observed.

Appendix

5.A Recognizable Probability Distributions on Strings

Probabilistic automatas and HMMs define the same family of probability distributions on strings (Dupont, Denis, and Esposito, 2005). All these distributions are recognizable but the converse is false (Denis and Esposito, 2008). The simplest counter examples can be built on a one-letter alphabet and a dimension equal to 3.

Let $\Sigma = \{a\}$ be a one-letter alphabet. We define a parametrized family of WAs by

$$\boldsymbol{\alpha} = \begin{bmatrix} \lambda \\ 0 \\ \sqrt{2}\lambda \end{bmatrix}, \mathbf{A} = \rho \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{\omega} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

where $\lambda > 0$ and $0 < \rho < 1$. Let p be the function computed by A. It can easily be seen that

$$p(a^n) = \rho^n \sqrt{2}\lambda(\cos(n\theta - \pi/4) + 1)) \geq 0$$

for all n. We also have

$$\sum_{w \in \Sigma^*} p(w) = \lambda \left(\frac{1 - \sqrt{2}\rho\cos(\theta - \pi/4)}{1 + \rho^2 - 2\rho\cos\theta} + \frac{\sqrt{2}}{1-\rho} \right)$$

and λ can always be chosen such that $\sum_{w \in \Sigma^*} p(w) = 1$, i.e. such that p is a probability distribution. It can be checked that the distribution p can be defined by a PA if and only if the set $\{\cos(n\theta - \pi/4) \ : \ n \geq 0\}$ is finite. It can then easily be seen that p can be defined by a PA if and only if $\theta/\pi \in \mathbb{Q}$. For example, if $\cos\theta = 3/5, \sin\theta = 4/5$, the corresponding distributions cannot be computed by a PA.

If $\rho = 0.5$, we have $\lambda = \frac{13}{6+26\sqrt{2}}$ and $\boldsymbol{\alpha} \simeq [0.304, 0, 0.430]^\top$. The construction described in Section 5.2.3 yields the distributions p^+ and p^- respectively defined by the following PAs:

$$\boldsymbol{\alpha}^+ = \begin{bmatrix} 0.4015 \\ 0 \\ 0.5985 \\ 0 \\ 0 \end{bmatrix}, \ \mathbf{A}^+ = \begin{bmatrix} 0.300 & 0 & 0 & 0 & 0.173 \\ 0.302 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.7 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.7 & 0.300 \end{bmatrix}, \ \boldsymbol{\omega}^+ = \begin{bmatrix} 0.527 \\ 0.398 \\ 0.5 \\ 0 \\ 0 \end{bmatrix},$$

$$\boldsymbol{\alpha}^- = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \ \mathbf{A}^- = \begin{bmatrix} 0.300 & 0 & 0 & 0.173 \\ 0.302 & 0.3 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 \\ 0 & 0 & 0.7 & 0.300 \end{bmatrix}, \ \boldsymbol{\omega}^- = \begin{bmatrix} 0.527 \\ 0.398 \\ 0 \\ 0 \end{bmatrix},$$

and the mixture parameters $\gamma^+ = 1.4364$ and $\gamma^- = 0.4364$.

If $\rho = 0.75$, the series p^+ and p^- computed by Lemma 7 do not converge. It is necessary to compute a WA $(\mathbb{R}^n, \boldsymbol{\alpha}', \{\mathbf{A}'\}, \boldsymbol{\omega}')$ computing p such that the function computed by the WA $(\mathbb{R}^n, |\boldsymbol{\alpha}'|, \{|\mathbf{A}'|\}, |\boldsymbol{\omega}'|)$ is convergent. This can be achieved using techniques

described in (Bailly and Denis, 2011). For example, we obtain the following WA

$$\boldsymbol{\alpha}' = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{A}' = \begin{bmatrix} 0 & 0.5675 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7125 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9566 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9753 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.8334 \\ 0.5662 & -0.1571 & 0 & 0 & 0 & 0.2750 \end{bmatrix}, \boldsymbol{\omega}' = \begin{bmatrix} 0.4325 \\ 0.2875 \\ 0.0434 \\ 0.0247 \\ 0.1666 \\ 0.3159 \end{bmatrix}$$

to which the construction described in Section 5.2.3 can be applied.

5.B Proof of Theorem 23

We will need the following results. The first one is a corollary of the *Sylvester's Law of Inertia*.

Lemma 10. *Let $\mathbf{Q} \in \mathbb{R}^{n \times n}$ be a symmetric real matrix. Suppose that there exists a non singular matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ and $w_1, \dots w_n \in \mathbb{R}$ such that $\mathbf{Q} = \mathbf{P}^\top \mathbf{D} \mathbf{P}$ where $\mathbf{D} = diag(w_1, \dots, w_n)$ is the diagonal matrix whose diagonal entries are $w_1, \dots, w_n$. Then, the number of negative eigenvalues of $\mathbf{Q}$ is equal to the number of negative coefficients w_i.*

Lemma 11. (Weyl's Inequality) *Let $\mathbf{A}$ and $\mathbf{B}$ be two $n \times n$ hermitian matrices. We have $\sigma_1(\mathbf{A}) + \sigma_i(\mathbf{B}) \leq \sigma_i(\mathbf{A} + \mathbf{B}) \leq \sigma_n(\mathbf{A}) + \sigma_i(\mathbf{B})$ for all $i \in [n]$, where $\sigma_i(\mathbf{M})$ denotes the ith smallest eigenvalue of $\mathbf{M}$.*

We use the previous two lemmas to prove the following one which will be key in proving Theorem 23.

Lemma 12. *Let $\{\mathbf{v}_i\}_{i=1}^k$ be a linearly dependent family of vectors of $\mathbb{R}^n$, where $n \geq k$, such that any of its subset of size $k-1$ is linearly independent. We consider the rank $k-1$ matrix $\mathbf{M} = \sum_{i=1}^k w_i \mathbf{v}_i \mathbf{v}_i^\top$ where $w_1, \cdots, w_k \neq 0$. Let l be the number of negative coefficients w_i. Then the first null eigenvalue of $\mathbf{M}$ is either the lth or the $(l+1)$-th smallest one.*

Proof. If $l = 0$, then $\mathbf{M}$ is positive semi-definite and $\sigma_1(\mathbf{M}) = 0$. If $l = k$, then $\mathbf{M}$ is negative semi-definite and $\sigma_k(\mathbf{M}) = 0$.

We suppose that $1 \leq l \leq k-1$. For $1 \leq j \leq k$, let $\mathbf{M}_j = \sum_{1 \leq i \neq j \leq k} w_i \mathbf{v}_i \mathbf{v}_i^\top$ and l_j be the number of negative coefficients in $\{w_i\}_{1 \leq i \neq j \leq k}$. Let V_j be the vector space spanned by $\{\mathbf{v}_1, \dots, \overline{\mathbf{v}}_j, \dots, \mathbf{v}_k\}$, where the notation $\overline{\mathbf{v}}_j$ means that $\mathbf{v}_j$ is omitted, and let $V_j^\perp$ be the orthogonal subspace of V_j. Let $\boldsymbol{\nu}_k, \cdots, \boldsymbol{\nu}_n$ be a linearly independent family of vectors in $V_j^\perp$ and let $\mathbf{P}$ be the non singular $n \times n$ matrix $[\mathbf{v}_1 \cdots, \overline{\mathbf{v}}_j, \cdots, \mathbf{v}_k, \boldsymbol{\nu}_k, \cdots, \boldsymbol{\nu}_n]^\top$. Clearly,

$$\mathbf{M}_j = \mathbf{P}^\top diag(w_1, \cdots, \overline{w}_j, \cdots, w_k, 0, \cdots, 0) \mathbf{P}$$

and therefore, from Lemma 10, l_j is the number of negative eigenvalues of $\mathbf{M}_j$.

For any $j \in [k]$, we consider the decomposition $\mathbf{M} = w_j \mathbf{v}_j \mathbf{v}_j^\top + \mathbf{M}_j$, sum of two hermitian matrices. The first summand is a rank one matrix whose only non null eigenvalue has the same sign as w_j, and the second has $k-1$ non zero eigenvalues, among which l_j are negative.

Let j be an index such that $w_j < 0$: from Weil's inequality $\sigma_i(\mathbf{M}) \leq 0 + \sigma_i(\mathbf{M}_j)$ for any $i \in [n]$. Since $\mathbf{M}_j$ has $l_j = l - 1$ negative eigenvalues, $\mathbf{M}$ has at least $l - 1$ negative eigenvalues.

Let j be such that $w_j > 0$: Weil's inequality gives $\sigma_i(\mathbf{M}) \geq 0 + \sigma_i(\mathbf{M}_j)$ for any $i \in [n]$, thus $\mathbf{M}$ has at least $k - l - 1$ positive eigenvalues, hence at most l negative ones.

Therefore, the first null eigenvalue of $\mathbf{M}$ must be either the lth or the $(l+1)$-th smallest one. □

Let $f(\mathbf{x}) = \sum_{i=1}^k w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i^2 \mathbf{I})$ be the PDF of the random vector $\mathbf{x}$, and let l be the number of negative weights w_i. We can now prove Theorem 23.

