NON-REDUNDANT TENSOR DECOMPOSITION

A Dissertation Presented

by

Olexiy Olexandrovych Kyrgyzov

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Electrical Engineering

Northeastern University
Boston, Massachusetts

July 30, 2010

Northeastern University

Abstract

Department of Electrical and Computer Engineering

Doctor of Philosophy in Electrical Engineering

by Olexiy Olexandrovych Kyrgyzov

This dissertation provides an overview and analysis of existing methods of tensor decomposition and describes a non-redundant tensor decomposition in terms of which we define the rank of a tensor. A tensor is a multidimensional or p-way array of scalars. Decompositions of tensors have applications in psychometrics, chemometrics, signal processing, numerical linear algebra, computer vision, numerical analysis, data mining, neuroscience, graph analysis, and elsewhere when we analyze order-p data with $p \geq 2$ [1]. The simplest case of tensor decomposition is singular value decompositions (SVD) when the order p equals 2, in this case SVD transforms data space into parametric space preserving cardinality. Two widely used tensor decompositions can be considered to be extensions of the SVD without preserving cardinality: CANDECOMP/PARAFAC (CP) decomposes a tensor as a sum of rank-1 tensors, and the Tucker decomposition is a higher-order form of principal component analysis (PCA) [1]. We present a tensor decomposition that includes SVD as a particular case, describes a tensor as a set of variables, defines an upper bound for the rank of tensors, and does not have redundancy as in the cases of CP and Tucker decompositions.

"Gedanken ohne Inhalt sind leer, Anschauungen ohne Begriffe sind blind."

Immanuel Kant

Acknowledgements

Моей любимой жене Кристиночке.

I would like to acknowledge my dissertation supervisor Professor Deniz Erdogmus for his help, trust, invaluable professional support, and long-run discussions. I am grateful to the committee members for their participation in my PhD defence: Professor Gilead Tadmor and Professor Octavia Camps. I thank them for thoughtful comments on my dissertation proposal which helped me to improve this manuscript. I also thank Professor Dana Brooks, Professor Hanoch Lev-Ari, and graduate coordinator Faith Crisley.

In this dissertation I have tried to cite and to refer all publications which were used to prepare it. In any case of coincidence, duplication and overlapping with other publications I have not aimed to appropriate their results and present them as my own. I also would like to mention very informative and helpful papers and technical reports of Prof. Tamara Kolda and Prof. Brett Bader from Sandia National Laboratories as well as

- Prof. Andrzej Cichocki from The Riken Brain Science Institute
- Prof. Danijel Skocaj from The University of Ljubljana
- Prof. Lieven De Lathauwer from The Katholieke Universiteit Leuven
- Prof. Morten Morup from The Technical University of Denmark
- Prof. Paul Leopardi from The Australian National University
- Prof. Pierre Comon from The University of Nice Sophia Antipolis
- Prof. Rafail Abramov from The University of Illinois at Chicago
- Prof. Rene Vidal from The Johns Hopkins University

I am very grateful to all colleagues at the department of Electrical and Computer Engineering for their help, advices, and excellent working atmosphere, especially to the team of Cognitive Systems Lab: Tanarat Dityam, Erhan Bas, Umut Orhan, Jernej Zupanc, Esra Cansizoglu, Shalini Purwar, Sheng You, and Hooman Nezamfar.

I wish to thank my family and my friends. Also I gratefully acknowledge my former colleagues from Dnipropetrovsk National University (Ukraine).

Separately many thanks to everybody who did not believe in my ideas.

Contents

List of Abbreviations

ALS	-	Alternating Least Squares.
BSS	-	Blind Signal Separation.
CANDECOMP	-	Canonical Decomposition.
CP	-	Candecomp/Parafac.
EEG	-	Electroencephalogram.
EVD	-	Eigenvalue Decomposition.
HITS	-	Hyperlink-Induced Topic Search.
ICA	-	Independent Component Analysis.
IFF	-	If And Only If.
IID	-	Independent And Identically Distributed.
ITL	-	Information Theoretic Learning.
KL	-	Kullback-Leibler Divergence.
LMS	-	Least Mean-Square.
LOG	-	Natural Logarithm.
MEE	-	Minimum Error Entropy.
MSE	-	Mean-Square Error.
NMF	-	Non-Negative Matrix Factorization.
NTF	-	Non-Negative Tensor Factorization.
PARAFAC	-	Parallel Factor Analysis.
PCA	-	Principal Component Analysis.
PDF	-	Probability Density Function.
SNR	-	Signal-To-Noise Ratio.
SVD	-	Singular Value Decomposition.

List of Symbols

a	-	scalar.
$\mathbf{a}$	-	vector.
$\mathbf{A}$	-	tensor.
$\{\mathbf{U}\}$	-	set of tensors $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(p)}$.
$\mathbf{B}^{-1}$	-	inverse of matrix $\mathbf{B}$.
$\mathbf{I}$	-	identity matrix.
$\mathbf{Q}$	-	permutation tensor.
$\mathbb{R}_+$	-	non-negative real number.
$\mathbb{R}^{n_1 \times \cdots \times n_p}$	-	order-p $(n_1, \ldots, n_p)$-dimensional real tensor.
$a_{i_1,\ldots,i_p}$	-	$(i_1, \ldots, i_p)^{th}$ element of an order-p tensor $\mathbf{A}$.
$(\mathbf{A})_{i_1,\ldots,i_p}$	-	$(i_1, \ldots, i_p)^{th}$ element of an order-p tensor $\mathbf{A}$.
$[\cdot]^T$	-	transpose of matrix or vector.
$\mathbf{A}^{\cdot p}$	-	rise to the power p each element of tensor $\mathbf{A}$.
$< \mathbf{A}, \mathbf{A} >$	-	Frobenius norm of tensor $\mathbf{A}$.
$[u]_+$	-	taking non-negative value, $max(0, u)$.
$arg \min_{\theta} e(\theta)$	-	denotes the value of θ that minimizes $e(\theta)$.
$det(\mathbf{B})$	-	determinant of matrix $\mathbf{B}$.
$E\{\cdot\}$	-	expectation operator.
$p(\mathbf{x})$	-	probability density function (p.d.f.) of $\mathbf{x}$.
$\lvert x \rvert$	-	absolute value (magnitude) of x.
$\lVert \mathbf{x} \rVert_p$	-	p-norm (length) of the vector $\mathbf{x}$, where $p = 0, \ldots, \infty$.
δ_{ij}	-	Kronecker delta.
∇	-	gradient operator.
$\circledast$	-	Hadamard product.
$\odot$	-	Khatri-Rao product.
$\circ$	-	outer product of vectors, e.g. $\mathbf{a} \circ \mathbf{b}$.
$\times_n$	-	n-way product.
$\times_{-n}$	-	product from each way except n^{th}.

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Problem Statement

Higher-order tensor decomposition appears in signal [2] and image [3] processing, factor analysis [4, 5, 6], speech and telecommunications [1], bioinformatics, etc., where we need to perform data analysis. Tensors and super symmetric tensors were actually already studied in the nineteenth century (homogeneous polynomials can be presented by symmetric tensors). In the '70s tensors appeared in Psychometrics and Linguistics, sciences in which multi-way arrays should to be analyzed. Subsequently they were used in chemometrics, where sets of excitation-emission matrices should be stacked in a third-order tensor. In the '90s non-Gaussian signal processing and higher-order statistics became popular, where the basic quantities are higher-order tensors. Near 2000 it was understood that the concept of "diversity" in telecommunication corresponds to the order of a tensor. Exponential signals, which are considered as the atoms of signal processing, can be represented by rank-1 tensors. Tensors have led to new efficient and accurate computation techniques. Semantic graphs, multilayer networks and hyperlink documents are represented by higher-order tensors. When one starts from a data matrix, one can wonder whether it would not be worthwhile to measure several matrices (under different conditions, at different time instances, etc.) so that one could make use of the more powerful structure of tensor algebra. This shift of paradigm concerns the most diverse aspects of mathematical engineering and goes together with the explosion of available information and the increase in computing power. The development of tensor methods is relevant to countless applications, e.g. tensor-based optimization, signal processing,

machine learning, data mining, diffusion tensor imaging, independent component analysis, blind source separation and factor analysis, diffusion tensor imaging.

Tensor analysis has become increasingly meaningful in theoretical and applied fields as it operates with data preserving their "native" state. For instance, Figure 1.1 shows us a result of data misrepresentation when we refuse to have a deal with tensors. Indeed, if we have an order-3 (5,5,2)-dimensional tensor representation we save "native" structure of data with two clusters. Otherwise, if we transform the tensor to habitual representation in form of matrix we will lose original relations between data points and final form of data has 3 clusters.

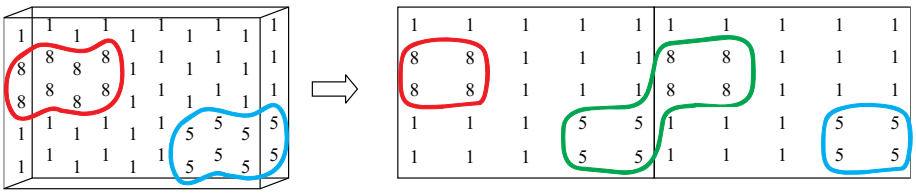


FIGURE 1.1: Tensor representation.

While the most data sets can be viewed as matrices, and therefore analyzed effectively using the SVD, certain data sets, such as a sequence of images, videos, and texts, cannot be represented as matrices without making some choices that might introduce unwanted artifacts, and thus make interpretation of the SVD results difficult. What is bewitching in tensors is that there is no equivalence to the SVD which is widely known. As this dissertation shows, there are several approaches to tensor decompositions, each with its own benefits. Their use depends a lot on the applications, and there are many applications for which tensor decomposition is the right tool. Below we provide some of interesting and useful examples.

As the size of the web increases, it becomes more and more important to analyze link structure considering context as well. Multilinear algebra provides a novel tool for incorporating anchor text and other information into the authority computation used by link analysis methods such as Hyperlink-Induced Topic Search (HITS) [7], Figure 1.2 [8]. T. Kolda and B.Bader [8] proposed Topical HITS (TOPHITS) method which uses a higher-order analogue of the matrix SVD called the PARAFAC model to analyze a three-way representation of web data. They compute hubs and authorities together with the terms that are used in the anchor text of the links between them. Adding a third dimension to the data greatly extends the applicability of HITS because the TOPHITS analysis can be performed in advance and offline. Like HITS,

the TOPHITS model reveals latent groupings of pages, but TOPHITS also includes latent term information. In their papers, the authors describe a faster mathematical algorithm for computing the TOPHITS model on sparse data, and Web data are used to compare HITS and TOPHITS. They also discuss how the TOPHITS model can be used in queries, such as computing context-sensitive authorities and hubs.

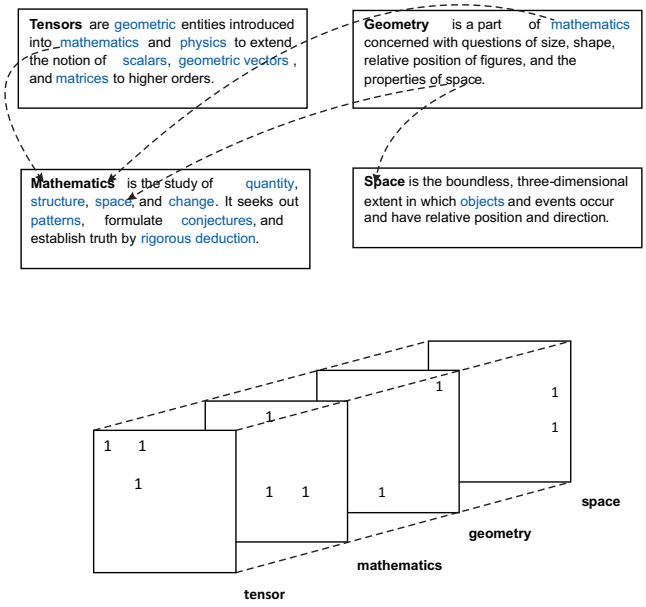


FIGURE 1.2: Three-dimensional view of the Web.

In the decomposition of multi-channel Electroencephalogram (EEG) signals, principal component analysis (PCA) and independent component analysis (ICA) have been widely used. However, as both methods are based on a two-way data processing, i.e., order-2 tensots or matrices, multi-way methods might improve the interpretation of frequency transformed multi-channel EEG of channel×frequency×time data, Figure 1.3. Morup at al. [9] were used PARAFAC to produce 5-way analysis of channel×frequency×time×subject×condition. They clearly presented how to perform data exploration using the decomposition on multi-way arrays. The PARAFAC decompositions were able to extract the expected features of a quantitative difference of coherent occipital gamma activity between conditions of a visual paradigm. Proposed method revealed a qualitative difference which has not previously been reported and showed the difference of regions of interest across modalities. The authors showed PARAFAC as a promising data exploratory tool in the analysis of the wavelets transformed event-related EEG.

An example of face recognition based on multilinear tensor algebra is presented in [10], Figure 1.4. The authors have investigated a technique for recognizing facial

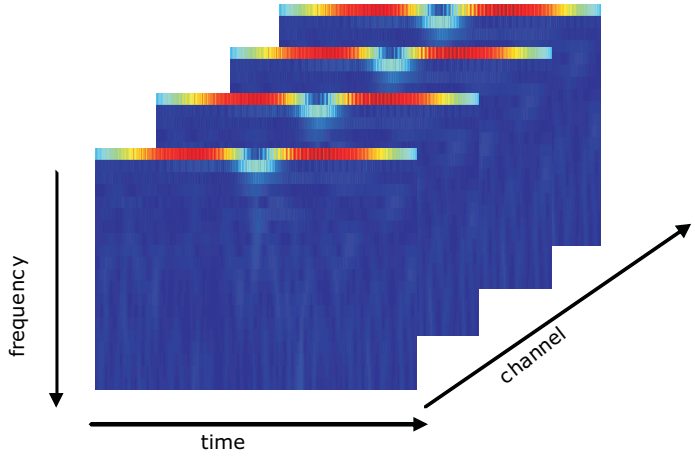


FIGURE 1.3: EEG signals analysis.

invariants of 3D facial expressions. Proposed potent framework possesses a remarkable ability to deal with the shortcomings of PCA in less constrained situations. A set of vector spaces are used to represent the variation of collections of face models with multiple formation factors across various modes, without destroying the details of each other. Using multilinear SVD yields better recognition rates than PCA. The authors have used a set of landmarks as the input data for their multilinear SVD recognition experiments. Results have shown that the choice of landmarks may contribute to the accuracy of recognition. They have used the face action coding system framework for manual selection of landmarks on prominent facial features as well as on muscle areas.

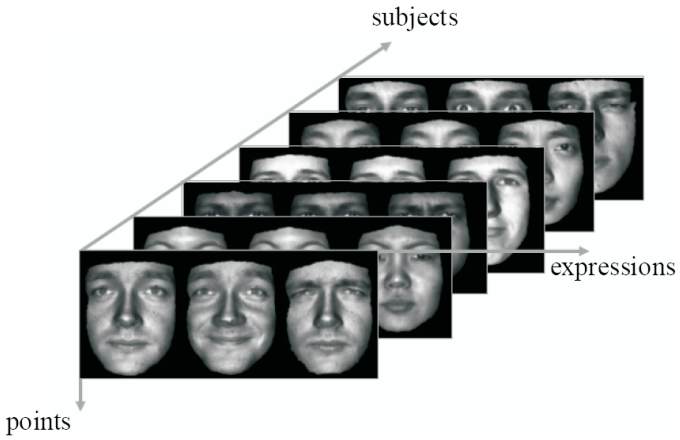


FIGURE 1.4: Tensor model for face recognition.

The Taylor series are used to define functions and "operators" in diverse areas of mathematics. For example, using Taylor series, one may define analytical functions

of matrices and operators, such as the matrix exponential that will be used in our evaluations below. The Taylor series represent a function as an infinite sum of terms calculated from the values of its derivatives at a single point [11]. In practice people use a finite number of terms of the series to approximate a function. Usually due to complexity ones truncate the Taylor series after the second member of sequence, such that we have a deal with scalars, vectors, and matrices. A function $f(x)$ that is infinitely differentiable in a neighborhood of a number a can be expressed in the Taylor series as:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots, \tag{1.1}$$

or in more compact form:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{1.2}$$

For several variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ we can generalize the Taylor series around some point $\mathbf{a} = [a_1, a_2, \ldots, a_n]$ as:

$$f(\mathbf{x}) = \sum_{i_1=0}^{\infty} \cdots \sum_{i_n=0}^{\infty} \frac{(x_1 - a_1)^{i_1} \cdots (x_n - a_n)^{i_n}}{i_1! \cdots i_n!} \left(\frac{\partial^{i_1 + \cdots + i_n} f}{\partial x_1^{i_1} \cdots \partial x_n^{i_n}} \right)(\mathbf{a}), \tag{1.3}$$

where all partial derivatives are represented as symmetric multidimensional higher-order tensors with $p = i_1 + \cdots + i_n$, Figure 1.5.

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}; \quad \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix};$$

$$(\nabla^p f)_{i_1 \ldots i_n} = \frac{\partial^p f}{\partial x_{i_1} \ldots \partial x_{i_n}};$$

FIGURE 1.5: Partial derivatives of multivariables function.

As tensors could emerge from higher-order statistics such as joint moments and cumulants (e.g. consider the symmetric tensor formed by order-p joint moments of n-dimensional random vector), we believe, tensor decompositions will play more

important role in statistical signal processing. In probability theory and statistics, the moment-generating function of any random variable is an alternate definition of its probability distribution [12]. In addition to univariate distributions, moment-generating functions can be defined for vector- or matrix-valued random variables, and can even be extended to generic cases of tensor form. The moment-generating function of a random variable x is

$$M_x(t) := E\left(e^{tx}\right), \quad t \in \mathbb{R}, \tag{1.4}$$

wherever this expectation exists the value of $M_x(0)$ is always equal to 1. A key problem with moment-generating functions is that moments and the moment-generating function may not exist, as the integrals do not need absolutely converge. In general case, for n-dimensional random vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ we have $M_{\mathbf{x}}(t) := E\left(e^{t^T\mathbf{x}}\right)$. The main properties of this function are that we can find all the moments of the distribution by the Taylor expression. The series expansion of e^{tx} is

$$e^{tx} = 1 + tx + \frac{t^2x^2}{2!} + \frac{t^3x^3}{3!} + \cdots . \tag{1.5}$$

For multidimensional random vector $\mathbf{x}$ the moment-generation function redefined by[1]

$$\mathbf{M}_{\mathbf{x}}(t) = E(e^{\mathbf{tx}}) = 1 + \mathbf{t}^T\mathbf{M}_1 + \frac{\mathbf{t}^T\mathbf{M}_2\mathbf{t}}{2!} + \frac{< \mathbf{t}^{\circ 3}, \mathbf{M}_3 >}{3!} + \cdots , \tag{1.6}$$

where $\mathbf{M}_i$ is the i^{th} moment of vector $\mathbf{x}$, Figure 1.6. To obtain i^{th} moment of random vector $\mathbf{x}$ we need to differentiate $\mathbf{M}_{\mathbf{x}}(\mathbf{t})$ i times with respect to $\mathbf{t}$ and then set $\mathbf{t} = 0$.

$$\mathbf{M}_1 = \begin{pmatrix} \sum \mathbf{x}_1 \\ \sum \mathbf{x}_2 \\ \vdots \\ \sum \mathbf{x}_n \end{pmatrix}; \quad \mathbf{M}_2 = \begin{pmatrix} \sum \mathbf{x}_1\mathbf{x}_1 & \sum \mathbf{x}_1\mathbf{x}_2 & \cdots & \sum \mathbf{x}_1\mathbf{x}_n \\ \sum \mathbf{x}_2\mathbf{x}_1 & \sum \mathbf{x}_2\mathbf{x}_2 & \cdots & \sum \mathbf{x}_2\mathbf{x}_n \\ \vdots & \vdots & \ddots & \vdots \\ \sum \mathbf{x}_n\mathbf{x}_1 & \sum \mathbf{x}_n\mathbf{x}_2 & \cdots & \sum \mathbf{x}_n\mathbf{x}_n \end{pmatrix};$$

$$(\mathbf{M}_p)_{i_1 \ldots i_p} = \sum (\mathbf{x}_{i_1}\mathbf{x}_{i_2} \ldots \mathbf{x}_{i_p});$$

FIGURE 1.6: Moments of random vector $\mathbf{x}$.

In 2008 The Defense Advanced Research Projects Agency (USA) established mathematical challenge on the tensor analysis called as "Beyond Convex Optimization: Can linear algebra be replaced by algebraic geometry in a systematic way?". Using the term algebraic geometry we mean higher-order multidimensional polynomials

[1] By the symbol "$\circ$" we denote outer product of vectors. More precisely we describe this operation in the next chapter.

while linear algebra operates just with scalars, vectors, and matrices. An order-p polynomial $f \in \mathbb{R}[\mathbf{x}]$ of n-dimensional vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ is expressed as:

$$f(\mathbf{x}) = a_0 + \mathbf{a}_1^T\mathbf{x} + \mathbf{x}^T\mathbf{A}_2\mathbf{x} + \mathbf{A}_3(\mathbf{x}, \mathbf{x}, \mathbf{x}) + \cdots + \mathbf{A}_p(\mathbf{x}, \ldots \mathbf{x}), \tag{1.7}$$

where $a_0, \mathbf{a}_1, \mathbf{A}_2$ are scalar, vector ,and matrix respectively, and $\mathbf{A}_p,\ p = 3, \ldots, \infty$ denotes order-p tensor. Here if we limit the upper bound of the order,p, equals to 2 then the polynomial $f(\mathbf{x})$ will depict numerical linear algebra. SVD is widely used in linear algebra for function analysis, contrariwise there is no such a decomposition for higher-order tensors. So the first step for algebraic geometry is to figure out the decomposition that involves SVD's properties for any tensor order.

1.2 Contribution of The Dissertation

In the context of this dissertation a tensor is a generalization of scalars, vectors, and matrices to higher-order structures. Tensor analysis is an increasingly relevant and interesting field of inquiry in signal processing and machine learning. The reason of such an interest is that generalization of rank-1 SVD of matrices have found considerable applications and success. Decompositions of tensors emerge and become relevant when high-order data or statistics are analyzed. Canonical decomposition (also called parallel factor analysis) and other decomposition methodologies exist and exhibit useful properties such as uniqueness. However, particular decompositions lack structure in its vector frame that forms the rank-one decomposition components, which prevent recursive solution formulations as in deflation of principal components. The main target of this dissertation is to analyze existing methods of tensor decomposition and develop a non-redundant tensor decomposition, which can be interpreted as a generalization of the spherical coordinate system of vectors and the orthogonal matrix group based on a predetermined frame of basis vectors. This imposed structure prevents the decomposition from achieving the minimum cross-product tensor rank as canonical decomposition does; however, we think it provides a solution that is more suitable for recursive calculations that is useful and important in dimension reduction applications.

Within this dissertation the following steps have been made:

- tensor representation and basic operations on them;
- matrix analysis with one-to-one reparametrization between data and set of variables;
- cardinal analysis of existing methods of tensor decomposition (CP and Tucker approaches);
- uniform distribution of points on a hyper-sphere;
- maximum entropy density estimation;
- generalized principal component analysis;
- solving systems of nonlinear equations;

Solving the problem of non-redundant tensor decomposition we reparameterize initial data set (tensor) into set of variables preserving cardinality. In this dissertation several new approaches to tensor decomposition are proposed:

- We describe the tensor by the set of free elements. In this term we define tensor factorization as a reparametrization procedure from data space to parameter space preserving cardinality. Heretofore any science researchs have not considered tensor decomposition in terms of a cardinal number property. We have shown such a property in the case of SVD and after that extended its to higher-order data. Final decomposition model has SVD as a particular case.
- In the Dissertation we outlined main properties of vector frame for tensor decomposition relaxing orthogonal behavior of vector frame for SVD. The structure of vector basis depends on the dimensions and the order of tensor. Existing method consider vector frame either as an orthogonal or an arbitrary directed vector frame. We also included Gram-Schmidt normalization for evaluation of decomposition model that was not involved in tensor decomposition before.
- Our approach gives exact equation for the tensor rank evaluation on the basis of its dimensions. In opposition to existing definitions of the rank our method does not have restrictions, fuzzyness or exceptional cases. We apply the same rule for any-dimensional tensor.

- In the experimental part of the Dissertation it is shown how tensor decomposition can be applied to the problems of density estimation, clustering, principal curve fitting, generalized and nonlinear principal component analysis. Wide range of existing applications on tensor decomposition does not usually involve these problems.

1.3 Dissertation Outline

The Dissertation is constructed in the following way: we start with notations and definitions of the main terms and operations on the tensor in Chapter 2. All examples presented in this chapter will consist of two parts: first, we describe an operation on matrix and then we generalize it to tensor form. Chapter 3 presents information on existing methods of tensor decomposition (CP and Tucker models) which now are mostly used in practice and are considered to be the basis for other methods. In the end of the chapter we present an analysis of them on cardinality between data and result of decomposition. The main properties of SVD which preserve cardinality are described and the main demands to non-redundant decomposition are defined. Data decomposition without redundancy is presented in Chapter 4. In this chapter we define the tensor rank and investigate the properties of such a model from the simplest case of the tensor ((2,2)-dimensional symmetric tensor) to general case of the tensor without any boundaries. This chapter also presents the main properties of basis vector frame. We also provide a differential model and show how it can be utilized to achieve minimum decomposition error. At the end of Chapter we compare proposed approach with existing ones. Chapter 5 presents solutions in different application fields for proposed model. A part of problems examined in this chapter is dedicated to restoration and application to data density distribution. The rest of the problems are extensions of existing linear approaches to signal processing. Finally, in Chapter 6 we conclude the main conceptions of this dissertation and provide future perspectives of non-redundant tensor decomposition.

Chapter 2

Notations and Definitions

The goal of this dissertation is to provide an overview of higher-order tensors, their decompositions, and then present our own non-redundant decomposition.

I want to acknowledge Prof. Tamara Kolda from The Sandia National Laboratories (CA\USA) as the most important author whose publications I have used from the beginning of my PhD program. Particularly, the material for this chapter has been primarily taken from [1] with additional citations for exact quotes, paraphrasing, etc.

"Though there has been active research on tensor decompositions and models (i.e., decompositions applied to data arrays for extracting and explaining their properties) for the past four decades, very little of this work has been published and applied in industry" [1]. Therefore, we wish to bring this research to the attention of applied mathematics readers.

As summarized in [1], the history of tensors is as follows. The terminology and theory of tensor decompositions firstly appeared in the papers of Hitchcock in 1927 [13, 14], and the extended idea of a multi-way data was proposed by Cattell in 1944 [15]. These conceptions became consistent until the works of Tucker on orthogonal base vector frame for tensor decomposition appeared in the middle of 1960s for the psychometrics problems [5]. In 1981 Appellof [16] was the first who utilized tensor decomposition in chemometrics after which tensors have become extremely popular in that field, even first book was published [17]. Simultaneously to the science investigations in psychometrics and chemometrics, Knuth [18] presented interesting approach in decompositions of bilinear forms in the algebraic complexity field.

Strassen matrix multiplication, which is applied for a decomposition of $4 \times 4 \times 4$ tensor through the 2×2 matrix multiplication, is an example of this.

Also we would like to point on the latest book in the field of non-negative tensor factorization [19]. Moreover, there are many software packages available for working with tensors from which we would like to note Matlab Tensor Toolbox from Sandia National Laboratories (CA/USA) [20, 21] .

2.1 Tensors

A tensor is a multidimensional array of scalars [1]. More precisely, an order-p tensor is an element of the space spanned by the outer product of p vectors, where each vector has its own dimension. We need to point a difference between the terms of tensor in mathematics and physics where it is considered as tensor fields, stress tensors or diffusion tensors [22]. There is an example of an order-4 (n_1, n_2, n_3, n_4)-dimensional tensor $\mathbf{A}$ with four different indices, Figure 2.1. By the subscript indices i_1, i_2, i_3, i_4 we denote entry value of the tensor, a_{i_1,i_2,i_3,i_4}.

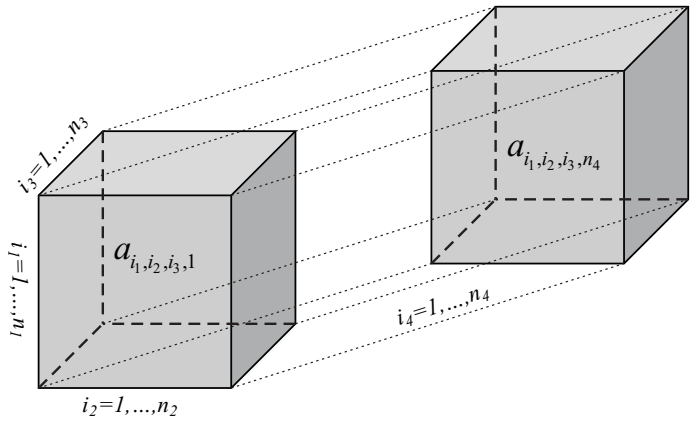


FIGURE 2.1: An order-4 tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$.

The main terms for tensors are the order, dimensions, and the rank which we describe in process of exposition. The order of a tensor is the number of non-singleton dimensions, also known as ways or modes. For instance, n-dimensional vector can be represented as a $(1, n)$- or $(n, 1)$-, or either $(1, 1, n)$-dimensional vector and just one non-singleton dimension is involved in evaluation of tensor's order. In general, an order-0 tensor is a scalar, an order-1 tensor is a vector, an order-2 tensor is a matrix, and tensors of order three or higher are higher-order tensors [1], they are presented on Figure 2.2.

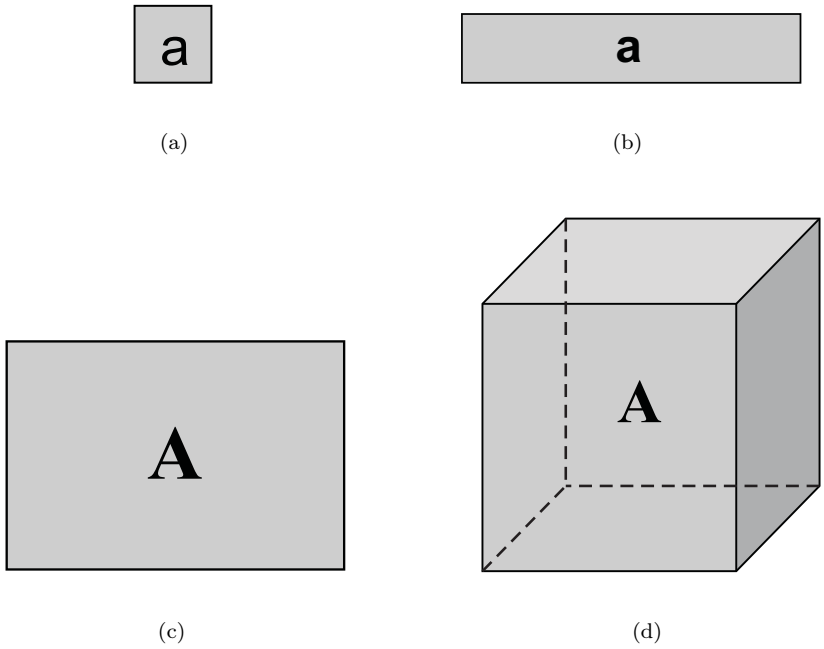


FIGURE 2.2: Tensors: (a) scalar – order-0, 1-dimensional tensor; (b) vector – order-1, n-dimensional tensor; (c) matrix – order-2, (n_1, n_2)-dimensional tensor; (d) higher-order tensor – order-3, (n_1, n_2, n_3)-dimensional tensor.

In this work we denote scalars by lowercase letters, a, vectors by boldface lowercase letters, $\mathbf{a}$, matrices and higher-order tensors (order-3 or higher) by boldface capital letters, $\mathbf{A}$.

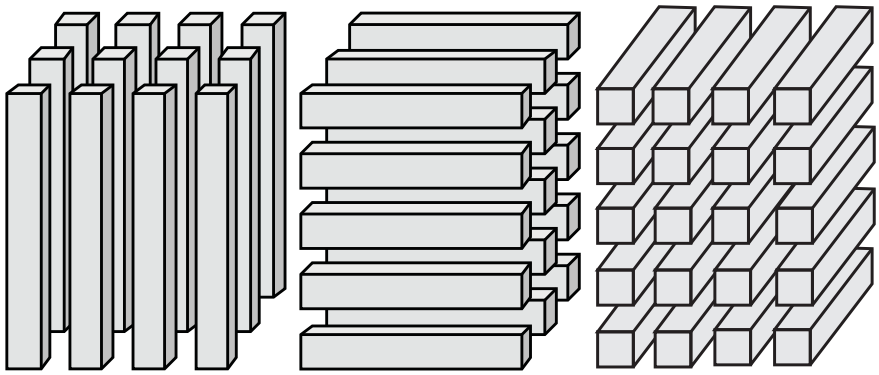
2.2 Basic Tensor Notations

In course of statement of the Dissertation, we have tried to stay as consistent as possible with terminology that would be familiar to applied mathematicians and with the terminology of previous publications in the area of tensor decompositions [1, 4, 23].