Theorem. *The average variance $\bar{\sigma}^2 = \sum_{i=1}^k w_i \sigma_i^2$ is an eigenvalue of the covariance matrix $\mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^\top]$. More precisely, if r is the number of negative eigenvalues of the matrix*

$$\mathbf{M} = \sum_{i=1}^k w_i (\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{x}]) \circ (\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{x}])$$

then $\bar{\sigma}^2$ is the $(r+1)$-th smallest eigenvalue of the covariance matrix. Furthermore, if l is the number of negative coefficients w_i, i.e. $l = |\{w_i : i \in [k],\ w_i < 0\}|$, then r is either equal to l or $l + 1$.

Let $\mathbf{v}$ be any unit-norm eigenvector corresponding to $\bar{\sigma}^2$ and let

$$\mathbf{m}_1 = \mathbb{E}[\mathbf{x}(\mathbf{v}^\top(\mathbf{x} - \mathbb{E}[\mathbf{x}]))^2], \qquad \mathbf{M}_2 = \mathbb{E}[\mathbf{x} \circ \mathbf{x}] - \bar{\sigma}^2 \mathbf{I}, \quad \textit{and}$$

$$\boldsymbol{\mathcal{M}}_3 = \mathbb{E}[\mathbf{x} \circ \mathbf{x} \circ \mathbf{x}] - \sum_{i=1}^d [\mathbf{m}_1 \circ \mathbf{e}_i \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{m}_1 \circ \mathbf{e}_i + \mathbf{e}_i \circ \mathbf{e}_i \circ \mathbf{m}_1]$$

where $\mathbf{e}_1, \cdots, \mathbf{e}_d$ is the coordinate basis of $\mathbb{R}^d$. Then,

$$\mathbf{m}_1 = \sum_{i=1}^k w_i \sigma_i^2 \boldsymbol{\mu}_i, \qquad \mathbf{M}_2 = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \ , \ \textit{and} \qquad \boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i.$$

Proof. Most of the proof of this theorem for usual Gaussian mixtures in (Hsu and Kakade, 2013) relies on the introduction of a discrete latent variable h: the sampling process is interpreted as first sampling h with $\mathbb{P}[h = i] = w_i$, and then sampling $\mathbf{x} = \boldsymbol{\mu}_h + \boldsymbol{z}_h$ where $\boldsymbol{z}_h$ is a multivariate Gaussian with mean $\mathbf{0}$ and covariance $\sigma_h^2 \boldsymbol{I}$. Allowing negative weights in the mixture, we cannot use the same strategy, but it will be sufficient to observe that $\mathbb{E}[g(\mathbf{x})] = \sum_{i=1}^k w_i \, \mathbb{E}[g(\boldsymbol{\mu}_i + \mathbf{z}_i)]$ for any function g, which is a direct consequence of the linearity of the expectation.

First, we need to identify the position of $\bar{\sigma}^2$ in the covariance matrix. Let $\bar{\boldsymbol{\mu}} = \mathbb{E}[\mathbf{x}] = \sum_{i=1}^k w_i \boldsymbol{\mu}_i$. The covariance matrix of $\mathbf{x}$ is

$$\mathbb{E}[(\mathbf{x} - \bar{\boldsymbol{\mu}}) \circ (\mathbf{x} - \bar{\boldsymbol{\mu}})] = \sum_{i=1}^k w_i (\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}) \circ (\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}) + \bar{\sigma}^2 \mathbf{I}.$$

Since the $\boldsymbol{\mu}_i$'s are linearly independent, $F = \{\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}\}_{i=1}^k$ is a linearly dependent family of vectors of $\mathbb{R}^n$ such that any of its subset of size $k-1$ is linearly independent. It follows from Lemma 12 that 0 is either the lth or $(l+1)$-th smallest eigenvalue of the matrix

$\sum_{i=1}^k w_i(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}) \circ (\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}})$, which implies that $\bar{\sigma}^2$ is the corresponding eigenvalue in the covariance matrix.

Note that the strict separation of $\bar{\sigma}^2$ from the other eigenvalues in the covariance matrix implies that every eigenvector corresponding to $\bar{\sigma}^2$ is in the null space of $\sum_{i=1}^k w_i(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}) \circ (\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}})$, hence $\mathbf{v}^\top(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}) = 0$ for all $i \in [k]$.

We now express $\mathbf{m}_1$, $\mathbf{M}_2$ and $\boldsymbol{\mathcal{M}}_3$ in terms of the parameters w_i, σ_i^2 and $\boldsymbol{\mu}_i$. First,

$$\begin{aligned}
\mathbf{m}_1 &= \mathbb{E}[\mathbf{x}(\mathbf{v}^\top(\mathbf{x} - \mathbb{E}[\mathbf{x}]))^2] \\
&= \sum_{i=1}^k w_i \, \mathbb{E}[(\boldsymbol{\mu}_i + \mathbf{z}_i)(\mathbf{v}^\top(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}} + \mathbf{z}_i))^2] \\
&= \sum_{i=1}^k w_i \, \mathbb{E}[(\boldsymbol{\mu}_i + \mathbf{z}_i)(\mathbf{v}^\top \mathbf{z}_i)^2] \;\; = \sum_{i=1}^k w_i \sigma_i^2 \boldsymbol{\mu}_i.
\end{aligned}$$

Next, since $\mathbb{E}[\mathbf{z}_i \circ \mathbf{z}_i] = \sigma_i^2 \mathbf{I}$ for all $i \in [k]$, we have

$$\begin{aligned}
\mathbf{M}_2 &= \mathbb{E}[\mathbf{x} \circ \mathbf{x}] - \bar{\sigma}^2 \mathbf{I} \\
&= \sum_{i=1}^k w_i \, \mathbb{E}[(\boldsymbol{\mu}_i + \mathbf{z}_i) \circ (\boldsymbol{\mu}_i + \mathbf{z}_i)] - \bar{\sigma}^2 \mathbf{I} \\
&= \sum_{i=1}^k w_i (\boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i + \mathbb{E}[\mathbf{z}_i \circ \mathbf{z}_i]) - \bar{\sigma}^2 \mathbf{I} = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i.
\end{aligned}$$

Finally, writing z_{ij} for the jth component of the vector $\mathbf{z}_i$, we have

$$\begin{aligned}
\sum_{i=1}^k w_i \, \mathbb{E}[\boldsymbol{\mu}_i \circ \mathbf{z}_i \circ \mathbf{z}_i] &= \sum_{i=1}^k w_i \sum_{p=1}^n \sum_{q=1}^n \mathbb{E}[z_{ip} z_{iq}] \boldsymbol{\mu}_i \circ \mathbf{e}_p \circ \mathbf{e}_q \\
&= \sum_{i=1}^k w_i \sigma_i^2 \sum_{j=1}^n \boldsymbol{\mu}_i \circ \mathbf{e}_j \circ \mathbf{e}_j \\
&= \sum_{j=1}^n \mathbf{m}_1 \circ \mathbf{e}_j \circ \mathbf{e}_j,
\end{aligned}$$

where we used the fact that $\mathbb{E}[z_{ip} z_{iq}] = \delta_{pq} \sigma_i^2$ for all $i \in [k]$, $p, q \in [n]$. Using the same derivation, we have $\sum_{i=1}^k w_i \, \mathbb{E}[\mathbf{z}_i \circ \boldsymbol{\mu}_i \circ \mathbf{z}_i] = \sum_{j=1}^n \mathbf{e}_j \circ \mathbf{m}_1 \circ \mathbf{e}_j$ and $\sum_{i=1}^k w_i \, \mathbb{E}[\mathbf{z}_i \circ \mathbf{z}_i \circ \boldsymbol{\mu}_i] = \sum_{j=1}^n \mathbf{e}_j \circ \mathbf{e}_j \circ \mathbf{m}_1$. Hence,

$$\begin{aligned}
\mathbb{E}[\mathbf{x}^{\circ 3}] &= \sum_{i=1}^k w_i \left(\boldsymbol{\mu}_i^{\circ 3} + \mathbb{E}[\boldsymbol{\mu}_i \circ \mathbf{z}_i \circ \mathbf{z}_i] + \mathbb{E}[\mathbf{z}_i \circ \boldsymbol{\mu}_i \circ \mathbf{z}_i] + \mathbb{E}[\mathbf{z}_i \circ \mathbf{z}_i \circ \boldsymbol{\mu}_i] \right) \\
&= \sum_{i=1}^k w_i \boldsymbol{\mu}_i^{\circ 3} + \sum_{j=1}^n (\mathbf{m}_1 \circ \mathbf{e}_j \circ \mathbf{e}_j + \mathbf{e}_j \circ \mathbf{m}_1 \circ \mathbf{e}_j + \mathbf{e}_j \circ \mathbf{e}_j \circ \mathbf{m}_1)
\end{aligned}$$

and $\boldsymbol{\mathcal{M}}_3 = \sum_{i=1}^k w_i \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i \circ \boldsymbol{\mu}_i$. □

5.C Proof of Theorem 24

Theorem. *Let $\boldsymbol{\mathcal{T}} \in (\mathbb{C}^d)^{\otimes 3}$ have a pseudo-orthonormal decomposition $\boldsymbol{\mathcal{T}} = \sum_{i=1}^k z_i \boldsymbol{\nu}_i^{\circ 3}$, and let T be the mapping defined by $T(\boldsymbol{\theta}) = \boldsymbol{\mathcal{T}}(I, \boldsymbol{\theta}, \boldsymbol{\theta})$ for any $\boldsymbol{\theta} \in \mathbb{C}^d$. Let $\boldsymbol{\theta}_0 \in \mathbb{C}^d$, suppose that $|z_1.\boldsymbol{\nu}_1^\top \boldsymbol{\theta}_0| > |z_2.\boldsymbol{\nu}_2^\top \boldsymbol{\theta}_0| \geq \cdots \geq |z_k.\boldsymbol{\nu}_k^\top \boldsymbol{\theta}_0| > 0$. For $t = 1, 2, \cdots$, define*

$$\boldsymbol{\theta}_t = \frac{T(\boldsymbol{\theta}_{t-1})}{[T(\boldsymbol{\theta}_{t-1})^\top T(\boldsymbol{\theta}_{t-1})]^{\frac{1}{2}}} \quad \text{and} \quad \lambda_t = \boldsymbol{\mathcal{T}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_t, \boldsymbol{\theta}_t) \tag{5.6}$$

where we assume that $\boldsymbol{\theta}_0$ is such that $T(\boldsymbol{\theta}_t)^\top T(\boldsymbol{\theta}_t) \neq 0$ for all t. Then, $\boldsymbol{\theta}_t \to \pm \boldsymbol{\nu}_1$ and $\lambda_t \to \pm z_1$ as $t \to \infty$.