The i^{th} entry of a vector $\mathbf{a}$ is denoted by a_i, element (i_1, i_2) of a matrix $\mathbf{A}$ is denoted by $a_{i_1 i_2}$, and element (i_1, i_2, i_3) of a third-order tensor $\mathbf{A}$ is denoted by $a_{i_1 i_2 i_3}$, etc. To point out a certain elements in a tensor preserving the face of letter we will use round brackets with subindexes, e.g., $a_i = (\mathbf{a})_i$ or $a_{i_1,i_2} = (\mathbf{A})_{i_1,i_2}$. Indices typically range from 1 to their upper value denoted by lowercase letters, e.g., $i_1 = 1, \ldots, n_1$. The k^{th} element in a sequence is denoted by a superscript in parentheses, e.g., $\mathbf{A}^{(k)}$ denotes the k^{th} matrix in a sequence [1].

Subarrays are formed when a subset of the indices is fixed. For matrices, these are the rows and columns. A colon (:) is used to indicate all elements of a mode. Thus, the i^{th} column of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_{:i}$, and the i^{th} row of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_{i:}$. Equivalently, the i^{th} column of a matrix, $\mathbf{a}_{:i}$, may be denoted more compactly as $\mathbf{a}_i$, when it is obvious from content [1].

Fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $\mathbf{x}_{:i_2i_3}$, $\mathbf{x}_{i_1:i_3}$, and $\mathbf{x}_{i_1i_2:}$, respectively, Figure 2.3 [1]. Fibers extracted from the tensor are always assumed to be oriented as column vectors [1].



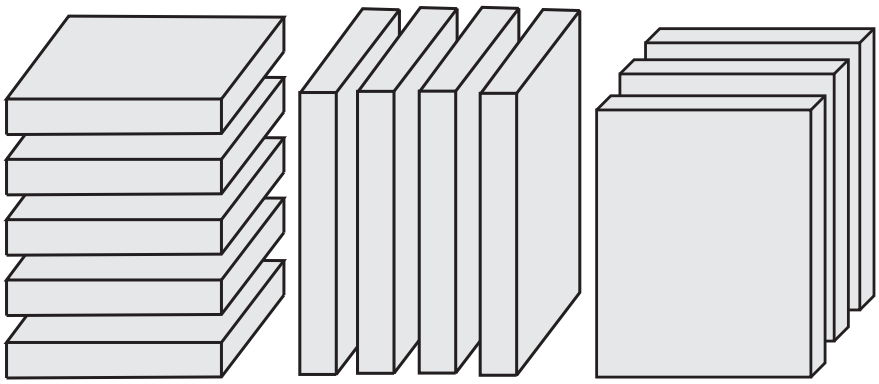
(a) mode-1 (column) fibers: $\mathbf{x}_{:i_2i_3}$ (b) mode-2 (row) fibers: $\mathbf{x}_{i_1:i_3}$ (c) mode-3 (tube) fibers: $\mathbf{x}_{i_1i_2:}$

FIGURE 2.3: Fibers of an order-3 (5,4,3)-dimensional tensor.

Slices are 2-dimensional sections of a tensor, defined by fixing all but two indices. Figure 2.4 shows the horizontal, lateral, and frontal slides of a third-order (5,4,3)-dimensional tensor $\mathbf{X}$, denoted by $\mathbf{X}_{i_1::}$, $\mathbf{X}_{:i_2:}$, and $\mathbf{X}_{::i_3}$, respectively [1]. Alternatively, the i_3^{th} frontal slice of a third-order tensor, $X_{::i_3}$, may be denoted more compactly as $\mathbf{X}_{i_3}$ [1].

The Frobenius norm of an order-p tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is the square root of the sum of the squares of all its elements, i.e.,

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_p=1}^{n_p} x_{i_1,i_2\ldots,i_p}^2} \tag{2.1}$$



(a) horizontal slices: $\mathbf{X}_{i_1::}$ (b) lateral slices: $\mathbf{X}_{:i_2:}$ (c) frontal slices: $X_{::i_3}$

FIGURE 2.4: Slices of an order-3 (5,4,3)-dimensional tensor.

This is an analogous to the matrix Frobenius norm, which is denoted as $\|\mathbf{A}\|_F$ for a matrix $\mathbf{A}$. The inner product of two same-sized tensors $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is the sum of the products of their entries, i.e.,

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_p=1}^{n_p} x_{i_1 i_2 \ldots i_p} y_{i_1 i_2 \ldots i_p} \tag{2.2}$$

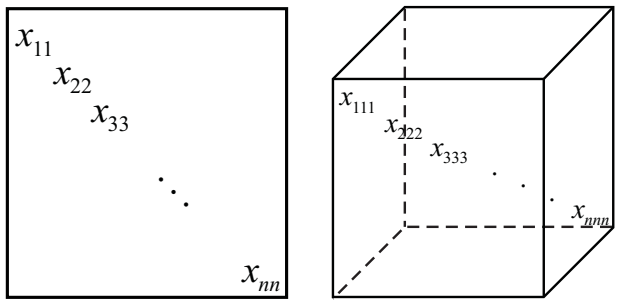
It follows immediately that $\langle \mathbf{X}, \mathbf{X} \rangle = \|\mathbf{X}\|_F^2$. As we utilize this inner-product and norm throughout the proposal, we will not explicitly refer to them as "Frobenius".

2.3 Diagonal Tensors

A tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is diagonal if $x_{i_1 i_2 \cdots i_p} \neq 0$ and $i_1 = i_2 = \cdots = i_p$. Figure 2.5 illustrates diagonal matrix and cubical tensor with non-zero entries along their main diagonals, respectively.

2.4 Symmetric Tensors

A tensor is called cubical if every mode has the same dimension, i.e., $\mathbf{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$. A cubical tensor is called supersymmetric [24], if its elements constantly remain under any permutation of the indices. For instance, an order-3 tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times n}$



(a) diagonal (n, n)-dimensional tensor.

(b) diagonal (n, n, n)-dimensional tensor.

FIGURE 2.5: Diagonal order-2 and order-3 tensors.

is symmetric if $x_{i_1i_2i_3} = x_{i_1i_3i_2} = x_{i_2i_1i_3} = x_{i_2i_3i_1} = x_{i_3i_1i_2} = x_{i_3i_2i_1}$ for all $i_1, i_2, i_3 = 1, \ldots, n$. In our work we are going to name supersymmetry as symmetry because we do not consider partial symmetry in tensors. Tensors can be partially symmetric in two or more modes as well. For example, an order-3 tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ is symmetric in modes one and two if all its frontal slices are symmetric, i.e., $\mathbf{X}_{i_3} = \mathbf{X}_{i_3}^T$ for all $i_3 = 1, \ldots, m$. Analysis of (super) symmetric tensors, which can be shown to be bijectively related to homogeneous polynomials, is presented in [14, 24].

2.5 Tensor Rank

An order-p tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is rank-1 if it can be written as the outer product of p vectors [1], i.e.,

$$\mathbf{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(p)} \tag{2.3}$$

The symbol " $\circ$ " represents the vector outer product. This means that each element of the tensor is the product of the corresponding vector elements [1]:

$$x_{i_1 i_2 \cdots i_p} = a_{i_1}^{(1)} a_{i_2}^{(2)} \cdots a_{i_p}^{(p)} \text{ for all } 1 \leq i_k \leq n_k. \tag{2.4}$$

Figure 2.6 illustrates $\mathbf{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}$, an order-3 rank-1 tensor, where the (i_1, i_2, i_3) element of $\mathbf{X}$ is given by $x_{i_1 i_2 i_3} = a_{i_1}^{(1)} a_{i_2}^{(2)} a_{i_3}^{(3)}$.

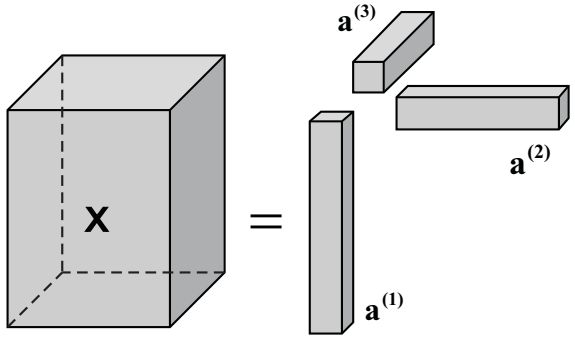


FIGURE 2.6: Rank-1 order-3 tensor $\mathbf{X}$.

Rank-r tensor $\mathbf{X}$, Figure 2.7, consists of the sum of r rank-1 tensors:

$$\mathbf{X} = \sum_{i=1}^{r} \mathbf{a}_i^{(1)} \circ \cdots \circ \mathbf{a}_i^{(p)}. \tag{2.5}$$

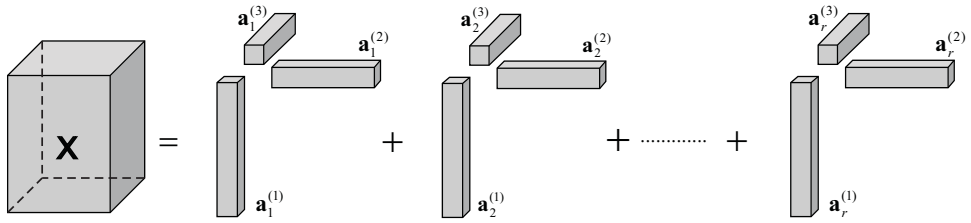


FIGURE 2.7: Rank-r order-3 tensor $\mathbf{X}$.

The rank of a tensor $\mathbf{X}$, denoted by $rank(\mathbf{X})$, is defined as the smallest number of rank-1 tensors that generate $\mathbf{X}$ as their sum [1]. In other words, this is the smallest number of components in an exact CP decomposition [1], see Section 3.1. The first definition of the rank proposed by Hitchcock in 1927 [13], and Kruskal did so independently 50 years later [25]. An exact CP decomposition with $r = rank(\mathbf{X})$ components is called the rank decomposition [1].

The definition of tensor rank is exactly the same as for matrix rank, but the properties of matrix and tensor ranks are quite different. One difference is that the rank of a real-valued tensor may actually be different over $\mathbb{R}$ and $\mathbb{C}$, see [1] for more details. Another major difference between matrix and tensor rank is that there is no straightforward algorithm to determine the rank of a specific given tensor; in fact, the problem is NP-hard. In practice, the rank of a tensor is determined numerically by fitting various CP or Tucker models [1]. In tensor calculus we can have a deal with maximum and typical ranks. The maximum rank is defined as the largest attainable

rank, whereas the typical rank is any rank that occurs with probability greater than zero. For the collection of (n_1, n_2)-dimensional matrices, the maximum and typical ranks are identical and equal to $min(n_1, n_2)$. For tensors, the two ranks may be different in state-of-the-art and identical in our approach [1]. In fact, Monte Carlo experiments based on existing theory reveal that the set of $(2, 2, 2)$-dimensional tensors of rank two fills about 79% of the space, while those of rank three fill 21%. Rank-1 tensors are possible but occur with zero probability [1].

One of the main achievements of the Dissertation is we can always compute the maximal tensor rank for any order and dimensions of a tensor. Below we shortly describe the most popular approach for the tensor rank evaluation proposed by Comon [26].

In 2002 Comon investigated the special case of symmetric tensors over $\mathbb{C}$. Let $\mathbf{X}$ be an order-p $(n, \ldots, n)$-dimensional symmetric tensor. Define the symmetric rank of $\mathbf{X}$ to be as

$$rank(X) = min\{r : \ \mathbf{X} = \sum_{i=1}^{r} \mathbf{a}_i^{\circ p}, \text{ where } \mathbf{A} \in \mathbb{C}^{n,r}\}, \tag{2.6}$$

i.e., the minimum number of symmetric rank-1 tensor [1]. Comon has shown that the lower bound of the rank for such a symmetric tensor is

$$rank(\mathbf{X}) \geq \left\lceil \frac{\binom{n+p-1}{p}}{n} \right\rceil, \tag{2.7}$$

except for the cases $(p, n) \in (3, 5),\ (4, 3),\ (4, 4),\ (4, 5)$, when it should be increased by one.

For the case of non-symmetric tensors the lower bound of the rank is evaluated by

$$rank(\mathbf{X}) \geq \left\lceil \frac{\prod_{i=1}^{p} n_i}{1 + \sum_{i=1}^{p}(n_i - 1)} \right\rceil. \tag{2.8}$$

2.6 Matricizing: Transforming a Tensor into a Matrix

In many numerical tasks there is often need to transform a tensor into a matrix or vector form and vice versa since it is easer to perform basic operations in vector-matrix forms. Especially it is important because all data in computer are stored in

a vector form with liner index and we do such a transformation all the time when we access to stored data.

One can find different names of a tensor-matrix transformation, we use term "matricizing" as in [1, 23], since it is a general name of final data form after rearrangement, where we reorder the elements of an order-p array into a matrix, Figure 2.8. Here both matrices and vectors are obtained by identical algorithms, the difference is that in tensor-vector transformation one of two dimensions is always equal one. Although some authors use term "unfolding" [2] or "flattening" [27] for the such operation.

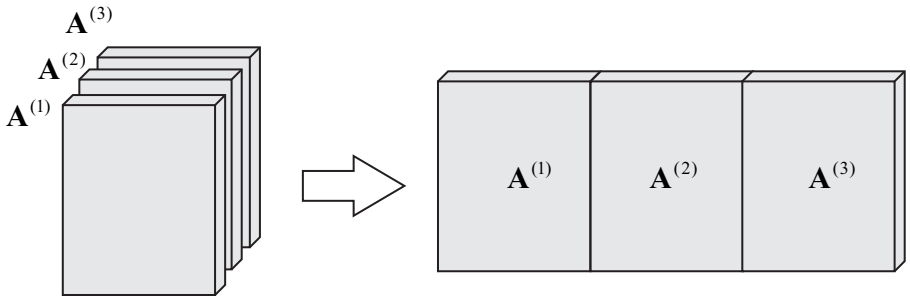


FIGURE 2.8: Transformation of a rank-3 tensor $\mathbf{A}$ to a matrix.

For instance, a $2 \times 3 \times 4$ tensor can be arranged as a 6×4 matrix or a 3×8 matrix, and so on [1]. In this work, we consider only the special case of mode-k matricizing because it is the only form relevant to our discussion. More general treatment of matricizing can be found in [28]. The mode-k matricizing of a tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is denoted by $\mathbf{A}_{(k)}$ and arranges the mode-k fibers to be the columns of the resulting matrix. The concept cam be more easily understood if we introduce it with an example [1]. Let the frontal slices of $\mathbf{A} \in \mathbb{R}^{3 \times 4 \times 2}$ be

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}. \tag{2.9}$$

Then the three mode-k unfoldings are:

$$\mathbf{A}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}, \mathbf{A}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}, \tag{2.10}$$

$$\mathbf{A}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & \cdots & 21 & 22 & 23 & 24 \end{bmatrix}.$$

Sometimes different papers use different orderings of the columns for the mode-k unfolding [1]. In general, the specific permutation of columns is not important as long as it is consistent across all related calculations. Last, we note that it is also possible to vectorize a tensor. Once again the ordering of the elements is not important as long as it is consistent. In the example above, the vectorized version of the data [1] is

$$vec(\mathbf{A}) = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ 23 \\ 24 \end{bmatrix}. \tag{2.11}$$

Now we present general operation of matricizing. Let $\mathbf{A}$ be an order-p $(n_1, n_2, \ldots, n_p)$-dimensional tensor, and it is supposed that we wish to rearrange this to be a matrix of size $m_1 \times m_2$ (or a vector when m_1 or m_2 equal one) [21]. Clearly, the number of entries in the matrix must equal the number of entries in the tensor; in other words, $\prod_{i=1}^{p} n_i = m_1 m_2$. Given m_1 and m_2 satisfy the aforementioned property, the mapping can be done by any number of ways, so long as we have a one-to-one mapping ϕ such that

$$\phi : \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \times \cdots \times \{1, \ldots, n_p\} \mapsto \{1, \ldots, m_1\} \times \{1, \ldots, m_2\} \tag{2.12}$$

Here we convert a tensor to a matrix as follows [21]. Let the set of indices be partitioned into two disjoint subsets: $\{1, \ldots, p\} = \{r_1, \ldots, r_k\} \cup \{c_1, \ldots, c_l\}$. The set $\{r_1, \ldots, r_k\}$ defines those indices that will be mapped to the row indices of the resulting matrix and the set $\{c_1, \ldots, c_l\}$ defines those indices that will likewise be

mapped to the column indices [21]. In this case,

$$m_1 = \prod_{j=1}^{k} n_{r_j} \text{ and } m_2 = \prod_{j=1}^{l} n_{c_j}. \tag{2.13}$$

Then we define $\phi(i_1, i_2, \ldots, i_p) = (j_1, j_2)$, where

$$j_1 = 1 + \sum_{j=1}^{k} [(i_{r_j} - 1) \prod_{\hat{j}=1}^{j-1} n_{r_{\hat{j}}}] \text{ and } j_2 = 1 + \sum_{j=1}^{l} [(i_{c_j} - 1) \prod_{\hat{j}=1}^{j-1} n_{c_{\hat{j}}}]. \tag{2.14}$$

Note that the sets $\{r_1, \ldots, r_k\}$ and $\{c_1, \ldots, c_l\}$ can be of any order and are not necessarily ascending [21].

2.7 Tensor Multiplication

Notations and operations for tensor multiplication are very similar to matrix ones, though obviously the symbols for such operations are much more complex. The main problem with such operations is to determine which dimensions are to be multiplied and how the dimensions of the result should be ordered [21]. In this section we consider three types of tensor multiplication: a tensor with a vector, a tensor with a matrix, and a tensor with a tensor. For more information on tensor multiplications we refer the reader to [21, 29].

2.7.1 Multiplying a Tensor With a Vector

Multiplication of a tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_p}$ with a vector $\mathbf{v} \in \mathbb{R}^{n_k}$ along k^{th}-mode is denoted by $\mathbf{A} \times_k \mathbf{v}$. The result of such a multiplication is an order-$(p-1)$ tensor, i.e., the size is $n_1 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_p$. Elementwise, we have

$$(\mathbf{A} \times_k \mathbf{v})_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_p} = \sum_{i_k=1}^{n_k} a_{i_1,\ldots,i_p} v_{i_k} \tag{2.15}$$

The idea is to compute the inner product of each mode-k fiber with the vector $\mathbf{v}$. For example, let $\mathbf{A}$ be as given in (2.9) and define $\mathbf{v} = [1\ 2\ 3\ 4]^T$ [1]. Then

$$\mathbf{A} \times_2 \mathbf{v} = \begin{bmatrix} 70 & 190 \\ 80 & 200 \\ 90 & 210 \end{bmatrix} \tag{2.16}$$

One source of confusion in k-mode multiplication is what to do to the singleton dimension in mode k when we multiply a tensor with a vector. If the singleton dimension is dropped (as is described above) then the commutativity of multiplies is not held because the order of the intermediate result changes and multiplication applies to the wrong mode [21]. When it comes to mode-k vector multiplication, precedence matters because the order of the intermediate results changes [1]. In other words,

$$(\mathbf{A} \times_m \mathbf{u}) \times_k \mathbf{v} \neq (\mathbf{A} \times_k \mathbf{v}) \times_m \mathbf{u}. \tag{2.17}$$

However, a different statement about commutativity may be made [21]. If we assume $m < k$, then

$$(\mathbf{A} \times_m \mathbf{u}) \times_{k-1} \mathbf{v} = (\mathbf{A} \times_k \mathbf{v}) \times_m \mathbf{u}. \tag{2.18}$$

Although we can usually determine the correct order of the result via the context of the equation [21], it is very important for software realization. As it will be shown in the next chapter, it is useful to calculate the product of a tensor and a sequence of vectors [21], for instance:

$$\mathbf{B} = \mathbf{A} \times_1 \mathbf{u}^{(1)} \times_2 \mathbf{u}^{(2)} \cdots \times_p \mathbf{u}^{(p)} \tag{2.19}$$

or

$$\mathbf{B} = \mathbf{A} \times_1 \mathbf{u}^{(1)} \cdots \times_{k-1} \mathbf{u}^{(k-1)} \times_{k+1} \mathbf{u}^{(k+1)} \cdots \times_p \mathbf{u}^{(p)}. \tag{2.20}$$

Alternative notations for these operations are $\mathbf{B} = \mathbf{A} \times \{\mathbf{u}\}$ and $\mathbf{B} = \mathbf{A} \times_{-n} \{\mathbf{u}\}$, respectively.

In practice, we must be careful, when calculating a sequence of contracted n-mode products, to perform the multiplications starting with the highest mode and proceeding sequentially to the lowest [21].

2.7.2 Multiplying a Tensor With a Matrix

In the case of matrix multiplication, the specification of which dimensions should be multiplied is straightforward Ů it is always the inner product of the rows of the first matrix with the columns of the second matrix [21]. When we apply a transposition of matrix then we just swaps the rows and columns. Because tensors may have an arbitrary number of dimensions, the situation is more complicated. In this case, we need to specify which dimension of the tensor is multiplied by the columns (or rows) of the given matrix [21]. Transposition for the tensor is not defined now in state-of-the-art so it is one of the future problems in tensor calculus.

The k-mode product of a tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times cdots \times n_p}$ with a matrix $\mathbf{U} \in \mathbb{R}^{j \times n_k}$ is denoted by $\mathbf{A} \times_k \mathbf{U}$ and dimension of the obtained result is $n_1 \times \cdots \times n_{k-1} \times j \times n_{k+1} \times \cdots \times n_p$. Elementwise, we have

$$(\mathbf{A} \times_k \mathbf{U})_{i_1 \cdots i_{k-1} j i_{k+1} \cdots i_p} = \sum_{i_k=1}^{n_k} a_{i_1 i_2 \cdots i_p} u_{j i_k}. \tag{2.21}$$

Each mode-k fiber is multiplied by the matrix $\mathbf{U}$. This idea can also be expressed in the terms of unfolded tensors [1]:

$$\mathbf{Y} = \mathbf{A} \times_k \mathbf{U} \;\Leftrightarrow\; \mathbf{Y} = \mathbf{U}\mathbf{A}_{(k)} \tag{2.22}$$

The k-mode product of a tensor with a matrix is related to a change of basis in the case when a tensor defines a multilinear operator [1]. As an example, let $\mathbf{A}$ be the tensor defined above in (2.9) and let $\mathbf{U} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$. Then the product $\mathbf{Y} = \mathbf{A} \times_1 \mathbf{U} \in \mathbb{R}^{2 \times 4 \times 2}$ is

$$\mathbf{Y}_1 = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix}, \; \mathbf{Y}_2 = \begin{bmatrix} 130 & 157 & 184 & 211 \\ 172 & 208 & 244 & 280 \end{bmatrix} \tag{2.23}$$

We also need to point out one property of the order of k-mode matrix product. For certain modes in a sequence of multiplications, the order of the multiplication is unimportant [1], e.g.,

$$\mathbf{A} \times_k \mathbf{U} \times_m \mathbf{Y} = \mathbf{A} \times_m \mathbf{Y} \times_k \mathbf{U} \quad \text{if} \quad m \neq k. \tag{2.24}$$

If the modes are the same, then

$$\mathbf{A} \times_k \mathbf{U} \times_k \mathbf{Y} = \mathbf{A} \times_k (\mathbf{YU}). \tag{2.25}$$

To understand k-mode multiplication in terms of matrices (i.e., order-2 tensors), suppose $\mathbf{A}$ is (m, n)-dimensional, $\mathbf{U}$ is (m, l)-dimensional, and $\mathbf{V}$ is (k, l)-dimensional tensors [21]. It follows that

$$\mathbf{A} \times_1 \mathbf{U}^T = \mathbf{U}^T\mathbf{A} \quad \text{end} \quad \mathbf{A} \times_2 \mathbf{V}^T = \mathbf{AV}. \tag{2.26}$$

Thereby, the matrix SVD can be written as

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \Sigma \times_1 \mathbf{U} \times_2 \mathbf{V}. \tag{2.27}$$

The k-mode product satisfies the following property [29]: Let $\mathbf{A}$ be an order-p $(n_1, n_2, \ldots, n_p)$-dimensional tensor. If $\mathbf{U}$ and $\mathbf{V}$ are (d_m, n_m) and (d_k, n_k)-dimensional tensors, respectively, then

$$\mathbf{A} \times_m \mathbf{U} \times_k \mathbf{V} = \mathbf{A} \times_k \mathbf{V} \times_m \mathbf{U}. \tag{2.28}$$

As in the case of tensor times a vector product it is useful to calculate the product of a tensor and a sequence of matrices [21]. Let $\mathbf{A}$ be an order-p $(n_1, \ldots, n_p)$-dimensional tensor, and let $\mathbf{U}^{(i)}$ denote a (m_i, n_i)-dimensional matrix, $i = 1 \ldots, p$ [21]. Then the sequence of products

$$\mathbf{B} = \mathbf{A} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_p \mathbf{U}^{(p)} \tag{2.29}$$

is order-p $(m_1, \ldots, m_p)$-dimensional tensor. Alternative notation for this operation is $\mathbf{B} = \mathbf{A} \times \{\mathbf{U}\}$. Another frequently used operation is multiplying by all but one of a sequence of matrices [21]:

$$\mathbf{B} = \mathbf{A} \times_1 \mathbf{U}^{(1)} \cdots \times_{k-1} \mathbf{U}^{(k-1)} \times_{k+1} \mathbf{U}^{(k+1)} \cdots \times_p \mathbf{U}^{(p)}. \tag{2.30}$$

Alternative notation is $\mathbf{B} = \mathbf{A} \times_{-k} \{\mathbf{U}\}$ [21]. We will use these notations when describe algorithms of redundant decompositions in the next chapter.

2.7.3 Multiplying a Tensor With a Tensor

The last class of tensor multiplication to consider is the product of two tensors [21]. We describe three general operations for tensor-to-tensor multiplication: outer product, contracted product, and inner product [21]. The outer product of two tensors is defined as follows. Let $\mathbf{A}$ be a $(n_1, \ldots, n_{p_1})$- and $\mathbf{B}$ be $(m_1, \ldots, m_{p_2})$-dimensional tensors. The outer product $\mathbf{A} \circ \mathbf{B}$ is an order-$(p_1 + p_2)$ $(n_1, \ldots, n_{p_1}, m_1, \ldots, m_{p_2})$-dimensional tensors is given by

$$(\mathbf{A} \circ \mathbf{B})_{i_1,\ldots,i_{p_1},j_1,\ldots,j_{p_2}} = a_{i_1,\ldots,i_{p_1}} b_{j_1,\ldots,j_{p_2}}; \tag{2.31}$$

The contracted product of two tensors is a generalization of the tensor-vector and the tensor-matrix multiplications discussed in the previous two subsections. The key distinction is that the modes should be multiplied and the ordering of the resulting modes is handled specially in the matrix and vector cases [21]. In this general case, let $\mathbf{A}$ be $(n_1, \ldots, n_m, d_1, \ldots, d_k)$-dimensional tensor and $\mathbf{B}$ be $(n_1, \ldots, n_m, g_1, \ldots, g_p)$-dimensional tensor. We can multiply both tensors along the first m modes [21], and the result is a $\mathbf{A}$ be $(d_1, \ldots, d_k, g_1, \ldots, g_p)$-dimensional tensor, given by

$$(< \mathbf{A}, \mathbf{B} >)_{i_1,\ldots,i_k,j_1,\ldots,j_p} = \sum_{l_1=1}^{n_1} \cdots \sum_{l_m=1}^{n_m} a_{l_1,\ldots,l_m,i_1,\ldots,i_k} b_{l_1,\ldots,l_m,j_1,\ldots,j_p}. \tag{2.32}$$

The remaining modes are ordered such that those from $\mathbf{A}$ come before $\mathbf{B}$.

The inner product of two tensors requires that both have equal dimensions [21]. Assuming both are $n_1, \ldots, n_p$-dimensional tensors, their inner product is given by

$$< \mathbf{A}, \mathbf{B} >= \sum_{i_1=1}^{n_1} \cdots \sum_{i_p=1}^{n_p} a_{i_1,\ldots,i_p} b_{i_1,\ldots,i_p} \tag{2.33}$$

Using this definition of inner product, the Frobenius norm of a tensor is then given by [21]

$$\|\mathbf{A}\|_F^2 =< \mathbf{A}, \mathbf{A} >= \sum_{i_1=1}^{n_1} \cdots \sum_{i_p=1}^{n_p} a_{i_1,\ldots,i_p}^2. \tag{2.34}$$

We have described all necessary terms and definitions, and we can proceed to the examination of two of the most widely recognized methods of tensor decomposition (CP and Tucker models). Both of these models involve some of the properties of SVD but are not identical to SVD in whole.

Chapter 3

Redundant Tensor Decompositions

This chapter has been prepared based on the publications of Prof. Brett Bader from The Sandia National Laboratories (CA\USA). Particularly, we refer the reader to his publication [21] to learn more comprehensive explanations on CP and Tucker's decomposition algorithms.

In this chapter we focus on two of the most widely known approaches for sum-of-rank-1 tensor decompositions. These are: (1) canonical decomposition (CANDECOMP) [30] or alternatively called parallel factor analysis (PARAFAC) [4]; (2) the Tucker [5] model. With the CP model, a tensor can be represented as a sum of rank-1 tensors with minimal number of bases in a unique fashion and by definition, the dimensionality of this basis set is the rank. The CP model does not constrain the geometry of the vectors that yield the rank-1 basis tensors; this is in contrast to the assumption of orthogonal vectors in SVD of matrices. While a matrix might be written as a sum of fewer unconstrained left-right vector products than prescribed by SVD, the orthogonality constraint on the geometry of the vectors has been found to be useful in many applications of SVD. As opposed to the CP, Tucker's proposed decomposition factors tensors as a finite sum assuming orthogonal vectors to generate the rank-1 basis tensors similar to SVD, but the result is not necessarily minimal. Two main properties of these decompositions are: first, they reparameterize the tensors with more variables than necessary (the number of variables describing decompositions is more than the number of free elements[1] in tensor), so we have redundancy in these models; second, these decomposition models do not provide explicit equations how to figure out the rank of tensor and we have to determine the rank on the basis of repeated decompositions.

[1] Degree of freedom of the tensor.

With the CP model, a tensor can be represented as a sum of rank-1 tensors in a unique fashion without any constrains and the dimensionality of this basis set is the rank:

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{u}_l^{(1)} \circ \mathbf{u}_l^{(2)} \circ \cdots \circ \mathbf{u}_l^{(p)} \tag{3.1}$$

where $\lambda = diag(\mathbf{\Lambda})$, $\mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \cdots \circ \mathbf{u}^{(p)}$ denotes the p-way outer-product of the vectors $\mathbf{u}_l^{(i)}$, and upper index (i), $i = 1, \ldots, p$ denotes the way of product. Figure 3.1 presents a schematic illustration of the CP model, where $\mathbf{U}^{(i)} = [\mathbf{u}_1^{(i)}, \ldots, \mathbf{u}_r^{(i)}]$, $i = 1, \ldots, p$ are the matrices whose columns form frames in n_i, $i = 1, \ldots, p$ dimensional spaces.

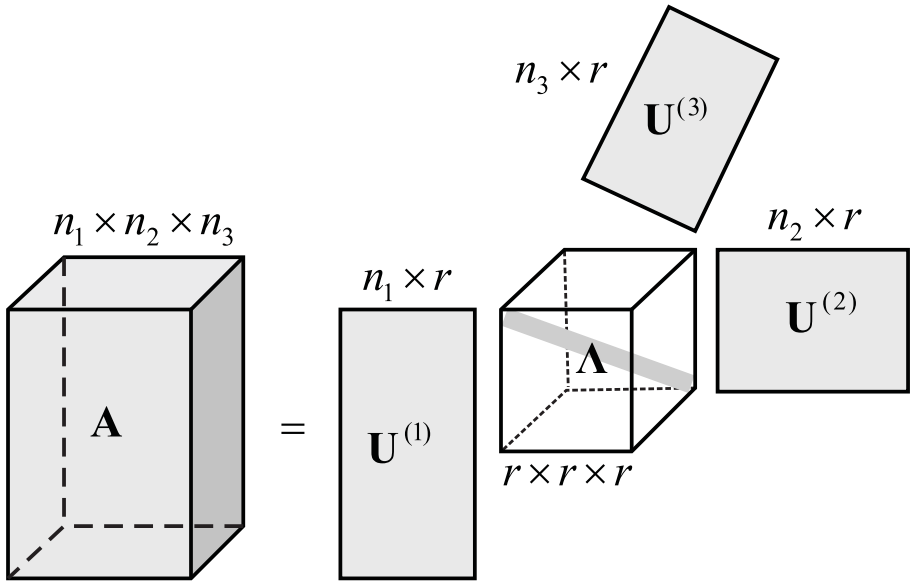


FIGURE 3.1: CP model of an order-3 (n_1, n_2, n_3)-dimensional tensor decomposition.

The CP model does not constrain the geometry of the vectors that yield the rank-1 basis tensors; this is in contrast with the assumption of orthogonal vectors in singular value decomposition (SVD) of matrices. While a matrix might be written as a sum of fewer unconstrained left-right vector products than prescribed by SVD, the orthogonality constraint on the geometry of the vectors has been found to be useful in many applications of SVD. As opposed to the CP, Tucker's proposed decomposition factors tensors as a finite sum assuming orthogonal vectors to generate the rank-1 basis tensors similar to SVD, but the result is not necessarily minimal; in fact it reparameterizes the tensors with more variables than necessary:

$$\mathbf{A} = \sum_{l_1=1}^{r_1} \sum_{l_2=1}^{r_2} \cdots \sum_{l_p=1}^{r_p} \mathbf{\Lambda}_{l_1, l_2, \ldots, l_p} \mathbf{u}_{l_1}^{(1)} \circ \mathbf{u}_{l_2}^{(2)} \circ \cdots \circ \mathbf{u}_{l_p}^{(p)} \tag{3.2}$$

An illustration of Tucker decomposition of an order-3 tensor is presented on Figure 3.2.