More precisely, let

$$M = \max\left\{1, \frac{|z_1|^2}{|z_i|^2}, |z_1|\frac{\|\nu_i\|}{|z_i|} : i \in [k]\right\} \quad \text{and} \quad \varepsilon_t = kM\left|\frac{z_2.\boldsymbol{\nu}_2^\top \boldsymbol{\theta}_0}{z_1.\boldsymbol{\nu}_1^\top \boldsymbol{\theta}_0}\right|^{2^t}.$$

Then for all $t \geq 2$ such that $\varepsilon_t < \frac{1}{2}$, we have

$$|e_t f_t \lambda_t - z_1| \leq 7|z_1|\varepsilon_t \quad \text{and} \quad \|e_t f_t \boldsymbol{\theta}_t - \boldsymbol{\nu}_1\| \leq \varepsilon_t\left(||\boldsymbol{\nu}_1|| + \sqrt{2}\right),$$

where $(e_t)_t$ and $(f_t)_t$ are two sequences defined in the proof and taking their values in $\{-1, 1\}$.

Proof. Let us first define the square root of a complex number $z = re^{i\theta}$, where $-\pi < \theta \leq \pi$ and $r \geq 0$, by $z^{1/2} = r^{1/2}e^{i\frac{\theta}{2}}$, and note that $z/(z^2)^{1/2} = z^{-1}(z^2)^{1/2} = 1$ if $-\pi/2 < \theta \leq \pi/2$ and -1 otherwise.

Now, let $c_i = \boldsymbol{\nu}_i^\top \boldsymbol{\theta}_0$ for $i \in [k]$, $\widetilde{\boldsymbol{\theta}}_0 = \boldsymbol{\theta}_0$, $\widetilde{\boldsymbol{\theta}}_t = T(\widetilde{\boldsymbol{\theta}}_{t-1})$, and $\rho_t = (\widetilde{\boldsymbol{\theta}}_t^\top \widetilde{\boldsymbol{\theta}}_t)^{\frac{1}{2}}$ for all $t \geq 1$. Check by induction on t that, for all $t \geq 1$,

$$\widetilde{\boldsymbol{\theta}}_t = \sum_{i=1}^k z_i^{2^t-1} c_i^{2^t} \boldsymbol{\nu}_i. \tag{5.7}$$

Let $e_t = \rho_{t+1}\rho_t^{-2} / \left(\rho_t^{-4}\rho_{t+1}^2\right)^{\frac{1}{2}}$, note that $e_t = \pm 1$, and check by induction that, for all $t \geq 2$,

$$\boldsymbol{\theta}_t = e_t \frac{\widetilde{\boldsymbol{\theta}}_t}{\rho_t}. \tag{5.8}$$

Let $\alpha_t = \rho_t^{-1} z_1^{2^t-1} c_1^{2^t}$. Using Eq. (5.7) and Eq. (5.8), we obtain

$$e_t\lambda_t = \rho_t^{-3}\sum_{i=1}^k z_i(z_i^{2^t-1}c_i^{2^t})^3 = \alpha_t^3 \sum_{i=1}^k \frac{z_1^3}{z_i^2}\left(\frac{z_i c_i}{z_1 c_1}\right)^{3\cdot 2^t} = \alpha_t^3 z_1\left[1 + \sum_{i=2}^k \frac{z_1^2}{z_i^2}\left(\frac{z_i c_i}{z_1 c_1}\right)^{3\cdot 2^t}\right], \text{ and}$$

$$e_t\boldsymbol{\theta}_t = \rho_t^{-1}\sum_{i=1}^k z_i^{2^t-1}c_i^{2^t}\boldsymbol{\nu}_i \quad = \alpha_t \sum_{i=1}^k \frac{z_1}{z_i}\left(\frac{z_i c_i}{z_1 c_1}\right)^{2^t}\boldsymbol{\nu}_i = \alpha_t\left[\boldsymbol{\nu}_1 + \sum_{i=2}^k \frac{z_1}{z_i}\left(\frac{z_i c_i}{z_1 c_1}\right)^{2^t}\boldsymbol{\nu}_i\right].$$

It can easily be checked that

$$\left|\sum_{i=2}^{k} \frac{z_1^2}{z_i^2}\left(\frac{z_i c_i}{z_1 c_1}\right)^{3\cdot 2^t}\right| \leq \varepsilon_t \quad \text{and} \quad \left\|\sum_{i=2}^{k} \frac{z_1}{z_i}\left(\frac{z_i c_i}{z_1 c_1}\right)^{2^t} \boldsymbol{\nu}_i\right\| \leq \varepsilon_t.$$

Moreover, it can be checked that

$$\alpha_t^{-1} = \frac{(\widetilde{\boldsymbol{\theta}}_t^\top \widetilde{\boldsymbol{\theta}}_t)^{1/2}}{z_1^{2^t-1} c_1^{2^t}} = f_t \left(\frac{\widetilde{\boldsymbol{\theta}}_t^\top \widetilde{\boldsymbol{\theta}}_t}{(z_1^{2^t-1} c_1^{2^t})^2}\right)^{1/2} = f_t \left[1 + \sum_{i=2}^{k} \frac{z_1^2}{z_i^2}\left(\frac{z_i c_i}{z_1 c_1}\right)^{2^{t+1}}\right]^{1/2}$$

where $f_t = (z_1^{2^t-1} c_1^{2^t})^{-1} \left((z_1^{2^t-1} c_1^{2^t})^2\right)^{\frac{1}{2}} = \pm 1$. Using the hypothesis $\varepsilon_t < \frac{1}{2}$ and making use of Lemma 13 below, it follows that

$$|\alpha_t| \leq \sqrt{2}, \quad |f_t \alpha_t - 1| \leq \varepsilon_t \quad \text{and} \quad |f_t \alpha_t^3 - 1| \leq 4\varepsilon_t.$$

Finally, combining these inequalities, we obtain

$$|e_t f_t \lambda_t - z_1| \leq 7|z_1|\varepsilon_t \quad \text{and} \quad \|e_t f_t \boldsymbol{\theta}_t - \boldsymbol{\nu}_1\| \leq \varepsilon_t \left(||\boldsymbol{\nu}_1|| + \sqrt{2}\right).$$

□

Lemma 13. *Let $k > 0$ and $z \in \mathbb{C}$ such that $|z| < 1/2$. Then,*

$$|(1+z)^{-k} - 1| \leq 2|z|(2^k - 1).$$

In particular,

$$|(1+z)^{-1/2} - 1| \leq |z| \text{ and } |(1+z)^{-3/2} - 1| \leq 4|z|.$$

Proof. Let $f(z) = (1+z)^{-k} - 1$ with $k > 0$ and $|z| < 1/2$. The derivative of f is given by $f'(z) = -k(1+z)^{-(k+1)}$. Let $\gamma : [0,1] \mapsto \mathbb{C}$ be defined by $\gamma(t) = tz$. We have

$$\begin{aligned}(1+z)^{-k} - 1 &= \int_\gamma f'(y)dy = \int_0^1 f'(\gamma(t))\gamma'(t)dt \\ &= -kz \int_0^1 (1+tz)^{-(k+1)} dt.\end{aligned}$$

Therefore,

$$\begin{aligned}|(1+z)^{-k} - 1| &\leq k|z| \int_0^1 (1-t|z|)^{-(k+1)} dt \\ &= [(1-t|z|)^{-k}]_0^1 = (1-|z|)^{-k} - 1 \\ &\leq 2|z|(2^k - 1).\end{aligned}$$

Indeed, let $g(x) = (1-x)^{-k} - 1 - 2x(2^k - 1)$. It can be checked that $g(0) = g(1/2) = 0$ and that g is convex on $[0, 1/2]$. □

Conclusion and Perspectives

In this thesis, we presented our contributions to address four distinct problems related to different topics including weighted automata theory, model reduction, structured regression and density estimation. Even though these problems are in essence mostly unrelated, they were all initially motivated from a machine learning perspective with the idea to leverage tensors and multilinear algebra tools in mind. Before trying to give a broader perspective on the works gathered in this manuscript, we first summarize our contributions and present open problems and future lines of research.