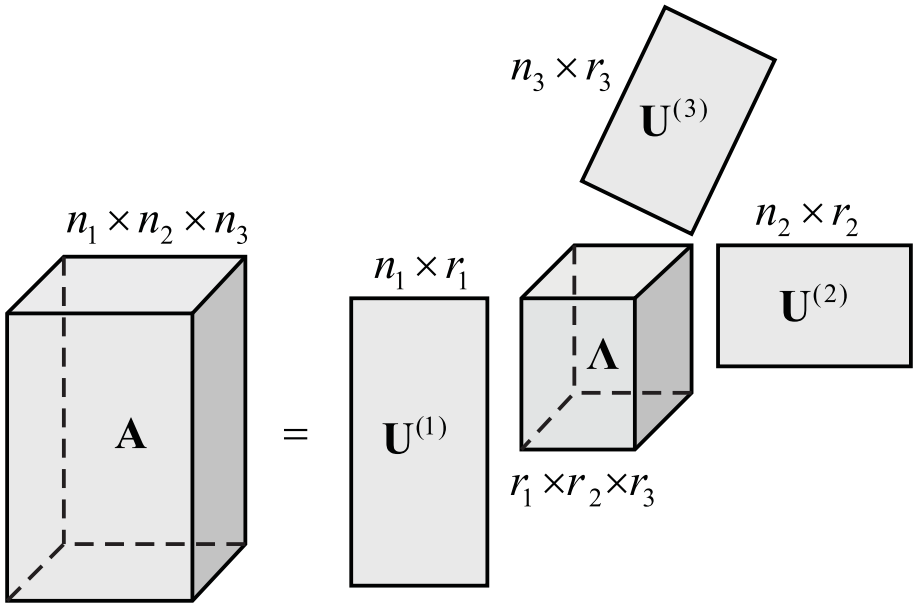


FIGURE 3.2: Tucker model of an order-3 (n_1, n_2, n_3)-dimensional tensor decomposition.

3.1 CANDECOMP/PARAFAC Decomposition

In 1927, Hitchcock [14] proposed the polyadic form of tensor, i.e., expressing a tensor as the sum of a finite number of rank-one tensors; and in 1944 Cattell [15] proposed ideas for parallel proportional analysis and the idea of multiple axes for analysis (circumstances, objects, and features) [1]. The concept finally became popular after its third introduction in 1970 to the psychometrics community, in the form of CANDECOMP (canonical decomposition) by Carroll and Chang [30] and PARAFAC (parallel factors) by [4] [1].

The CP decomposition factorizes a tensor into a weighted sum of rank-1 tensors, given by

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{U}_{:l}^{(1)} \circ \mathbf{U}_{:l}^{(2)} \circ \cdots \circ \mathbf{U}_{:l}^{(p)} \tag{3.3}$$

Here, λ is a vector of size r and each $\mathbf{U}^{(i)}$ is a matrix of size $n_i \times r$, for $i = 1, \ldots, p$.

For example, an order-3 tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is given and we wish to write it as

$$\mathbf{X} \approx \sum_{i=1}^{r} \lambda_i \mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \mathbf{u}_i^{(3)}, \tag{3.4}$$

where r is a positive integer and $\lambda \in \mathbb{R}^r$, $\mathbf{u}_i^{(1)} \in \mathbb{R}^{n_1}$, $\mathbf{u}_i^{(2)} \in \mathbb{R}^{n_2}$,and $\mathbf{u}_i^{(3)} \in \mathbb{R}^{n_3}$ for $i = 1, \ldots, r$. Elementwise, (3.4) is written as

$$x_{i_1 i_2 i_3} \approx \sum_{l=1}^{r} \lambda_l \mathbf{U}_{i_1 l}^{(1)} \mathbf{U}_{i_2 l}^{(2)} \mathbf{U}_{i_3 l}^{(3)}, \quad \text{for} \quad \begin{smallmatrix} i_1=1,\ldots,n_1; \\ i_2=1,\ldots,n_2; \\ i_3=1,\ldots,n_3; \end{smallmatrix} \tag{3.5}$$

This is illustrated on Figure 3.3.

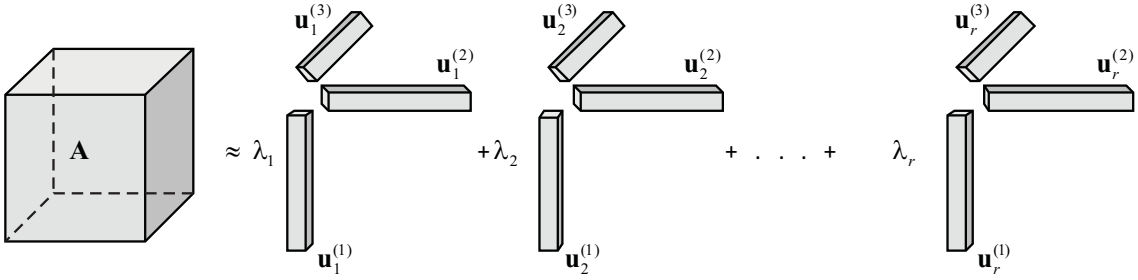


FIGURE 3.3: CP decomposition of an order-3 array.

The implementation of the higher-order power method [6] presented in Algorithm 1 is a multilinear generalization of the best rank-1 approximation problem for matrices [21]. This is also the same as the Alternating Least Squares algorithm for fitting a rank-1 CP model [21]. In the case of finding the best rank-r CP model, applying best rank-1 CP approximation to the residuals recursively (as on the matrix SVD) does not work. The best rank-1 approximation problem is when a tensor $\mathbf{A}$ is given, we want to find a $\mathbf{B}$ of the form

$$\mathbf{B} = \lambda \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \cdots \circ \mathbf{u}^{(p)}, \tag{3.6}$$

such that $\|\mathbf{A} - \mathbf{B}\|$ is as small as possible [21]. The higher-order power method computes a $\mathbf{B}$ that approximately solves this problem. Essentially, this method works as follows. It fixes all $\mathbf{u}$-vectors except $\mathbf{u}^{(1)}$, and then solves for the optimal $\mathbf{u}^{(1)}$, likewise for $\mathbf{u}^{(2)}$, $\mathbf{u}^{(3)}$, and so on, cycling through the indices until the specified number of iterations is exhausted [21].

Algorithm 1 Higher-order power method.

1: In: $\mathbf{A}$ of size $n_1 \times n_2 \times \cdots \times n_p$.
2: Out: $\mathbf{B}$ of size $n_1 \times n_2 \times \cdots \times n_p$, an estimate of the best rank-1 approximation of $\mathbf{A}$.
3: Compute initial values: Let $\mathbf{u}_0^{(k)}$ be the dominant left singular vector of $\mathbf{A}_{(k)}$ for $k = 2, \ldots, p$.
4: **for** $l = 0, 1, 2, \ldots$ (until converged) **do**
5: **for** $i = 1, \ldots, p$ **do**
6: $\tilde{\mathbf{u}}_{l+1}^{(i)} = \mathbf{A} \times_{-i} \{\mathbf{u}_l\}$
7: $\lambda_{l+1}^{(i)} = \|\tilde{\mathbf{u}}_{l+1}^{(i)}\|$
8: $\mathbf{u}_{l+1}^{(i)} = \|\tilde{\mathbf{u}}_{l+1}^{(i)}\| / \lambda_{l+1}^{(i)}$
9: **end for**
10: **end for**
11: Let $\lambda = \lambda_l$ and $\{\mathbf{u}\} = \{\mathbf{u}_l\}$, where l is the index of the final result of step (4).
12: Set $\mathbf{B} = \lambda \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \cdots \circ \mathbf{u}^{(p)}$.

3.2 Tucker Decomposition

The Tucker decomposition [5], also called a rank-$(r_1, r_2, \ldots, r_p)$ decomposition [6], is another way of summing decomposed tensors [21] and is given by

$$\mathbf{A} = \sum_{i_1=1}^{r_1} \sum_{i_2=1}^{r_2} \cdots \sum_{i_p=1}^{r_p} \lambda_{i_1, i_2, \ldots, i_p} \mathbf{U}_{:i_1}^{(1)} \circ \mathbf{U}_{:i_2}^{(2)} \circ \cdots \circ \mathbf{U}_{:i_p}^{(p)} \tag{3.7}$$

Here, $\mathbf{\Lambda}$ is a tensor of size $r_1 \times r_2 \times \cdots \times r_p$ itself, and each $\mathbf{U}^{(k)}$ is a matrix of size $n_k \times r_k$, for $k = 1, \ldots, p$. As before, the notation $\mathbf{U}_{:i}^{(k)}$ denotes the i^{th} column of the matrix $\mathbf{U}^{(k)}$. The tensor $\mathbf{\Lambda}$ is often called the "core array" or "core tensor" [21].

Thus, in the three-way case where $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we have

$$\mathbf{X} \approx \mathbf{\Lambda}^{(1)} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} = \sum_{i_1=1}^{r1} \sum_{i_2=1}^{r_2} \sum_{r=1}^{r_3} \lambda_{i_1 i_2 i_3} \mathbf{U}_{i_1}^{(1)} \circ \mathbf{U}_{i_2}^{(2)} \circ \mathbf{U}_{i_3}^{(3)}. \tag{3.8}$$

Here, $\mathbf{U}^{(1)} \in \mathbb{R}^{n_1 \times r_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{n_2 \times r_2}$, and $\mathbf{U}^{(3)} \in \mathbb{R}^{n_3 \times r_3}$ are the factor matrices (which are usually orthogonal) [1]. The entries of the core tensor $\mathbf{\Lambda} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ show the level of interaction between the different components [1].

Elementwise, the Tucker decomposition in 3.8 is

$$x_{i_1 i_2 i_3} \approx \sum_{l_1=1}^{r_1} \sum_{l_2=1}^{r_2} \sum_{l_3=1}^{r_3} \lambda_{l_1 l_2 l_3} \mathbf{U}_{i_1 l_1}^{(1)} \mathbf{U}_{i_2 l_2}^{(2)} \mathbf{U}_{i_3 l_3}^{(3)}, \quad \text{for } \begin{smallmatrix} i_1=1,\ldots,n_1; \\ i_2=1,\ldots,n_1; \\ i_3=1,\ldots,n_3; \end{smallmatrix} \tag{3.9}$$

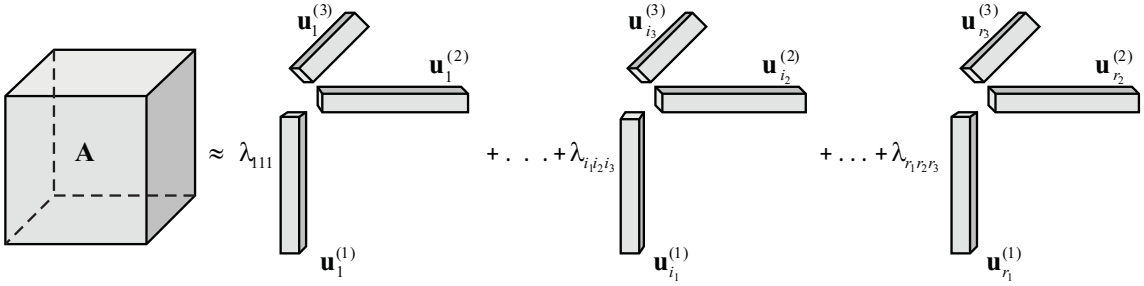


FIGURE 3.4: Tucker decomposition of a three-way array.

Here r_1, r_2, and r_3 are the numbers of components (i.e., columns) in the factor matrices $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$, respectively [1]. If r_1, r_2, r_3 are smaller than n_1, n_2, n_3, the core tensor $\mathbf{\Lambda}$ is considered as a compressed version of $\mathbf{X}$. The Tucker decomposition is illustrated on Figure 3.4 [1]. Most fitting algorithms assume that the factor matrices are columnwise orthonormal, but this is not required. In fact, CP can be viewed as a special case of Tucker where the core tensor is diagonal and $r_1 = r_2 = r_3$ [1].

The higher-order orthogonal iteration finds the best rank-$(r_1, r_2, \ldots, r_p)$ approximation of a higher-order tensor [21], Algorithm 2 in [6]. It is the multilinear generalization of the best rank-r approximation problem for matrices [21].

Algorithm 2 Higher-order orthogonal method.

1: In: $\mathbf{A}$ of size $n_1 \times n_2 \times \cdots \times n_p$.
2: Out: $\mathbf{B}$ of size $n_1 \times n_2 \times \cdots \times n_p$, an estimate of the best rank-$(r_1, r_2, \ldots, r_p)$ approximation of $\mathbf{A}$.
3: Compute initial values: Let $\mathbf{U}_0^{(k)} \in \mathbb{R}^{n_k \times r_k}$ be an orthogonal basis for the dominant r_k-dimensional left singular space of $\mathbf{A}_{(k)}$ for $k = 2, \ldots, p$.
4: **for** $l = 0, 1, 2, \ldots$ (until converged) **do**
5: **for** $i = 1, \ldots, p$ **do**
6: $\tilde{\mathbf{U}} = \mathbf{A} \times_{-i} \{\mathbf{U}_l^T\}$
7: Let $\mathbf{W}$ of size $n_i \times \mathbf{r}_k$ solve:
8: $\max \|\tilde{\mathbf{U}} \times_k \mathbf{W}^T\|$ subject to $\mathbf{W}^T\mathbf{W} = \mathbf{I}$
9: $\mathbf{U}_{l+1}^{(i)} = \mathbf{W}$
10: **end for**
11: **end for**
12: Let $\{\mathbf{U}\} = \{\mathbf{U}_l\}$, where l is the index of the final result of step (4).
13: Set $\lambda = \mathbf{A} \times \{\mathbf{U}^T\}$.
14: Set $\mathbf{B} = \lambda \times \{\mathbf{U}\}$.

3.3 Combination of CP and Tucker Models

Of late years several research institutions have offered different models that combine properties both CP and Tucker. As we know from previous explanation CP decomposition defines a tensor as the sum of rank-1 tensors. In proposed models, a tensor is determined as a sum of low-rank Tucker tensors, i.e., block decomposition. For instance, for an order-3 tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we have

$$\mathbf{X} = \sum_{l=1}^{r} \mathbf{\Lambda}_l \times_1 \mathbf{U}_l^{(1)} \times_1 \mathbf{U}_l^{(2)} \times_3 \mathbf{U}_l^{(3)}. \tag{3.10}$$

Here we assume that $\mathbf{\Lambda}_l$ is $(d_l^{(1)}, d_l^{(2)}, d_l^{(3)})$-dimensional tensors and $\mathbf{U}_l^{(k)}$ is $(n_k, d_l^{(k)})$-dimensional matrices with $l = 1, \ldots, r$ and $k = 1, \ldots, 3$. Figure 3.5 shows an example.

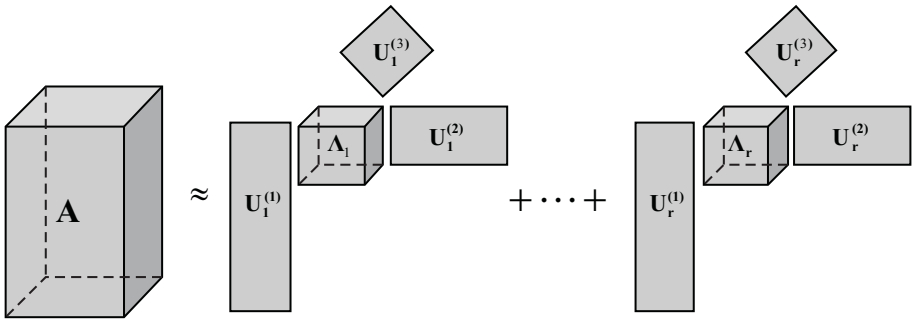


FIGURE 3.5: Block decomposition of an order-3 tensor.

The first authors which proposed a version of block decomposition [31, 32] called this algorithm PARALIND model. De Almeida at.al [33] gave an overview of some aspects of models of the form in (3.10) and their application to the problems in blind beamforming and multiantenna coding [1]. In a series of papers, De Lathauwer [29, 34] and Nion [35] explored a general class of decompositions as in 3.10, looking at their uniqueness properties, computational algorithms, and applications in wireless communications; see also [36]. Vasilescu and Terzopoulos [27] explored higher-order versions of ICA, a variation of PCA that, in some sense, rotates the principal components so that they are statistically independent [1]. An example of probabilistic ICA based in extended CP model described in [37] extended CP. Version of maximum-likelihood CP formulated in [38] and [39].

There are many other tensor decompositions, including INDSCAL, PARAFAC2, CANDELINC, DEDICOM, PARATUCK2, etc. [1], as well as non-negative variants of all of the above.

3.4 Non-Negative Tensor Decomposition

This section has been prepared based on the publications of Prof. Andrzej Cichocki from The Riken Brain Science Institute (Japan). Particularly, we refer the reader to his publication [19] to learn more comprehensive explanations on non-negative tensor decomposition.

Many real-world data are non-negative and the corresponding hidden components have a physical meaning only when non-negative [19]. In practice, both non-negative and sparse decompositions of data are often either desirable or necessary when the underlying components have a physical interpretation [19]. For example, in image processing and computer vision, involved variables and parameters may correspond to pixels, and non-negative sparse decomposition is related to the extraction of relevant parts from the images [40, 41].The reader can be also interested in examples of non-negative tensor factorization (NTF) in computer vision [42], application of NTF method to image data as an order-3 tensor [43], in [44] authors considered clustering based on non-negative factorizations of symmetric tensors, [45, 46] describe utilization of NTF for EEG signals treatment. A sparse representation of the data by a limited number of components is also an important research problem. In machine learning, sparseness is closely related to feature selection and certain generalizations in learning algorithms, while non-negativity relates to probability distributions [19].

Generally, compositional data are natural representations when the features are essentially the probabilities of complementary and mutually exclusive events. Moreover, NTF is an additive model which does not allow subtraction so it can be considered as a part-based representation in which a zero-value represents the absence and a positive number represents the presence of some event or component [19]. Specifically, in the case of facial image data, the additive or part-based nature of NTF has been shown to result in a basis of facial features, such as eyes, nose, and lips [19].

Below we present non-negative version of CP model. For more information we refer the reader to recently published book of Andrzej Cichocki et al. [19].

Let given an order-3 (n_1, n_2, n_3)-dimensional tensor $\mathbf{Y}$ which is decomposable by non-negative matrices $\mathbf{U}^{(i)},\ i = 1, \ldots, 3$ and positive index, r, defines the number of vectors in $U^{(i)}$. Then we can write an approximate decomposition of $\mathbf{Y}$ as

$$\mathbf{Y} = \sum_{j=1}^{r} \mathbf{u}_j^{(1)} \circ \mathbf{u}_j^{(2)} \circ \mathbf{u}_j^{(3)} + \mathbf{E}, \tag{3.11}$$

where the tensor $\mathbf{E}$ denotes decomposition error.

In order to compute the non-negative component matrices $\{\mathbf{U}\}$ we usually apply constrained optimization approach as by minimizing a suitable design cost function [19]. Typically, we minimize (with respect to the component matrices) the following global cost function

$$e = \|\mathbf{Y} - \mathbf{U}^{(1)} \circ \mathbf{U}^{(2)} \circ \mathbf{U}^{(3)}\|_F^2 + \alpha_1 \|\mathbf{U}^{(1)}\|_F^2 + \alpha_2 \|\mathbf{U}^{(2)}\|_F^2 + \alpha_3 \|\mathbf{U}^{(3)}\|_F^2, \tag{3.12}$$

subject to non-negativity constraints, where $\alpha_i,\ i = 1, \ldots, 3$ are non-negative regularization parameters.

The most popular approach for solving this optimization problem is to apply the ALS technique. In this approach we compute the gradient of the cost function with respect to each individual component matrix (assuming that the others are fixed and independent) [19]:

$$\begin{aligned}
\nabla e(U^{(1)}) &= -\mathbf{Y}_{(1)}(\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)}) + \mathbf{U}^{(1)}[((\mathbf{U}^{(3)})^T\mathbf{U}^{(3)}) \circledast ((\mathbf{U}^{(2)})^T\mathbf{U}^{(2)}) + \alpha_1\mathbf{I}], \\
\nabla e(U^{(2)}) &= -\mathbf{Y}_{(2)}(\mathbf{U}^{(3)} \odot \mathbf{U}^{(1)}) + \mathbf{U}^{(2)}[((\mathbf{U}^{(3)})^T\mathbf{U}^{(3)}) \circledast ((\mathbf{U}^{(1)})^T\mathbf{U}^{(1)}) + \alpha_2\mathbf{I}], \\
\nabla e(U^{(3)}) &= -\mathbf{Y}_{(3)}(\mathbf{U}^{(2)} \odot \mathbf{U}^{(1)}) + \mathbf{U}^{(3)}[((\mathbf{U}^{(2)})^T\mathbf{U}^{(2)}) \circledast ((\mathbf{U}^{(1)})^T\mathbf{U}^{(1)}) + \alpha_3\mathbf{I}].
\end{aligned} \tag{3.13}$$

By equating the gradient components to zero and applying the nonlinear projection to maintain non-negativity of components we obtain efficient and relatively simple non-negative ALS update rules for the NTF [19]:

$$\begin{aligned}
\mathbf{U}^{(1)} &= \left[\mathbf{Y}_{(1)}(\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)})[((\mathbf{U}^{(3)})^T\mathbf{U}^{(3)}) \circledast ((\mathbf{U}^{(2)})^T\mathbf{U}^{(2)}) + \alpha_1\mathbf{I}]^{-1}\right]_+, \\
\mathbf{U}^{(2)} &= \left[\mathbf{Y}_{(2)}(\mathbf{U}^{(3)} \odot \mathbf{U}^{(1)})[((\mathbf{U}^{(3)})^T\mathbf{U}^{(3)}) \circledast ((\mathbf{U}^{(1)})^T\mathbf{U}^{(1)}) + \alpha_2\mathbf{I}]^{-1}\right]_+, \\
\mathbf{U}^{(3)} &= \left[\mathbf{Y}_{(3)}(\mathbf{U}^{(2)} \odot \mathbf{U}^{(1)})[((\mathbf{U}^{(2)})^T\mathbf{U}^{(2)}) \circledast ((\mathbf{U}^{(1)})^T\mathbf{U}^{(1)}) + \alpha_3\mathbf{I}]^{-1}\right]_+,
\end{aligned} \tag{3.14}$$

where symbols $\odot$ and $\circledast$ denote the Khatri-Rao and Hadamard products, respectively. The operation $[u]_+$ means taking non-negative value, $max(0, u)$.

The main advantage of ALS algorithms is high convergence speed and its scalability for large-scale problems [19].

3.5 Redundancy and Cardinality

Let us note the main two differences on the above described models: CP model has a diagonal tensor of linear combination coefficients $\mathbf{\Lambda}$ and arbitrary directed vectors in frames $\mathbf{U}^{(i)}$, on the other hand, Tucker model has fully filled tensor of linear combination coefficients $\mathbf{\Lambda}$ and orthogonal vector frames $\mathbf{U}^{(i)}$. When one does analysis/decomposition/transformation of data he[2] can say that his method does not loose data when, at least, his method preserves cardinality of data. Cardinality or cardinal number of a set of scalars is the number, c, of free elements (degree of freedom) in a set (in our case it is a tensor), e.g. the cardinal number of any full-rank symmetric two dimensional matrix equals three. Indeed, any reparametrization (or coordinate transformation) must preserve the cardinality (or the number of dimensions or degrees of freedom).

TABLE 3.1: Cardinal properties of existing models of tensor decomposition.

	Matrix SVD	CP model	Tucker model
Cardinality of data	Matrix $\mathbf{A}[2 \times 2]$, $c = 4$	Tensor $\mathbf{B}[2\times2\times2]$, $c = 8$	Tensor $\mathbf{B}[2\times2\times2]$, $c = 8$
Properties of model	Diagonal $\mathbf{\Lambda}$, orthogonal $\mathbf{U}^{(1)}$ & $\mathbf{U}^{(2)}$	Diagonal $\mathbf{\Lambda}$, non-orthogonal $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, & $\mathbf{U}^{(3)}$	Non-diagonal $\mathbf{\Lambda}$, orthogonal $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, & $\mathbf{U}^{(3)}$
Cardinality of model	d.o.f. of $\mathbf{\Lambda} = 2$, d.o.f. of $\mathbf{U}^{(1)}$ & $\mathbf{U}^{(2)} = 2$, $c = 4$	d.o.f. of $\mathbf{\Lambda} = r$, d.o.f. of $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, & $\mathbf{U}^{(3)} = 3r$, $c = 4r$	d.o.f. of $\mathbf{\Lambda} = r_1r_2r_3$, d.o.f. of $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, & $\mathbf{U}^{(3)} = 3$, $c = 3 + r_1r_2r_3$

In the Table 3.1 we present a brief analysis of described models and compare their cardinality with SVD. Let us use a random (2,2)-dimensional matrix $\mathbf{A}$ and (2,2,2)-dimensional tensor $\mathbf{B}$. Cardinal numbers of $\mathbf{A}$ and $\mathbf{B}$ equal 4 and 8 respectively. In the general case, to describe the SVD of $\mathbf{A}$ we need two singular values, $r = 2$, and, because of orthogonality of vector frames $\mathbf{U}^{(i)}$, two rotation angles to define the orientations of orthogonal vector frames in 2-dimensional space. So we can see that SVD preserves cardinality of decomposed data. For tensor $\mathbf{B}$, CP models hold

[2]We use "he", "she" and similar third person phrases to include all genders, and wish to avoid s/he type make-shift solutions that are necessitated by the English language.

r linear combination coefficients and, due to the absence of any restriction on vector frame structure, $3r$ rotational angles. Supposing that CP model decomposed tensor $\mathbf{B}$ with rank $r = 1$, in this case, CP model does not preserve cardinality, $c = 4$. For the case of $r = 2$ cardinality of CP model, $c = 8$, equals cardinality of tensor $\mathbf{B}$. But the same rank, $r = 2$, we have for matrix $\mathbf{A}$ that is order-2 data. We assume that this case is just a coincidence, because more complicated data must have a higher rank. For rank $r = 3$ and above cardinality of CP model is always higher than cardinality of tensor $\mathbf{B}$. Tucker model, due to orthogonality of $\mathbf{U}^{(i)}$, holds 3 rotation angles for each way of $\mathbf{B}$ and $r_1 r_2 r_3$ linear coefficients. In any case of values of r_1, r_2, r_3 Tucker model does not preserve cardinality of tensor $\mathbf{B}$ at all. We need to note that when we apply CP and Tucker models to matrix $\mathbf{A}$ we get the same results as for SVD. So CP and Tucker models marginally preserve cardinality of data in case when we apply them to matrices and do not do it in the general case.

The main goal of this work is to determine a decomposition model that preserves cardinality of any-order any-dimensionality data. Below we investigate a standard decomposition of symmetric and non-symmetric matrices and extend their properties for our decomposition model. In the first part of the next chapter we provide a model for the decomposition of real-valued symmetric tensors and then we extend our approach to the non-symmetric case for any dimensions and orders.

Chapter 4

Non-redundant Tensor Decomposition

In this chapter we present an alternative decomposition model that utilizes a set of geometrically constrained vectors to generate rank-1 tensor bases; we refrain from calling these basis vectors 'eigen' or 'singular' because at this time we have not proven any invariance property that would warrant this nomenclature. The constraint relaxes the orthogonality assumption in matrix SVD which turns out to be a one-to-one reparameterization of the tensor; however, we cannot also claim that the identified decomposition is minimum rank in CP-sense. The proposition is a generalization of the orthogonal coordinate frame interpretation of the singular vectors of non-symmetric real-valued matrices, which remain as order-2 special cases. The corresponding special case of decomposition model for symmetric tensors was proposed and solved for using iterative random search presented here [47]. Iterative descent technique for the optimization of the vector frame rotation parameters and corresponding linear combination coefficients is based on Levenberg-Marquardt method for Givens rotation angles [48]. In the first part we provide a model for decomposition of real-valued symmetric tensors and then we extend our approach to non-symmetric case for any dimensions and orders.

4.1 Decomposition of a Symmetric Tensor

The decomposition of a real-valued symmetric tensor $\mathbf{A}$ into a sum of rank-1 tensors utilizes basis tensors that are p-way outer-products of the same vector (referred to

as rank-1 symmetric tensors) [1, 2]:

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{u}_l^{\circ p} \tag{4.1}$$

where $\mathbf{u}^{\circ p}$ denotes the p-way outer-product of the vector $\mathbf{u}$.

Tensor decomposition problem is fundamental to the extension of subspace analysis techniques in signal processing that arise from the study of second order statistics of vector-valued measurements to higher order statistics. Existing examples of such applications include blind source separation. For instance, an exponential multivariate family as a signal model can be factorized using a sum-of-rank-1 tensor decomposition; consider an n-variate order-p polynomial $q(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x}^{\circ p} = \sum_{l_1=0}^{n} \cdots \sum_{l_p=0}^{n} \mathbf{A}_{l_1,\ldots,l_p} \cdot \mathbf{x}_{l_1} \cdot \ldots \cdot \mathbf{x}_{l_p}$ where $x_0 = 1$. If the (symmetric) tensor $\mathbf{A}$ containing these polynomial coefficients is decomposed into the desired form, then the polynomial can be written as $\mathbf{A} \cdot \mathbf{x}^{\circ p} = \sum_{l=1}^{r} \lambda_l (\mathbf{u}_l^T \mathbf{x})^{\circ p}$, and an exponential density $e^{q(\mathbf{x})}$ can be factorized into a product of univariate exponentials. Other applications are reviewed in [1] and include finding polynomial factorizations [49, 50].

4.1.1 Order-2 n-Dimensional Symmetric Tensors

A symmetric n-dimensional order-2 tensor is a symmetric matrix. Eigenvector bases for the real symmetric matrices are orthogonal, and can be always transformed to orthonormal basis. Figure 4.1 illustrates the eigendecomposition of a matrix. Thus a real n-dimensional symmetric matrix can be decomposed as

$$\mathbf{A} = \sum_{l=1}^{n} \lambda_l \mathbf{u}_l^{\circ 2} \tag{4.2}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ is the matrix in which columns form an orthogonal frame in n-dimensional space.

For numerical eigendecomposition of (4.2) we can use, for instance, the Jacobi algorithm [51] that tries to find $q = \binom{n}{2}$ rotation angles $\{\theta_k, k = 1, \ldots, q\}$, such that we can construct a rotation matrix $\mathbf{R}(\theta_k)$ in plane $(i, j)\{i = 1, \ldots, n-1, j = i+1, \ldots, n\}$ with angle θ_k (with a one-to-one correspondence between the indices k and (i, j) in this Givens angle parameterization). This eigendecomposition solution consists of q rotation angles and n eigenvalues. The number of free elements of a symmetric

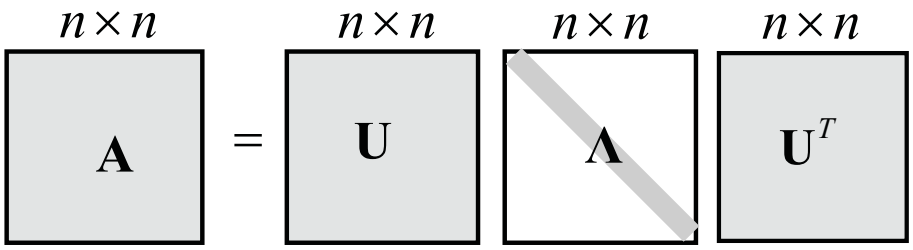


FIGURE 4.1: Decomposition of order-2 n-dimensional symmetric tensor.

n-dimensional matrix $\mathbf{A}$, $m_f(n,2) = n(n+1)/2$, equals the sum $n+q$. Consequently, eigendecomposition is simply a reparameterization procedure. Solving for the rotation matrices $\mathbf{R}(\theta_k)$, we can get the orthonormal eigenvectors given by $\mathbf{U}$:

$$\mathbf{U} = \prod_{k=1}^{q} \mathbf{R}(\theta_k), \quad \mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I} \tag{4.3}$$

Due to orthonormality of $\mathbf{U}$, the eigenvalues are uniquely identified by the Frobenius inner-product vector between the target matrix and the basis matrices [1]:

$$\langle \mathbf{u}_l^{\circ 2}, \mathbf{A} \rangle = \sum_{i=1}^{n} \lambda_i \langle \mathbf{u}_l^{\circ 2}, \mathbf{u}_i^{\circ 2} \rangle = \sum_{i=1}^{n} \lambda_i (\mathbf{u}_l^T \mathbf{u}_i)^2 = \lambda_l \tag{4.4}$$

4.1.2 Order-p 2-Dimensional Symmetric Tensors

Let $\mathbf{A}$ be a 2-dimensional order-p real symmetric tensor. In a 1-1 reparameterization, the number of linear combination coefficients r, plus the number of parameters that characterize r corresponding vectors s, must be equal to the number of free elements in the tensor[1] (i.e., its total dimensionality); that is $(r+s)=(p+1)$, since order-p 2-dimensional symmetric tensors have $(p+1)$ free entries.