Chapter 2. Our initial motivation for this work was to design learning algorithms for functions defined on graphs. Inspired by the mathematical elegance of the spectral method to learn weighted automata on strings and trees introduced in (Bailly, Denis, and Ralaivola, 2009; Balle et al., 2014; Hsu, Kakade, and T. Zhang, 2008; Bailly, Habrard, and Denis, 2010), our first step was to try to extend the algebraic definition of weighted automata on which the spectral method relies to graphs. Using tensor networks to give a straightforward insight on the connection between the computation of a weighted automaton and the structure of its input, we revealed a natural extension of weighted automata to labeled graphs: Graphs Weighted Models (GWMs). In order to legitimate this model as a valid and promising computational model on graphs, we studied on which conditions it satisfies desirable properties such as closure under basic operations and recognizability of finite support functions.

In the last section of this chapter, we came back to our original motivation and considered the problem of learning GWMs defined over the simple family of circular strings. Even though we showed that it is possible to learn a function computed by a GWM on circular strings using e.g. the classical spectral method for string weighted automata, recovering the parameters of a GWM computing this function is a far less trivial task. We proposed a learning algorithm relying on tensor decomposition techniques that directly tries to learn a GWM from a set of labeled examples, but it relies on the assumption that the GWM used to generate the data is of full rank (i.e. the matrices associated with each symbol generate the full matrix algebra). The question of whether the method we propose could be extended to the case where the GWM is not of full rank remains open. Another open problem is the following: on which conditions a real valued function computed by a GWM with complex coefficients can be computed by a GWM with real coefficients? We conjectured in this chapter that a sufficient condition would be to consider GWMs defined on a family of in-out graphs.

There are several general lines of research that we find both promising and exciting as continuations of this work (of course the ideas listed below are not exhaustive). First, some real-valued functions defined on graphs that are particularly relevant to

machine learning are probability distributions. A study of GWMs from this perspective would be of great interest and could start with some simple problems: how can we construct GWMs computing probability distributions? For circular strings, choosing all the matrices $\{\mathbf{M}^\sigma\}_{\sigma\in\Sigma}$ of the GWM with non negative coefficients and such that $\mathrm{Tr}[(\mathbf{I} - \sum_\sigma \mathbf{M}^\sigma)^{-1}] = 1$ would be sufficient, but there certainly exist GWMs computing probability distributions that do not satisfy these properties. And what about GWMs defined over other families of graphs? Another interesting question naturally arises: given a GWM computing a probability distribution how can we draw samples from this distribution? We plan to investigate these questions in the near future.

A second line of research that we already mentioned in the conclusion of this chapter is to design learning algorithms for richer families of graphs than the one of circular strings. We think that since the family of rooted in-out graphs exhibits a lot of *good* properties, studying the learnability of GWMs defined over this family is a reasonable long term objective. A first step that we are currently investigating is to derive a learning algorithm for picture weighted automata, i.e. GWMs defined on 2d-words. Studying the notion of minimality and the equivalence problem for GWMs is also a very fundamental direction that deserves interest and that is certainly connected to the question of learning GWMs (see Maruvsic and Worrell (2015) for connections between minimization, equivalence and learning for weighted tree automata).

Lastly, this chapter presented an intrinsic connection between tensor networks and weighted automata and we strongly believe that tensor networks are a powerful tool that deserves a great attention from the machine learning community. Besides their ability to visually depict complex operations on tensors, tensor networks are at the core of powerful tensor decomposition techniques such as hierarchical Tucker and tensor train decompositions. Moreover, they have been used by the physics community for some time now and several optimization algorithms have been developed and are available. This last line of research is quite general and could be summarized as developing efficient machine learning algorithms relying on the power of tensor networks. The very recent work presented in (Stoudenmire and Schwab, 2016) is a good illustration of what kind of ideas we would like to develop in the future: in this paper the authors adapt algorithms for optimizing tensor networks to a supervised learning setting using matrix product trains (i.e. tensor train format) to parameterize the model.

Chapter 3. In this chapter we tackled a model reduction problem by proposing a principled way to reduce the size of a weighted tree automaton. This problem is deeply connected to the spectral learning method for weighted automata. Indeed, in the case of a weighted automaton computing a probability distribution, the spectral method consists in first estimating the Hankel matrix from a finite sample drawn from the distribution and then applying basic linear algebra operations on this estimate to compute a weighted automaton approximating the original function. Given a target number of states n, the spectral method boils down to finding a low rank approximation of the estimated Hankel matrix (of rank n) and then performing a linear regression to recover the parameters of a model whose Hankel matrix is close to the low rank projection of the estimate. In practice the Hankel matrix will have a finite size, say $M \times M$, and will always be of full rank. Roughly speaking, one can think of the empirical distribution of the observed data as a function computed by an automaton $\hat{A}$ with M states and

learning is achieved by trying to find an automaton with n states computing a function that is close to the empirical distribution, i.e. a good approximation of the automaton $\hat{A}$. Thus learning a weighted automaton from training data can be seen as performing an approximate minimization of this (fictive) empirical automaton $\hat{A}$.

The method we proposed generalizes the method proposed in (Balle, Panangaden, and Precup, 2015) and the principal difficulty resided in the approximation of the Gram matrices associated with a rank factorization of the Hankel matrix: while these matrices were defined by a linear system of equations having an analytic solution in the string case, they are defined by a system of polynomial equations in the tree case. Using results from fixed point theory we designed an efficient algorithm to approximate these Gram matrices in the tree case. This shift from a linear system to a polynomial one comes from the fact that the computations of a WA boil down to matrix products whereas the WTA computations relies on higher order tensor contractions. This shows once again that problems often get considerably more difficult when going from matrices to higher order tensors.

In our experimental studies we observed that our method obtains very good results in terms of perplexity and ℓ_2 distance but that it does not perform as well on the parsing task. In the future, we would like to better understand this behavior and hopefully find possible modifications to improve the parsing accuracy. Promising directions include trying to preserve sparsity throughout the approximate minimization process and combining our method with existing ones. In (Narayan and Cohen, 2016), the authors propose to use search algorithms to optimize spectral learning of latent probabilistic context free grammar for parsing. Even though in their case the optimization concerns the estimation of the number of latent states, a similar idea of looking for optimizations specifically designed for the parsing task could be an interesting direction.

From a theoretical point of view, the approximation bound we provided seems quite loose (especially in comparison with the bound obtained in the string case) and we will keep on trying to get better theoretical guarantees in the future. Since directly extending the proof techniques used in (Balle, Panangaden, and Precup, 2015) does not seem possible, we think that getting a bound on the ℓ_2 distance between the original function and its approximation could shed a new light on the string case (indeed WTAs are a generalization of string weighted automata and a bound in the tree case would directly imply one for the string case).

Another line of research we would like to investigate is related to representation learning. The algebraic representation of a weighted automaton naturally induces an embedding of the inputs (strings or trees) into a finite dimensional vector space. In the case of trees, WTA induces a joint embedding space for trees and contexts and we think that this representation space could be useful to tackle machine learning tasks. For example, would it be possible to learn a binary classifier on the space of trees by first finding an appropriate WTA, and then learning a linear classifier in the representation space induced by this WTA? Note that we already considered a related problem in a prior work[d] (Rabusseau and Denis, 2014) where we showed that this representation space can be leveraged to find a tree maximizing a given WTA.

[d] This work was realized during my Master degree and has not been included in this thesis.

Chapter 4. In this chapter, we considered a regression task for tensor structured outputs. An example of such a task is the spatio-temporal forecasting task where the outputs consist in different variables measured in different locations; in this example the outputs can naturally be structured as 2-dimensional arrays. The algorithm we proposed to tackle this task (HOLRR) is a generalization of the reduced rank regression method that learns a linear vector-valued function by minimizing the squared error on the training data while enforcing a low rank constraint on the regression matrix. The low rank constraint encourages collaboration between the regression functions associated to the different components of the vector output. In order to take into account the tensor structure of the outputs we replaced this matrix rank constraint by a multilinear rank constraint on the regression tensor. At this point the problem we consider is the minimization of a least square criterion under a multilinear rank constraint. This problem had already been considered in (Romera-Paredes et al., 2013) in the context of multilinear multitask learning where the authors proposed two approaches: the first one relies on a convex regularization of the problem using the trace norm of the matricizations of the regression tensor, and the second one directly tackles the non-convex problem using an alternating gradient descent method. While the second approach led to better predictive performances it is computationally very expensive and is only guaranteed to return a local minimizer of the objective function (without approximation guarantees). Our primary objective for this work was to directly tackle the non-convex problem (without using a convex relaxation) and to provide an efficient algorithm while still being able to provide theoretical guarantees. We proposed a method that achieves this objective: HOLRR offers approximation and statistical guarantees and our complexity analysis and experiments showed that it is an efficient algorithm. We managed to design such an algorithm by reducing the initial problem to a multilinear subspace identification problem for which we provided a sound approximation strategy. However, these achievements came with a price: HOLRR is limited to the squared error loss, and in the case where both inputs and outputs are tensors (such as the spatio-temporal forecasting task) it is not yet able to leverage the tensor structure of the inputs. Furthermore, the formulation of the problem we obtain with the squared error loss assume that all the components of the output tensors are known at training time, which is not the case in the multilinear multitask setting. In this setting, the inputs are feature vectors (e.g. characteristics of a restaurant) and the components of the outputs correspond to different tasks that can be naturally arranged as a tensor (e.g. ratings from three different critics on three different criteria such as food quality, host and price); for each input in the training data only a few tasks may been known (e.g. only two critics went to one of the restaurants).