Incorporating these conditions into the design of the rank-1 sum decomposition on the right-hand side of (4.1), we obtain that real-symmetric, 2-dimensional order-p tensor $\mathbf{A}$ has the following decomposition:

$$\mathbf{A} = \sum_{l=1}^{p} \lambda_l \mathbf{u}_l^{\circ p}, \quad \mathbf{u}_l = \begin{bmatrix} cos(\theta + (l-1)\pi/p) \\ sin(\theta + (l-1)\pi/p) \end{bmatrix} \tag{4.5}$$

[1] We do not refer to these coefficients and vectors as eigen until some suitable invariance property is proven.

Figure 4.2 shows a graphical illustration of the decomposition of an order-3 2-dimensional symmetric tensor $\mathbf{A}$.

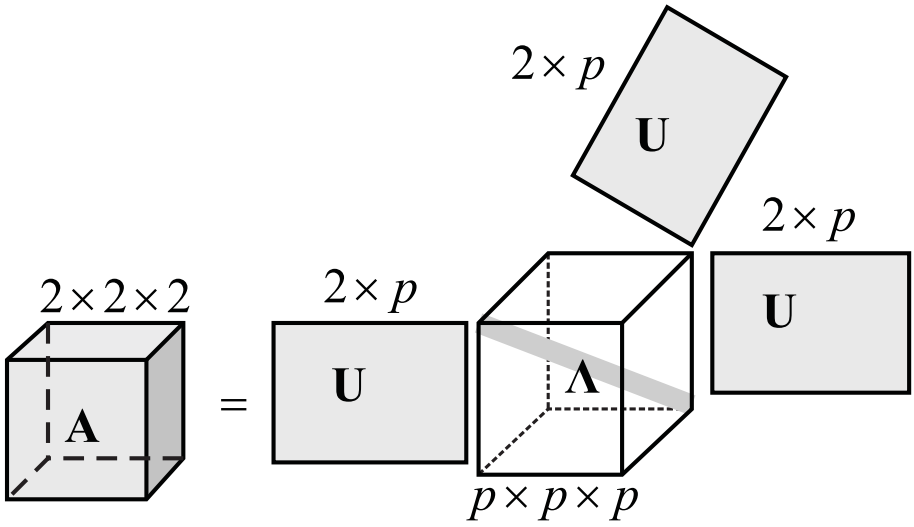


FIGURE 4.2: Decomposition of order-p 2-dimensional symmetric tensor, $r = p$.

In this case, a simple line search for θ in the interval $[0, \pi/p)$ is sufficient to fit optimally the decomposition to the tensor with zero error. Employing Gram-Schmidt orthogonalization, the linear combination coefficient vector $\boldsymbol{\lambda}$ is uniquely identified by the inner-product matrix between the basis rank-1 symmetric tensor pairs and the inner-product vector between the target tensor and the basis tensors; i.e., at the optimal decomposition, $\boldsymbol{\lambda} = \mathbf{B}^{-1}\mathbf{c}(\boldsymbol{\theta})$. Here the matrix $\mathbf{B}$ (invariant with respect to θ, since the pairwise angles between the basis vectors leading to the basis rank-1 symmetric tensors are fixed by the frame) and the vector $\mathbf{c}$ are defined elementwise as follows, assuming Frobenius tensor inner product as in (4.4):

$$B_{ij} = \langle \mathbf{u}_i^{\circ p}, \mathbf{u}_j^{\circ p} \rangle_F = \langle \mathbf{u}_i, \mathbf{u}_j \rangle_F^p = (\mathbf{u}_i^T \mathbf{u}_j)^p, \; c_i(\boldsymbol{\theta}) = \langle \mathbf{u}_i^{\circ p}, \mathbf{A} \rangle \tag{4.6}$$

where $i, j = 1, \ldots, p$. Specifically note that each entry of $\mathbf{B}$ reduces to the following: $B_{ij} = \cos^p((i-j)\pi/p)$. For symmetric matrices, this matrix is simply identity, $\mathbf{B} = \mathbf{I}$.

4.1.3 Order-p n-Dimensional Symmetric Tensors

The number of free elements of a symmetric n-dimensional order-p tensor is given by: $m_f(n, p) = \binom{n+p-1}{p}$. General structure of vector frames for order-p n-dimensional tensor is presented in next section but based on the two special cases examined above we can conclude that the decomposition of any symmetric tensor can consist

of some fixed frame of vectors rotated in n-dimensional space and any angle between pairwise vectors can be constant and depends on order p. As in matrices, we need q rotation angles to decompose any symmetric n-dimensional order-p tensor as a finite sum of rank-1 tensors as in (4.1). The number of vectors in this decomposition is: $r = m_r(n,p) = \binom{n+p-1}{p} - \binom{n}{2}$. On Figure 4.3 we present a graphical illustration for decomposition of order-3 n-dimensional symmetric tensor $\mathbf{A}$.

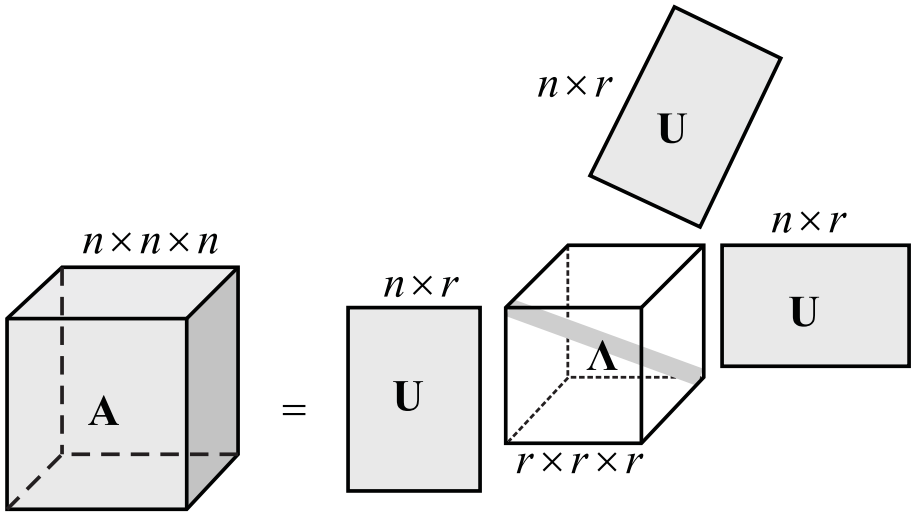


FIGURE 4.3: Decomposition of an order-p n-dimensional symmetric tensor.

To obtain the decomposition numerically, we construct a frame of r initial vectors placed in columns of a matrix $\mathbf{F}$ and optimize the q rotation angles $\boldsymbol{\theta}$ such that the Frobenius norm of the error tensor is minimized (to zero). In the spirit of block coordinate descent and fixed point algorithms, for a given candidate frame orientation, the linear combination coefficients are always obtained using (4.6) and $\boldsymbol{\lambda} = \mathbf{B}^{-1}\mathbf{c}(\boldsymbol{\theta})$.

The optimization is done iteratively and in a fixed point manner, updating the linear combination coefficients and updating the rotation matrix for the frame of vectors in order to minimize the error tensor Frobenius norm. At each iteration, basis vectors are expressed as (all rotations multiply from left):

$$\mathbf{U} = \left(\prod_{k=1}^{q} \mathbf{R}(\theta_k) \right) \mathbf{F} \tag{4.7}$$

4.2 Vector Frames and Uniform Distribution of Points on the Hypersphere

As stated above, the solution of a symmetric tensor decomposition consists of normalized vectors, $\mathbf{U}$, and corresponding them linear combination coefficients, λ. In this section we do not consider or analyze linear coefficients but focus on the vector frame for symmetric tensor decompositions. From the case of CP decomposition of an order-p n-dimensional symmetric tensor we know that such a decomposition does not have any restrictions of vector frame. In other words, vectors can form random points on the hypersphere, Appendix A. So for the large number of vectors the rank of a tensor can be approximated with some error as hypersphere in n-dimensional space. To be more precisely, to approximate a hypersphere we need to involve as well an inversion of the vectors, $-\mathbf{U}$. Imagine we have an decomposition of an order-2 2-dimensional symmetric tensor. The vector frame such a decomposition and its inversion part form a perfect square inscribed into a circle. The solution of Tucker model always provides us an orthogonal vector frame such that the vectors form a regular polyhedra in n-dimensional space.

For the proposed non-redundant decomposition we need to form a vector frame, $\mathbf{F}$, as an initial step of the decomposition algorithm. In the cases when at least one of the parameters of a tensor, n or p, is equal to 2 we do not have any problems: the vectors are either orthogonal frame in n-dimensional space or invariant to π/p 2-dimensional frame.

Based on the cases of vector frames for order-2 n-dimensional and order-p 2-dimensional symmetric tensors, where separation angles between closest vectors are $\pi/2$ and π/p, respectively, we can conclude that structure of any vector frame $\mathbf{F}$ must maximize minimal distance between vectors [52]. This issue is equivalent to the problem of uniform distribution of points on the hypersphere [53]. Below we describe existing methods of uniform distribution of points and after we present our own one.

4.2.1 Uniformly Distributing Points on the Hypersphere

This section has been prepared based on the publications of Prof Paul Leopardi and Prof. Rob Womersley from The University of New South Wales. Particularly, we refer the reader to their publications [54, 55] to learn more comprehensive explanations

on uniformly distributing points on the hypersphere.

The problem of distributing points uniformly on the hypersphere appears in such fields viral morphology, crystallography, molecular structure, electrostatics, polynomial approximation, interpolation and integration over the sphere, etc., where we need to approximate some spherical surfaces. In real life we have a deal with 3-dimensional unit sphere S^2 which is the set of points $\mathbf{f}_i,\ i = 1, \ldots, m$, in $\mathbb{R}^3$ such that the distance $|\mathbf{f}_i| = 1$ from the origin. The notation S^2 means we are considering data on the surface of a ball. In contrast to the circle, it is not possible to equally distribute points on the hypersphere except in the case of the platonic solids, see Appendix B. In the case when $m \to \infty$ to solve this task one can apply some optimization criteria: minimum energy, covering, packing, Voronoi cells, volume of their convex hull, maximum determinant, cubature weights and norms of the Lagrange polynomials. Most of these criteria are rotationally invariant, that is a rotation of the whole set of points leaves the objective function unchanged. Thus the final solution of point sets can consists of $\mathbf{F}$ and $-\mathbf{F}$. Existence of many local minima is a property of optimization problems on the hyperspere. With the exception of some cases, it is not proven that a set of points is the global minimum. To read more and see 3-dimensional illustrations we refer the reader to [54, 55].

Convex Hull, Voronoi Cells and Delaunay Triangulation. Given a set of points on the unit hypersphere one quantity of interest is the volume of the convex hull of the set of points [56]. The convex hull is the set of all points which lies on a line segment joining two points in the set. It is also the intersection of all half spaces containing all the points. Maximum convex hull points maximize the volume of their convex hull, which lies inside the unit hypersphere. The dual of the maximum convex hull problem is to arrange the points so that the tangent planes to the hypersphere at these points encloses a polyhedron of minimum volume. The geodesic distance between two points $\mathbf{x}$, $\mathbf{y}$ on the unit hypersphere is $d(\mathbf{x}, \mathbf{y}) = cos^{-1}(\mathbf{x}^T\mathbf{y})$. Using this geodesic distance, a Voronoi cell [57] V_j associated with a point $\mathbf{f}_j$ is the set of all points on the unit hypersphere which are closer to $\mathbf{f}_j$ than any of the other points. The Delaunay triangulation considers that no other point is contained in the circumcircle of any triangle. The Voronoi cells are spherical polygons, see Appendix B.

Riesz s-Energy. The Riesz s-energy ($s > 0$) of a set of m points $\mathbf{f}_j,\ j = 1, \ldots, m$ on the unit hypersphere is the sum over all pairs of distinct points of the terms $1/|x_i - x_j|^s$. The standard Coulomb potential used to model electrons repelling

each other corresponds to $s = 1$. Finding point sets which minimize the Coulomb potential is known as the Thomson problem [58]. The limit of Riesz s-energy, as s grows larger, corresponds to the packing problem [53]. For $s = 0$ one maximizes the logarithm of the sum of the distances $|\mathbf{f}_i - \mathbf{f}_j|$, which is equivalent to maximizing the product of the distances. The minimum energy problem is to find a set of m points on the unit hypersphere which minimize their Riesz energy.

Covering and Packing with Spherical Caps. A spherical cap $C(\mathbf{f}, c)$ centred at $\mathbf{f}$ with radius c is the set of all points on the hypersphere a distance of at most c from the point $\mathbf{f}$. For a fixed number of points m, the packing problem is to find the centers $\mathbf{f}_j, j = 1, \ldots, m$ of m identical non-overlapping spherical caps so that their common radius is maximized. This packing radius is two times smaller than angle between the points. The set of points is called a spherical code [53]. The problem of finding a point set to maximize the minimum angle between the points is known as Tammes or Toth problem [59, 60]. On the other hand the covering problem is to find the centers $\mathbf{f}_j, j = 1, \ldots, m$ of m identical spherical caps which completely cover the hypersphere so that their common radius is minimized. This covering radius is given by the distance of the furthest point on the hypersphere from the closest point in the set $\mathbf{f}_j, j = 1, \ldots, m$, which is also known as the mesh norm. The mesh norm can be calculated using the Voronoi cells or Delaunay triangulation. Another measure of the equidistribution of a point set is the ratio (always greater than 1) of the covering radius to the packing radius. To read more on this problem we refer the reader to the Dissertation of [54].

Norms of the Lagrange Polynomials. Spherical polynomials, that is polynomials in three variables restricted to the surface of the hypersphere, of degree at most p form a space of dimension $d_p = (p+1)^2$ [61]. The most common basis is the spherical harmonics [62]. A set of d_p points $\mathbf{f}_j$ on the hypersphere forms a fundamental system if a polynomial of degree at most p. If this polynomial is equal to zero at the points than it must be equal to zero at every points on the hypersphere. Given a fundamental set, the Lagrange polynomials $l_j(\mathbf{x})$ for $j = 1, \ldots, d_p$ are the polynomials of degree p such that $l_j(\mathbf{f}_i) = 1$ if $i = j$ and 0 for all i not equal to j. The norms of the Lagrange polynomials are of fundamental interest in approximation theory. The Lagrangian sum of squares is the sum of squares of the Lagrange polynomials, while the infinity norm is the maximum of the absolute value of the Lagrange polynomials.

Interpolatory Cubature, Cubature Weights and Determinants. The integral of a function $\phi(\mathbf{f})$ over the hypersphere can be approximated by the weighted

(weights w_j) sum of the values of the function values $\phi(\mathbf{f}_j)$ at the points $\mathbf{f}_j$ for $j = 1, \ldots, m$. An interpolatory cubature rule has $m = d_p$ points and is exact for all spherical polynomials of degree not higher than p. For a fundamental system the weights in an interpolatory cubature rule are given by $w_j = \int_{S^2} l_j(\mathbf{f})$. As such cubature rules must be exact for the constant polynomial, the sum of the cubature weights w_j must be equal to the area $|S^2|$ of the hypersphere [63]. The linear system that must be solved to find the cubature weights can be so ill-conditioned that a computed solution is totally unreliable. The linear system is well conditioned if the points are chosen to maximize the determinant of a basis matrix [64].

Form this short overview of uniformly distributing points on the hypersphere we can see that there is no an ideal and uniquely agreed-upon approach for such a problem. Below we describe our own approximation method that is alike to the interpolatory determinants but grounded on tensor non-redundant properties.

4.2.2 Vector Frames for Symmetric Tensors

In this section, we propose a geometrically constrained basis vector frame that yields a set of rank-1 symmetric tensor bases that, when rotated appropriately, is able to attain a sum-of-rank-1 decomposition of any order, any dimensional symmetric tensor. The number of variables that parameterizes the proposed decomposition is equal to the number of free elements (dimensions) in the symmetric tensor.

The assumption that angle between neighbor vectors in n-dimensional space must be equal π/p is not true in general and can be utilized just for order-2 n-dimensional and order-p 2-dimensional tensors, because even for 3-dimensional space there are exactly 5 convex polyhedra (called Platonic solids) with equal distances between neighbor vectors (or vertices) [65]. For the case of dimensions $n > 3$ there are just three solids with equal distance between neighbor vertexes: cube, octahedron, and simplex.