A direct continuation of this work would consist in trying to circumvent these limitations: extend HOLRR to other losses (in particular the weighted square loss which would allow us to apply HOLRR in the multilinear multitask setting), and extend HOLRR to the setting where the regression tensor is of low multilinear rank on both the inputs and outputs modes. Trying to derive an alternative extension of HOLRR to the non linear setting by taking a multilinear perspective on the operator-valued kernel framework is also an interesting direction. Concerning the theoretical analysis of HOLRR, we want to go further than the consistency result we provided in this chapter by deriving sample complexity bounds, and we would like to show that HOLRR provides better convergence rates than the regularized least square or the reduced rank estimators when the true

regression tensor is of low multilinear rank (which is verified by simulations).

A more general line of research we want to pursue is related to the notions of rank and regularization. We chose to use a multilinear rank constraint in this chapter mainly because it allowed us to obtain good approximation guarantees inspired from the ones provided by the HOSVD algorithm. However, other choices would have been possible: a constraint on the CP-rank, on the rank induced by a tensor train decomposition, or more generally by enforcing any kind of structural constraints on the regression tensor (e.g. hierarchical tucker, tensor ring decomposition (Zhao et al., 2016), etc.). We plan to develop a theoretical study of the regularization power of these different notions of rank. To begin with, the generalization bound provided in this chapter could easily be adapted to other notions of ranks using the same techniques based on the notion of pseudo-dimension. Assuming that we manage to design efficient algorithms to tackle the problem using other forms of low rank constraints, we would like to perform an empirical study of these different tensor rank regularizations on real world data sets. This may confirm our intuition that different kinds of tensor rank regularization could be more adapted to different data sets.

Chapter 5. In this chapter, we introduced the notion of *algebraic mixture models*: probability distributions that can be expressed as a weighted sum of distributions $f = w_1 f_1 + \cdots + w_k f_k$ where some of the weights w_i may be negative. The starting point of this work was in some sense the fundamental relation between probabilistic automata (PA) and weighted automata (WA) that we showed in this chapter: any WA can be expressed as an algebraic mixture of two PAs. It is this uncanny relation that led us to study this unusual model of distributions. Unlike classical mixture models that have a natural interpretation in terms of latent variable models, algebraic mixtures cannot be interpreted from this point of view. However, we showed that algebraic mixtures can be seen as distorted distributions where the positive components of the mixture act as an underlying *true* distribution, while the negative components act as a mask that will retain samples drawn from the positive components. In this sense algebraic mixtures could be relevant to missing data scenarios where the data is missing not at random, such as selection bias. This is one of the future line of research that we want to develop. In particular we would like to investigate other learning approaches than the one proposed in this chapter, which would try to leverage the idea that data is missing from the training set. One such idea would be to estimate the distribution of this missing data, which should be possible if we have some prior knowledge on the underlying true distribution.

The fact that algebraic mixtures are not latent variable models prevented us from applying the Expectation-Maximization algorithm that is commonly used to learn classical mixture models. However, the tensor method of moments proposed in (Anandkumar et al., 2014) to learn mixture models did not fundamentally rely on the fact that mixture models are latent variable models, and thus looked like a pertinent starting point to design a learning algorithm for algebraic mixtures. We extended their method to the case of algebraic mixtures and we showed in particular how this approach can be applied to the problem of learning algebraic mixtures of spherical Gaussians. However, the simulation study we provided in this chapter for the problem of learning an algebraic mixture of two 2-dimensional spherical Gaussians showed that the tensor method of moments requires the size of the training sample to be very large to outperform baselines such

as the classical Gaussian Mixture Model. This is comparable to the spectral learning method for weighted automata that often struggles to outperform the EM algorithm. In the future, we would like to try to better understand this behavior in order to improve the quality of the estimator returned by spectral methods on small training samples. One reason that may cause this lack of robustness may be that spectral methods rely on the estimation of eigenvectors that are sometimes associated with very small eigenvalues, which is known to be sensitive to noise. Exploring techniques relying on simultaneous diagonalization or triangularization such as the ones proposed in (Kuleshov, Chaganty, and Liang, 2015) and (Colombo and Vlassis, 2016) could be a promising direction to try to improve the spectral method for learning weighted automata, which we plan to pursue.

To conclude, we will now briefly try to give a broader perspective on the works gathered in this manuscript. We started by observing that classical methods and models in machine learning often rely on linear algebra tools and that extending these methods/models to the multilinear setting is both a challenging and rewarding task that is currently actively pursued by the machine learning community. Tensors and multilinear algebra are indeed powerful tools and independent research on these topics is also very active in other communities (e.g. physics, biology, neuro-sciences, etc.). Trying to gather and give a unifying view of the methods developed in these communities and of their relevance to machine learning is a vast and promising line of research. For example, the algorithms used in physics to manipulate tensor networks could be of great interest to the machine learning community. However, we saw that the power of tensors and multilinear algebra comes with a price: most problems related to tensors are NP-hard (Hillar and Lim, 2013), and going from matrices to tensors often implies going from problems with a closed form solution to intractable ones for which only approximate solutions can be found. Nonetheless, the recent advances on tensor decomposition techniques and the availability of more and more computing resources make it nowadays possible to design efficient algorithms relying on tensors and on fundamental tools from multilinear algebra.

The works presented in this thesis offered different perspectives on these considerations. We showed in Chapter 2 that tensor networks, which are mainly used in physics and numerical analysis, can be relevant to formal language theory and machine learning by proposing a natural computational model on graphs fundamentally relying on the tensor network formalism. In Chapter 3, we considered a model reduction technique for weighted tree automata and we showed that even though going from strings to trees (i.e. from matrices to tensors) implied going from a linear to a polynomial system of equations, approximation algorithms can be designed to still address the problem in an efficient way. Similarly in Chapter 4 we obtained an efficient approximation algorithm for the NP-hard tensor counterpart to the reduced rank regression problem. We also exhibited in this chapter the benefits of using tensor rank regularization instead of matrix rank regularization when the problem at hand involves tensor structured data. Finally in Chapter 5, we showed that tensor methods can be used to derive learning schemes for algebraic models (e.g. mixtures with negative weights) that cannot be naturally handled with traditional learning algorithms that rely on e.g. positiveness assumptions such as the Expectation-Maximization algorithm. Overall, we showed that tensors are powerful tools that can be used to design expressive computational models, that they allow one to

take into account the underlying structure of data arising in multivariate analysis, and that their algebraic nature can prove useful to elaborate learning schemes for models relying on algebraic structures (such as algebraic mixtures or graph weighted models on circular strings).

More generally, we think that a lot remains to be done in the exploration of the fruitful relations between tensors and machine learning, and that machine learning will greatly benefit from using tensors and multilinear algebra in the following years. Understanding the regularization ability of the various notions of tensor ranks, designing efficient learning algorithms relying on tensor decomposition techniques and tensor networks, combining the powers of tensors and kernels to bring non linearity in multilinear models, leveraging tensor methods to derive learning schemes relying on algebraic geometry, using the quantum physics theoretical tools relying on tensors to develop quantum machine learning algorithms, etc., these are all exciting directions that may bring further the power of tensors and multilinear algebra to machine learning. And who knows, the next revolution in machine learning may come from tensors.

Bibliography

[1] Jonathan L Alperin. *Local representation theory: Modular representations as an introduction to the local representation theory of finite groups*. Vol. 11. Cambridge University Press, 1993 (cit. on p. 54).

[2] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. "Kernels for vector-valued functions: A review". In: *Foundations and Trends in Machine Learning* 4.3 (2012), pp. 195–266 (cit. on p. 99).

[3] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. "Tensor decompositions for learning latent variable models." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 2773–2832 (cit. on pp. 3, 6, 13, 117–120, 127–130, 133, 146).

[4] Theodore Wilbur Anderson. "Estimating linear restrictions on regression coefficients for multivariate normal distributions". In: *The Annals of Mathematical Statistics* (1951), pp. 327–351 (cit. on p. 90).

[5] Dana Angluin. "Local and global properties in networks of processors". In: *Proceedings of the twelfth annual ACM symposium on Theory of computing*. ACM. 1980, pp. 82–93 (cit. on p. 44).

[6] Parvaneh Babari and Manfred Droste. "A Nivat theorem for weighted picture automata and weighted MSO logic". In: *Language and Automata Theory and Applications*. Springer, 2015, pp. 703–715 (cit. on pp. 39, 88).

[7] Mohammad Taha Bahadori, Rose Yu, and Yan Liu. "Fast multivariate spatio-temporal analysis via low rank tensor learning". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3491–3499 (cit. on pp. 87, 89, 96, 104, 108).

[8] Timothy L Bailey and Charles Elkan. "Fitting a mixture model by expectation maximization to discover motifs in bipolymers". In: (1994) (cit. on p. 117).

[9] Raphaël Bailly and François Denis. "Absolute convergence of rational series is semi-decidable". In: *Information and Computation* 209.3 (2011), pp. 280–295 (cit. on pp. 126, 137).

[10] Raphaël Bailly, François Denis, and Guillaume Rabusseau. "Recognizable series on hypergraphs". In: *Language and Automata Theory and Applications*. Springer, 2015, pp. 639–651 (cit. on pp. 25, 35).

[11] Raphaël Bailly, François Denis, and Liva Ralaivola. "Grammatical inference as a principal component analysis problem". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 33–40 (cit. on pp. 25, 62, 142).

[12] Raphaël Bailly, Amaury Habrard, and François Denis. “A spectral approach for probabilistic grammatical inference on trees”. In: *Algorithmic Learning Theory*. Springer. 2010, pp. 74–88 (cit. on pp. 25, 62, 81, 142).

[13] Pierre Baldi and Kurt Hornik. “Neural networks and principal component analysis: Learning from examples without local minima”. In: *Neural networks* 2.1 (1989), pp. 53–58 (cit. on p. 89).

[14] Borja Balle, Xavier Carreras, Franco M Luque, and Ariadna Quattoni. “Spectral learning of weighted automata”. In: *Machine learning* 96.1-2 (2014), pp. 33–63 (cit. on pp. 25, 62, 68, 142).

[15] Borja Balle, Prakash Panangaden, and Doina Precup. “A canonical form for weighted automata and applications to approximate minimization”. In: *Logic in Computer Science (LICS), 2015 30th Annual ACM/IEEE Symposium on*. IEEE. 2015, pp. 701–712 (cit. on pp. 5, 62, 63, 68, 70, 76, 77, 144).

[16] Leonard E Baum and John Alonzo Eagon. “An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology”. In: *Bull. Amer. Math. Soc* 73.3 (1967), pp. 360–363 (cit. on p. 26).

[17] Mikhail Belkin and Kaushik Sinha. “Polynomial learning of distribution families”. In: *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE. 2010, pp. 103–112 (cit. on p. 117).

[18] Richard A Berk. “An introduction to sample selection bias in sociological data”. In: *American Sociological Review* (1983), pp. 386–398 (cit. on pp. 116, 124).

[19] Jean Berstel and Christophe Reutenauer. “Recognizable formal power series on trees”. In: *Theoretical Computer Science* 18.2 (1982), pp. 115–148 (cit. on pp. 3, 25, 72).

[20] Jean Berstel and Christophe Reutenauer. *Rational series and their languages*. Springer-Verlag Berlin, 1988 (cit. on pp. 25, 26, 37, 52).

[21] Gilles Blanchard, Gyemin Lee, and Clayton Scott. “Semi-supervised novelty detection”. In: *Journal of Machine Learning Research* 11.Nov (2010), pp. 2973–3009 (cit. on p. 135).

[22] Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. “Closing the learning-planning loop with predictive state representations”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 954–966 (cit. on p. 62).

[23] Symeon Bozapalidis and Archontia Grammatikopoulou. “Recognizable picture series”. In: *Journal of Automata, Languages and Combinatorics* 10.2/3 (2005), pp. 159–183 (cit. on pp. 25, 39).

[24] Symeon Bozapalidis and Olympia Louscou-Bozapalidou. “The rank of a formal tree power series”. In: *Theoretical Computer Science* 27.1 (1983), pp. 211–215 (cit. on p. 67).

[25] Bokai Cao, Hucheng Zhou, Guoqiang Li, and Philip S Yu. “Multi-view machines”. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM. 2016, pp. 427–436 (cit. on p. 3).

[26] J-F Cardoso. “Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem”. In: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE. 1990, pp. 2655–2658 (cit. on p. 4).

[27] Jack W. Carlyle and Azaria Paz. “Realizations by stochastic finite automata”. In: *Journal of Computer and System Sciences* 5.1 (1971), pp. 26–40 (cit. on p. 43).

[28] Jor-Ting Chan, Chi-Kwong Li, and Nung-Sing Sze. “Mappings preserving spectra of products of matrices”. In: *Proceedings of the American Mathematical Society* 135.4 (2007), pp. 977–986 (cit. on p. 53).

[29] Eric C Chi and Tamara G Kolda. “On tensors, sparsity, and nonnegative factorizations”. In: *SIAM Journal on Matrix Analysis and Applications* 33.4 (2012), pp. 1272–1299 (cit. on p. 76).

[30] Andrzej Cichocki. “Era of big data processing: A new approach via tensor networks and tensor decompositions”. In: *arXiv preprint arXiv:1403.2048* (2014) (cit. on p. 11).

[31] 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”. In: *IEEE Signal Processing Magazine* 32.2 (2015), pp. 145–163 (cit. on p. 4).

[32] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-Ichi Amari. *Alternating Least Squares and Related Algorithms for NMF and SCA Problems*. Wiley Online Library, 2009, pp. 203–266 (cit. on p. 87).

[33] 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 (cit. on p. 4).

[34] Shay B Cohen, Giorgio Satta, and Michael Collins. “Approximate PCFG parsing using tensor decomposition.” In: *HLT-NAACL*. 2013, pp. 487–496 (cit. on pp. 5, 63, 76, 78).

[35] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. “Spectral learning of latent-variable PCFGs: Algorithms and sample complexity”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 2399–2449 (cit. on p. 62).

[36] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle H Ungar. “Experiments with spectral learning of latent-variable PCFGs.” In: *HLT-NAACL*. 2013, pp. 148–157 (cit. on pp. 4, 76).

[37] Michael Collins and Shay B Cohen. “Tensor decomposition for fast parsing with latent-variable PCFGs”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 2519–2527 (cit. on pp. 4, 5, 63).

[38] Nicolo Colombo and Nikos Vlassis. “Tensor decomposition via joint matrix schur decomposition”. In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 2820–2828 (cit. on pp. 132, 147).

[39] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree automata techniques and applications*. 2007 (cit. on p. 27).

[40] Andrew Critch. "Algebraic geometry of hidden Markov and related models". PhD thesis. University of California, Berkeley, 2013 (cit. on p. 28).

[41] Andrew Critch and Jason Morton. "Algebraic geometry of matrix product states". In: *SIGMA* 10.095 (2014), p. 095 (cit. on p. 28).

[42] Anirban DasGupta. *Probability for statistics and machine learning: fundamentals and advanced topics*. Springer Science & Business Media, 2011 (cit. on p. 110).

[43] Lieven De Lathauwer. *Signal processing based on multilinear algebra*. Katholieke Universiteit Leuven, 1997 (cit. on pp. 18, 89).

[44] Lieven De Lathauwer. "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization". In: *SIAM journal on Matrix Analysis and Applications* 28.3 (2006), pp. 642–666 (cit. on p. 132).

[45] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278 (cit. on pp. 18, 89).

[46] Vin De Silva and Lek-Heng Lim. "Tensor rank and the ill-posedness of the best low-rank approximation problem". In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1084–1127 (cit. on p. 16).

[47] François Denis and Yann Esposito. "On rational stochastic languages". In: *Fundamenta Informaticae* 86.1, 2 (2008), pp. 41–77 (cit. on p. 136).

[48] Manfred Droste and Stefan Dück. "Weighted automata and logics on graphs". In: *Mathematical Foundations of Computer Science 2015*. Springer, 2015, pp. 192–204 (cit. on pp. 25, 58).

[49] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009 (cit. on pp. 25–27).

[50] Pierre Dupont, François Denis, and Yann Esposito. "Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms". In: *Pattern recognition* 38.9 (2005), pp. 1349–1371 (cit. on pp. 125, 136).

[51] Samuel Eilenberg and Bret Tilson. *Automata, languages, and machines*. Vol. 59. Academic press New York, 1976 (cit. on p. 24).

[52] Laurent El Ghaoui. "Inversion error, condition number, and approximate inverses of uncertain matrices". In: *Linear algebra and its applications* 343 (2002), pp. 171–193 (cit. on p. 84).

[53] Herbert Federer. *Geometric measure theory*. Springer, 2014 (cit. on pp. 55, 130).

[54] Rina Foygel, Michael Horrell, Mathias Drton, and John D Lafferty. "Nonparametric reduced rank regression". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1628–1636 (cit. on p. 89).

[55] Silvia Gandy, Benjamin Recht, and Isao Yamada. “Tensor completion and low-n-rank tensor recovery via convex optimization”. In: *Inverse Problems* 27.2 (2011), p. 025010 (cit. on p. 104).

[56] Hicham Ghennioui, Nadège Thirion-Moreau, Eric Moreau, Abdellah Adib, and Driss Aboutajdine. “Non unitary joint block diagonalization of complex matrices using a gradient approach”. In: *Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 201–208 (cit. on p. 52).

[57] Dora Giammarresi and Antonio Restivo. “Two-dimensional finite state recognizability”. In: *Fundamenta Informaticae* 25.3, 4 (1996), pp. 399–422 (cit. on p. 39).

[58] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. “Monadic second-order logic over rectangular pictures and recognizability by tiling systems”. In: *Information and computation* 125.1 (1996), pp. 32–45 (cit. on p. 39).

[59] E Mark Gold. “Language identification in the limit”. In: *Information and control* 10.5 (1967), pp. 447–474 (cit. on p. 50).

[60] Joshua Goodman. “Parsing algorithms and metrics”. In: *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1996, pp. 177–183 (cit. on p. 77).

[61] Lars Grasedyck. “Hierarchical singular value decomposition of tensors”. In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2029–2054 (cit. on pp. 18, 20).

[62] Lars Grasedyck, Daniel Kressner, and Christine Tobler. “A literature survey of low-rank tensor approximation techniques”. In: *GAMM-Mitteilungen* 36.1 (2013), pp. 53–78 (cit. on pp. 10, 16, 19).

[63] Rajarshi Guhaniyogi, Shaan Qamar, and David B Dunson. “Bayesian tensor regression”. In: *arXiv preprint arXiv:1509.06490* (2015) (cit. on p. 87).

[64] Weiwei Guo, Irene Kotsia, and Ioannis Patras. “Tensor learning for regression”. In: *IEEE Transactions on Image Processing* 21.2 (2012), pp. 816–827 (cit. on p. 87).

[65] Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*. Vol. 42. Springer Science & Business Media, 2012 (cit. on p. 10).

[66] Wolfgang Hackbusch and Stefan Kühn. “A new scheme for the tensor representation”. In: *Journal of Fourier Analysis and Applications* 15.5 (2009), pp. 706–722 (cit. on p. 20).

[67] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”. In: *SIAM review* 53.2 (2011), pp. 217–288 (cit. on p. 100).

[68] James J Heckman. *Sample selection bias as a specification error (with an application to the estimation of labor supply functions)*. National Bureau of Economic Research Cambridge, Mass., USA, 1977 (cit. on pp. 116, 124).

[69] Christopher J Hillar and Lek-Heng Lim. “Most tensor problems are NP-hard”. In: *Journal of the ACM (JACM)* 60.6 (2013), p. 45 (cit. on pp. 18, 101, 119, 147).

[70] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. “On manifolds of tensors of fixed TT-rank”. In: *Numerische Mathematik* 120.4 (2012), pp. 701–731 (cit. on p. 20).

[71] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. “The alternating linear scheme for tensor optimization in the tensor train format”. In: *SIAM Journal on Scientific Computing* 34.2 (2012), A683–A713 (cit. on p. 11).

[72] Ming Hou and Brahim Chaib-draa. “Hierarchical tucker tensor regression: Application to brain imaging data analysis”. In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1344–1348 (cit. on p. 87).

[73] Tailen Hsing and Randall Eubank. *Theoretical foundations of functional data analysis, with an introduction to linear operators*. John Wiley & Sons, 2015 (cit. on p. 69).

[74] Daniel Hsu and Sham M Kakade. “Learning mixtures of spherical gaussians: moment methods and spectral decompositions”. In: *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. ACM. 2013, pp. 11–20 (cit. on pp. 119, 127, 138).

[75] Daniel Hsu, Sham M Kakade, and Tong Zhang. “A spectral algorithm for learning hidden Markov models”. In: *arXiv preprint arXiv:0811.4413* (2008) (cit. on pp. 25, 62, 142).

[76] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. “Correcting sample selection bias by unlabeled data”. In: *Advances in neural information processing systems*. 2006, pp. 601–608 (cit. on p. 135).

[77] Thomas Huckle, K Waldherr, and T Schulte-Herbrüggen. “Computations in quantum tensor networks”. In: *Linear Algebra and its Applications* 438.2 (2013), pp. 750–781 (cit. on p. 11).

[78] Katsushi Inoue and Akira Nakamura. “Nonclosure properties of two-dimensional on-line tessellation acceptors and one way parallel sequential array acceptors”. In: *IEICE TRANSACTIONS (1976-1990)* 60.9 (1977), pp. 475–476 (cit. on p. 39).

[79] Alan Julian Izenman. “Reduced-rank regression for the multivariate linear model”. In: *Journal of multivariate analysis* 5.2 (1975), pp. 248–264 (cit. on pp. 89–91).

[80] Alan Julian Izenman. *Modern multivariate statistical techniques*. Vol. 1. Springer, 2008 (cit. on p. 91).

[81] R Jiang, MJ Zuo, and H-X Li. “Weibull and inverse Weibull mixture models allowing negative weights”. In: *Reliability Engineering & System Safety* 66.3 (1999), pp. 227–234 (cit. on p. 116).

[82] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002 (cit. on p. 2).

[83] Hachem Kadri, Emmanuel Duflos, Philippe Preux, Stéphane Canu, Alain Rakotomamonjy, and Julien Audiffren. “Operator-valued kernels for learning from functional response data”. In: *Journal of Machine Learning Research* 16 (2015), pp. 1–54 (cit. on p. 99).

[84] Purushottam Kar and Harish Karnick. “Random feature maps for dot product kernels”. In: *arXiv preprint arXiv:1201.6530* (2012) (cit. on p. 100).

[85] Kim A Keating and Steve Cherry. “Use and interpretation of logistic regression in habitat-selection studies”. In: *Journal of Wildlife Management* 68.4 (2004), pp. 774–789 (cit. on pp. 116, 124).

[86] Stefan Kiefer, Ines Marusic, and James Worrell. “Minimisation of multiplicity tree automata”. In: *Foundations of Software Science and Computation Structures*. 2015, pp. 297–311 (cit. on p. 70).

[87] Etienne de Klerk, Cristian Dobre, and Dmitrii V Pasechnik. “Numerical block diagonalization of matrix*-algebras with application to semidefinite programming”. In: *Mathematical programming* 129.1 (2011), pp. 91–111 (cit. on p. 52).

[88] Tamara G Kolda. “Symmetric orthogonal tensor decomposition is trivial”. In: *arXiv preprint arXiv:1503.01375* (2015) (cit. on p. 132).

[89] Tamara G Kolda and Brett W Bader. “Tensor decompositions and applications”. In: *SIAM review* 51.3 (2009), pp. 455–500 (cit. on pp. 10, 16, 19).

[90] Daniel Kressner and Christine Tobler. “htucker—A MATLAB toolbox for tensors in hierarchical Tucker format”. In: *Mathicse, EPF Lausanne* (2012) (cit. on p. 11).

[91] Werner Kuich and Arto Salomaa. *Semirings, automata, languages*. Vol. 5. Springer Science & Business Media, 2012 (cit. on p. 25).

[92] Volodymyr Kuleshov, Arun Tejasvi Chaganty, and Percy Liang. “Tensor factorization via matrix factorization.” In: *AISTATS*. 2015 (cit. on pp. 132, 147).

[93] Alex Kulesza, Nan Jiang, and Satinder Singh. “Low-rank spectral learning with weighted loss functions.” In: *AISTATS*. 2015 (cit. on p. 62).

[94] Alex Kulesza, N Raj Rao, and Satinder Singh. “Low-rank spectral learning.” In: *AISTATS*. 2014, pp. 522–530 (cit. on p. 62).

[95] Serge Lang. *Linear algebra. Undergraduate texts in mathematics*. 1987 (cit. on p. 2).

[96] Michel Latteux and David Simplot. “Recognizable picture languages and domino tiling”. In: *Theoretical computer science* 178.1 (1997), pp. 275–283 (cit. on p. 39).

[97] Alan J Laub. *Matrix analysis for scientists and engineers*. Society for Industrial and Applied Mathematics, 2004 (cit. on p. 13).

[98] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. “Speeding-up convolutional neural networks using fine-tuned cp-decomposition”. In: *arXiv preprint arXiv:1412.6553* (2014) (cit. on p. 4).

[99] Chi-Kwong Li and Stephen Pierce. “Linear preserver problems”. In: *The American Mathematical Monthly* 108.7 (2001), pp. 591–605 (cit. on p. 53).

[100] Xuelian Long, Lei Jin, and James Joshi. “Exploring trajectory-driven local geographic topics in foursquare”. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM. 2012, pp. 927–934 (cit. on p. 108).

[101] Aurelie C Lozano, Hongfei Li, Alexandru Niculescu-Mizil, Yan Liu, Claudia Perlich, Jonathan Hosking, and Naoki Abe. “Spatial-temporal causal modeling for climate change attribution”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 587–596 (cit. on p. 108).

[102] Haiping Lu, Konstantinos N Plataniotis, and Anastasios Venetsanopoulos. *Multilinear Subspace Learning: Dimensionality Reduction of Multidimensional Data*. CRC press, 2013 (cit. on p. 87).

[103] Takanori Maehara and Kazuo Murota. "Algorithm for error-controlled simultaneous block-diagonalization of matrices". In: *SIAM Journal on Matrix Analysis and Applications* 32.2 (2011), pp. 605–620 (cit. on p. 52).

[104] Johann A Makowsky and Tomer Kotek. "Connection matrices and the definability of graph parameters". In: *Logical Methods in Computer Science* 10 (2014) (cit. on pp. 25, 58).

[105] Andreas Maletti and Giorgio Satta. "Parsing algorithms based on tree automata". In: *Proceedings of the 11th International Conference on Parsing Technologies*. Association for Computational Linguistics. 2009, pp. 1–12 (cit. on p. 62).

[106] Ines Maruvsic and James Worrell. "Complexity of equivalence and learning for multiplicity tree automata". In: *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Part I. 2015, pp. 414–425 (cit. on pp. 57, 143).

[107] Ina Mäurer. "Recognizable and rational picture series". In: *Conference on Algebraic Informatics*. Citeseer. 2005, pp. 141–155 (cit. on p. 39).

[108] Ina Mäurer. "Weighted picture automata and weighted logics". In: (2006), pp. 313–324 (cit. on p. 39).

[109] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004 (cit. on p. 115).

[110] Charles A Micchelli and Massimiliano Pontil. "On learning vector-valued functions". In: *Neural computation* 17.1 (2005), pp. 177–204 (cit. on p. 99).

[111] RV Mises and Hilda Pollaczek-Geiringer. "Praktische Verfahren der Gleichungsauflösung." In: *ZAMM-Journal of Applied Mathematics and Mechanics* 9.2 (1929), pp. 152–164 (cit. on p. 120).

[112] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012 (cit. on pp. 103, 113, 114).

[113] Ashin Mukherjee and Ji Zhu. "Reduced rank ridge regression and its kernel extensions". In: *Statistical analysis and data mining* 4.6 (2011), pp. 612–622 (cit. on pp. 89–91, 100, 104).

[114] Philipp Müller, Simo Ali-Löytty, Marzieh Dashti, Henri Nurminen, and Robert Piché. "Gaussian mixture filter allowing negative weights and its application to positioning using signal strength measurements". In: *Positioning Navigation and Communication (WPNC), 2012 9th Workshop on*. IEEE. 2012, pp. 71–76 (cit. on p. 116).

[115] Kazuo Murota, Yoshihiro Kanno, Masakazu Kojima, and Sadayoshi Kojima. "A numerical algorithm for block-diagonal decomposition of matrix {*}-algebras with application to semidefinite programming". In: *Japan Journal of Industrial and Applied Mathematics* 27.1 (2010), pp. 125–160 (cit. on p. 52).

[116] Shashi Narayan and Shay B Cohen. "Optimizing spectral learning for parsing". In: *arXiv preprint arXiv:1606.02342* (2016) (cit. on p. 144).

[117] Maximilian Nickel and Volker Tresp. “An analysis of tensor models for learning on structured data”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 272–287 (cit. on p. 89).

[118] Dimitri Nion. “A tensor framework for nonunitary joint block diagonalization”. In: *Signal Processing, IEEE Transactions on* 59.10 (2011), pp. 4585–4594 (cit. on p. 52).

[119] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. “Tensorizing neural networks”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 442–450 (cit. on pp. 4, 11, 61).

[120] James M Ortega. *Numerical analysis: a second course*. Vol. 3. Siam, 1990 (cit. on p. 81).

[121] Ivan V Oseledets. “Tensor-train decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317 (cit. on p. 20).

[122] Jennie L Pearce and Mark S Boyce. “Modelling distribution and abundance with presence-only data”. In: *Journal of applied ecology* 43.3 (2006), pp. 405–412 (cit. on pp. 116, 124).

[123] Roger Penrose. “Applications of negative dimensional tensors”. In: *Combinatorial mathematics and its applications* (1971), pp. 221–244 (cit. on p. 11).

[124] Steven J Phillips, Miroslav Dudik, Jane Elith, Catherine H Graham, Anthony Lehmann, John Leathwick, and Simon Ferrier. “Sample selection bias and presence-only distribution models: implications for background and pseudo-absence data”. In: *Ecological Applications* 19.1 (2009), pp. 181–197 (cit. on pp. 116, 124).

[125] Richard S Pierce. *The associative algebra*. Springer, 1982 (cit. on p. 54).

[126] Lawrence R Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286 (cit. on p. 26).

[127] Guillaume Rabusseau and François Denis. “Maximizing a tree series in the representation space.” In: *ICGI*. 2014, pp. 124–138 (cit. on p. 144).

[128] Guillaume Rabusseau and Hachem Kadri. “Low-Rank Regression with Tensor Responses”. In: *Advances In Neural Information Processing Systems*. 2016 (cit. on p. 90).

[129] Guillaume Rabusseau, Hachem Kadri, and François Denis. “Régression de faible rang non-paramétrique pour réponses tensorielles”. In: *Colloque International Francophone de Traitement du Signal et de l’Image, GRETSI 2015, Lyon, France, September 8-11* (2015) (cit. on p. 99).

[130] Ali Rahimi and Benjamin Recht. “Random features for large-scale kernel machines”. In: *Advances in neural information processing systems*. 2007, pp. 1177–1184 (cit. on p. 100).

[131] Kurt Reidemeister. *Einfuhrung in die kombinatorische Topologie*. Vol. 86. American Mathematical Soc., 1950 (cit. on p. 44).

[132] Steffen Rendle. “Factorization machines”. In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 995–1000 (cit. on p. 3).

[133] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. “Multilinear multitask learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 1444–1452 (cit. on pp. 88, 89, 100, 104, 145).

[134] Grzegorz Rozenberg. *Handbook of graph grammars and computing by graph transformation*. Vol. 1. World Scientific, 1997 (cit. on p. 25).

[135] Jacques Sakarovitch. *Elements of automata theory*. Cambridge University Press, 2009 (cit. on pp. 25, 26).

[136] Arto Salomaa and Matti Soittola. *Automata-theoretic aspects of formal power series*. Springer Science & Business Media, 2012 (cit. on p. 25).

[137] Arthur L Samuel. “Some studies in machine learning using the game of checkers”. In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229 (cit. on p. 1).

[138] Tyler Sanderson and Clayton Scott. “Class proportion estimation with application to multiclass anomaly rejection.” In: (2014), pp. 850–858 (cit. on p. 135).

[139] Marcel Paul Schützenberger. “On the definition of a family of automata”. In: *Information and control* 4.2 (1961), pp. 245–270 (cit. on p. 24).

[140] James Scicluna and Colin De La Higuera. “PCFG Induction for unsupervised parsing and language modelling.” In: *EMNLP*. 2014, pp. 1353–1362 (cit. on p. 77).

[141] Clayton Scott. “A Rate of convergence for mixture proportion estimation, with application to learning from noisy labels.” In: *AISTATS*. 2015 (cit. on p. 135).

[142] Peter Semrl. “Maps on matrix spaces”. In: *Linear algebra and its applications* 413.2 (2006), pp. 364–393 (cit. on p. 54).

[143] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. “Tensor decomposition for signal processing and machine learning”. In: *arXiv preprint arXiv:1607.01668* (2016) (cit. on pp. 10, 16).

[144] Marco Signoretto, Lieven De Lathauwer, and Johan AK Suykens. “Learning tensors in reproducing kernel Hilbert spaces with multilinear spectral penalties”. In: *arXiv preprint arXiv:1310.4977* (2013) (cit. on pp. 89, 99, 109).

[145] Marco Signoretto, Quoc Tran Dinh, Lieven De Lathauwer, and Johan AK Suykens. “Learning with tensors: a framework based on convex optimization and spectral regularization”. In: *Machine Learning* 94.3 (2014), pp. 303–351 (cit. on pp. 87, 89).

[146] Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. “An annotation scheme for free word order languages”. In: *Proceedings of the fifth conference on Applied natural language processing*. Association for Computational Linguistics. 1997, pp. 88–95 (cit. on p. 76).

[147] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. Citeseer. 2013, p. 1642 (cit. on p. 65).

[148] Le Song, Animashree Anandkumar, Bo Dai, and Bo Xie. "Nonparametric estimation of multi-view latent variable models". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014, pp. 640–648 (cit. on p. 117).

[149] Nathan Srebro. "Learning with matrix factorizations". PhD thesis. Citeseer, 2004 (cit. on pp. 103, 113).

[150] Nathan Srebro, Noga Alon, and Tommi S Jaakkola. "Generalization error bounds for collaborative prediction with low-rank matrices". In: *Advances In Neural Information Processing Systems*. 2004, pp. 1321–1328 (cit. on p. 89).

[151] E Miles Stoudenmire and David J Schwab. "Supervised learning with quantum-inspired tensor networks". In: *arXiv preprint arXiv:1605.05775* (2016) (cit. on pp. 11, 143).

[152] Wolfgang Thomas. *On logics, tilings, and automata*. Springer, 1991 (cit. on p. 25).

[153] Raja Velu and Gregory C Reinsel. *Multivariate reduced-rank regression: theory and applications*. Vol. 136. Springer Science & Business Media, 2013 (cit. on pp. 91, 96).

[154] Hugh E Warren. "Lower bounds for approximation by nonlinear manifolds". In: *Transactions of the American Mathematical Society* 133.1 (1968), pp. 167–178 (cit. on pp. 103, 113).

[155] Steven R White. "Density matrix formulation for quantum renormalization groups". In: *Physical Review Letters* 69.19 (1992), p. 2863 (cit. on p. 20).

[156] Kishan Wimalawarne, Masashi Sugiyama, and Ryota Tomioka. "Multitask learning meets tensor factorization: task imputation via convex optimization". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2825–2833 (cit. on p. 89).

[157] Rose Yu, Dehua Cheng, and Yan Liu. "Accelerated online low-rank tensor learning for multivariate spatio-temporal streams". In: *International Conference on Machine Learning (ICML)*. 2015 (cit. on p. 88).

[158] Baibo Zhang and Changshui Zhang. "Finite mixture models with negative components". In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2005, pp. 31–41 (cit. on p. 116).

[159] Qibin Zhao, Cesar F Caiafa, Danilo P Mandic, Zenas C Chao, Yasuo Nagasaka, Naotaka Fujii, Liqing Zhang, and Andrzej Cichocki. "Higher order partial least squares (HOPLS): A generalized multilinear regression method". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.7 (2013), pp. 1660–1673 (cit. on pp. 89, 104).

[160] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. “Tensor ring decomposition”. In: *arXiv preprint arXiv:1606.05535* (2016) (cit. on pp. 11, 21, 146).

[161] Hua Zhou, Lexin Li, and Hongtu Zhu. “Tensor regression with applications in neuroimaging data analysis”. In: *Journal of the American Statistical Association* 108.502 (2013), pp. 540–552 (cit. on p. 87).

[162] James Y Zou, Daniel Hsu, David C Parkes, and Ryan P Adams. “Contrastive learning using spectral methods”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 2238–2246 (cit. on pp. 124, 131).

Index