The vector frame $\mathbf{F}$ consists of r vectors $\mathbf{f}_i, i = 1, \ldots, r$, defined as its n-dimensional columns. Based on the examined uniform distributions of points on the hypersphere and properties of non-redundant tensor decomposition we can conclude that structure of any vector frame $\mathbf{F}$ must maximize minimal distance between vectors. There is no analytic solution for this task so we propose to solve this problem iteratively minimizing the squared error (SE):

$$e^2 = \langle \mathbf{B} - \mathbf{I}, \mathbf{B} - \mathbf{I} \rangle_F = \sum_{i=1}^{r} \sum_{j=1}^{r} (\mathbf{B}_{ij} - \mathbf{I}_{ij})^2, \tag{4.8}$$

where as in (4.6) $\mathbf{B}_{ij} = (\mathbf{u}_i^T \mathbf{u}_j)^p = (\mathbf{f}_i^T \mathbf{f}_j)^p$.

Vectors $\mathbf{f}_i$ in frame $\mathbf{F}$ can be defined, for instance, in hyperspherical coordinate system with constant radial values equal 1. Thus all diagonal entries in $\mathbf{B}$ equal 1 and all non-diagonal entries describe distances between vectors or distances between points on hypersphere of unit radius. So minimization of (4.8) leads to maximization of minimal distances between vectors. Solution for minimization of (4.8) will be described below in Section 4.6.5. Figure 4.4 presents results of proposed algorithm for vector frames of different dimensions and orders.

We can note that proposed algorithm gives us Platonic solids (convex polyhedra with maximal possible constant distance between nearest vectors) as particular cases of vector frames and empirically we figured out that for any vector frame exist planes such that frame vectors on those planes separated by π/p radian. This optimization procedure for frame construction gives us r vectors, which yield a full-rank $\mathbf{B}$ in (4.6), and do not engage an elimination procedure as in our previous work that used an overcomplete frame [47].

It is possible to build a vector frames without numerical optimization for marginal cases of tensors, when p or $n = 2$. We propose to use the following approach: the frame consists of vectors that are recursive rotations of, for instance, the first column of the n-dimensional identity matrix multiplied by a rotation matrix with an angle of π/p in consecutive dimension index pairs. Specifically, this leads to a system of r vectors, placed in columns of a matrix $\mathbf{F}$, denoted by $\mathbf{f}_i$, using the Algorithm 3, where $\mathbf{R}_{l,l+1}(\pi/p)$ is a rotation matrix in the plane $(l, l+1)$, $l = 1, \ldots, n$ with angle π/p.

Algorithm 3 Vector frame for the cases of $p = 2$ or $n = 2$.

In: The first vector $\mathbf{f}_1 = [1\, 0\, \ldots\, 0]^T$ of size in $\mathbf{F}$.
Out: A system of r vectors $\mathbf{F}$, $i = 1, \ldots, r$, defined as columns of $\mathbf{F}$.
for $i = 2 : r$ **do**
  for $l = mod(i + n - 3, n - 1) + 1$ **do**
    $\mathbf{f}_i = R_{l,l+1}(\pi/p)\mathbf{f}_{i-1}$.
  end for
end for

The results of an application of such an algorithm are in phase with the results on Figure 4.4(a),(b). Note that the number of vectors in this frame is the same as

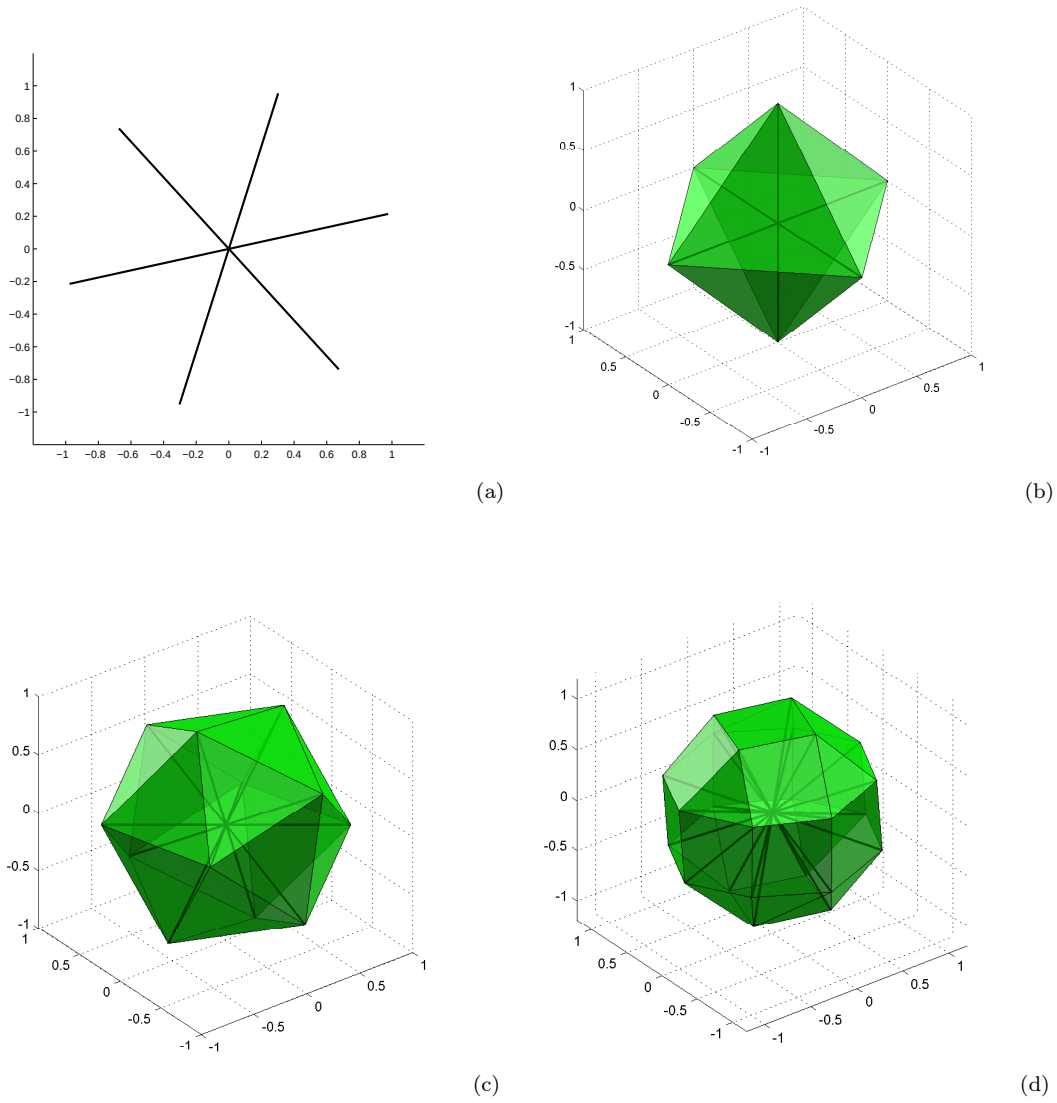


FIGURE 4.4: Vector frames for (a) 2-dimensional order-3 tensor with 3 unique vectors, (b) 3-dimensional order-2 tensor with 3 unique vectors, (c) Platonic solid (icosahedron with 6 unique vectors, which does not correspond to any order-p 3-dimensional tensor), and (d) 3-dimensional order-4 tensor with 12 unique vectors.

the number of vectors needed: r, which yields a full-rank $\mathbf{B}$ in (4.6). This iterative procedure for frame construction gives us the result just for r steps and does not engage an elimination procedure as in our first work that used an overcomplete frame [47].

4.3 Decomposition of a Non-Symmetric Tensor

The decomposition of a non-symmetric tensor $\mathbf{A}$ into a sum of rank-1 tensors utilizes basis tensors that are p-way outer-products of different vectors (referred to as rank-1 non-symmetric tensors) [1, 2]:

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{u}_l^{(1)} \circ \mathbf{u}_l^{(2)} \circ \cdots \circ \mathbf{u}_l^{(p)} \tag{4.9}$$

where $\mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \cdots \circ \mathbf{u}^{(p)}$ denotes the p-way outer-product of the vectors $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(p)}$ and upper index (i), $i = 1, \ldots, p$ denotes the way of product.

4.3.1 Order-2 (n_1,n_2)-Dimensional Non-Symmetric Tensors

A non-symmetric (n_1,n_2)-dimensional order-2 tensor is a non-symmetric matrix and we have to deal with the common case of SVD. Singular vector bases for non-symmetric matrices are always selected orthogonal, and can always be made into an orthonormal basis. Thus a real full-rank $(n_1,\ n_2)$-dimensional non-symmetric matrix can be decomposed as

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{u}_l^{(1)} \circ \mathbf{u}_l^{(2)}, \tag{4.10}$$

where $r = \min(n_1,n_2)$, $\mathbf{U}^{(1)} = [\mathbf{u}_1^{(1)}, \ldots, \mathbf{u}_r^{(1)}]$ and $\mathbf{U}^{(2)} = [\mathbf{u}_1^{(2)}, \ldots, \mathbf{u}_r^{(2)}]$ are the matrices where columns form orthogonal frames in n_1 and n_2 dimensional spaces, Figure 4.5.

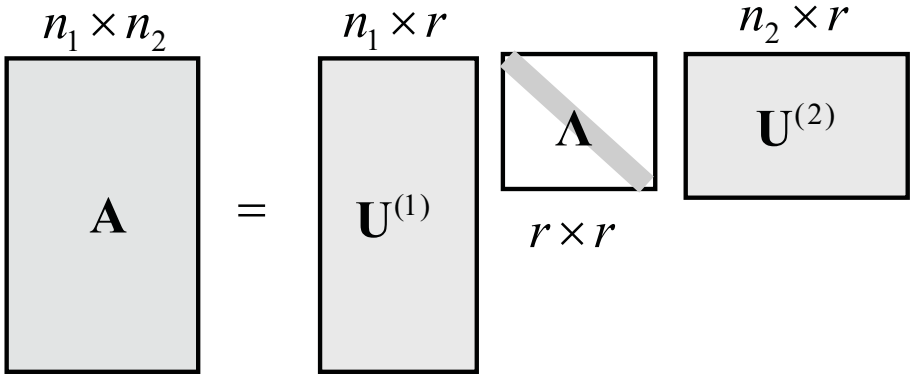


FIGURE 4.5: Decomposition of order-2 (n_1, n_2)-dimensional non-symmetric tensor.

For numerical determination of (4.10) we can use, as in symmetric case, the Jacobi algorithm [51] for each vector frame that tries to find $q_1 + q_2$ rotation angles $\{\theta^{(1)}_{k_1}, \theta^{(2)}_{k_2}; k_1 = 1, \ldots, q_1; k_2 = 1, \ldots, q_2\}$, such that we can construct rotation matrices $\mathbf{R}(\theta^{(1)}_{k_1})$ in plane (i, j) $\{i = 1, \ldots, n_1 - 1; j = i + 1, \ldots, n_1\}$ with angle $\theta^{(1)}_{k_1}$ and $\mathbf{R}(\theta^{(2)}_{k_2})$ respectively. Due to cross influence of rotation matrices, $q_1 = \binom{n_1}{2} - \binom{n1-n2}{2}$ and $q_2 = \binom{n_2}{2} - \binom{n2-n1}{2}$, where $\binom{n}{k} = 0$ if $k > n$. This singular decomposition solution consists of $q = q_1 + q_2$ rotation angles and r singular values. The number of free elements of a non-symmetric (n_1, n_2)-dimensional matrix $\mathbf{A}$ equals the product of its dimensions so $r = n_1 \cdot n_2 - q$ that satisfies definition of the rank in linear algebra where $r = min(n_1, n_2)$. Again we can see that singular decomposition is simply a reparameterization procedure. Solving for the rotation matrices $\mathbf{R}(\theta^{(1)}_{k_1})$ and $\mathbf{R}(\theta^{(2)}_{k_2})$ we can get the orthonormal singular vectors given by $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$:

$$\mathbf{U}^{(i)} = \prod_{k_i=1}^{q_i} \mathbf{R}(\theta^{(i)}_{k_i}),\ (\mathbf{U}^{(i)})^T\mathbf{U}^{(i)} = \mathbf{U}^{(i)}(\mathbf{U}^{(i)})^T = \mathbf{I}, i = 1, 2; \tag{4.11}$$

Due to orthonormality of $\mathbf{U}^{(i)}$, the singular values are uniquely identified by the Frobenius inner-product between the target matrix and the basis matrices [1]:

$$\lambda_l = \langle \mathbf{u}^{(1)}_l \circ \mathbf{u}^{(2)}_l, \mathbf{A} \rangle_F = \sum_{i=1}^{r} \lambda_i \langle \mathbf{u}^{(1)}_l \circ \mathbf{u}^{(2)}_l, \mathbf{u}^{(1)}_i \circ \mathbf{u}^{(2)}_i \rangle_F. \tag{4.12}$$

Described combination of vectors $\{(1 \circ 1), (2 \circ 2), \ldots, (r \circ r)\}$ in (4.10) is not unique [48]. We can construct $r!$ combinations of vectors on the basis of permutation matrices [66] and any of them can be utilized for (n_1,n_2)-dimensional order-2 non-symmetric tensor decomposition. Figure 4.6(a) shows two possible combinations of singular values based on 2-dimensional order-2 permutation matrices. Here the permutation matrix is an order-2 r-dimensional binary-matrix that has exactly one entry of 1 in each row and column and 0's elsewhere. In case of symmetry n-dimensional order-2 tensor vectors can be arranged only in ascending order that is the initial form of permutation.

In general, decomposition of an order-2 (n_1,n_2)-dimensional tensor $\mathbf{A}$, Fig 4.5, can consists of (n_1, n_1)-dimensional matrix $\mathbf{U}^{(1)}$, (n_2, n_2)-dimensional matrix $\mathbf{U}^{(2)}$, and (n_1, n_2)-dimensional matrix of singular values $\mathbf{\Lambda}$. Where singular elements in $\mathbf{\Lambda}$ are located by selecting a permutation matrix. Figure 4.6(b) shows all possible combinations of singular values based in (3,2)-dimensional order-2 matrix $\mathbf{\Lambda}$.

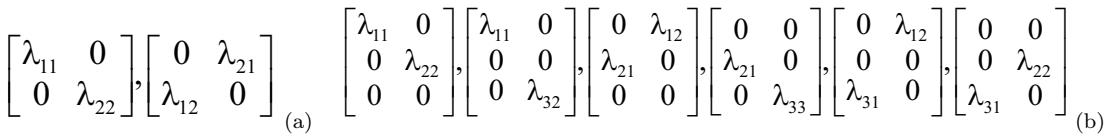


FIGURE 4.6: Permutation matrices $\mathbf{\Lambda}$: a) possible conformations of order-2 (2,2)-dimensional matrix, b) possible conformations of order-2 (3,2)-dimensional matrix.

4.3.2 Order-p 2-Dimensional Non-Symmetric Tensors

Let $\mathbf{A}$ be a 2-dimensional order-p real non-symmetric tensor. In a 1-1 reparameterization, the number of linear combination coefficients, r, plus the number of parameters that characterize r corresponding vectors, s, must be equal to the number of free elements in the tensor; that is $(r + s) = 2^p$, since the number of free elements in order-p 2-dimensional non-symmetric tensors equals to product of dimensions. The number of rotation angles equals p. So the rank of such a tensor is $r = 2^p - p$, where $r \geq p$.

From the expression of the rank for this case of tensors we see that the rank r is more than its order p and from case of symmetric 2-dimensional order-p tensor we know that for each way of tensor we can construct just p vectors. Incorporating these conditions into the design of the rank-1 sum decomposition on the right-hand side of (4.9), we obtain non-symmetric 2-dimensional order-p tensor $\mathbf{A}$ which can be expressed as:

$$\mathbf{A} = \sum_{l=1}^{r} \lambda_l \mathbf{u}_l^{(1)} \circ \mathbf{u}_l^{(2)} \circ \cdots \circ \mathbf{u}_l^{(p)}; \mathbf{u}_l^{(i)} = \begin{bmatrix} cos(\theta^{(i)} + (k_l^{(i)} - 1)\pi/p) \\ sin(\theta^{(i)} + (k_l^{(i)} - 1)\pi/p) \end{bmatrix}; \begin{matrix} i = 1, \ldots, p \\ k_l^{(i)} \in \{1, \ldots, p\} \end{matrix} \tag{4.13}$$

Based on the case of symmetric order-p 2-dimensional tensors and non-symmetric order-2 (n_1,n_2)-dimensional tensors examined above we conclude that the combinations of vectors $\{(k_l^{(1)}, k_l^{(2)}, \ldots, k_l^{(p)})\}$ for rank-1 tensors in (4.13) must be chosen on the basis of a permutation tensor. Here the permutation tensor is an order-p p-dimensional binary-tensor that has exactly one entry of 1 in each way and 0's elsewhere, Figure 4.7.

In this case, a simple line search for each $\theta^{(i)}$ in the interval $[0, \pi/p)$ is sufficient to fit optimally the decomposition to the tensor with zero error. Employing Gram-Schmidt orthogonalization, the linear combination coefficient vector $\boldsymbol{\lambda}$ is uniquely identified

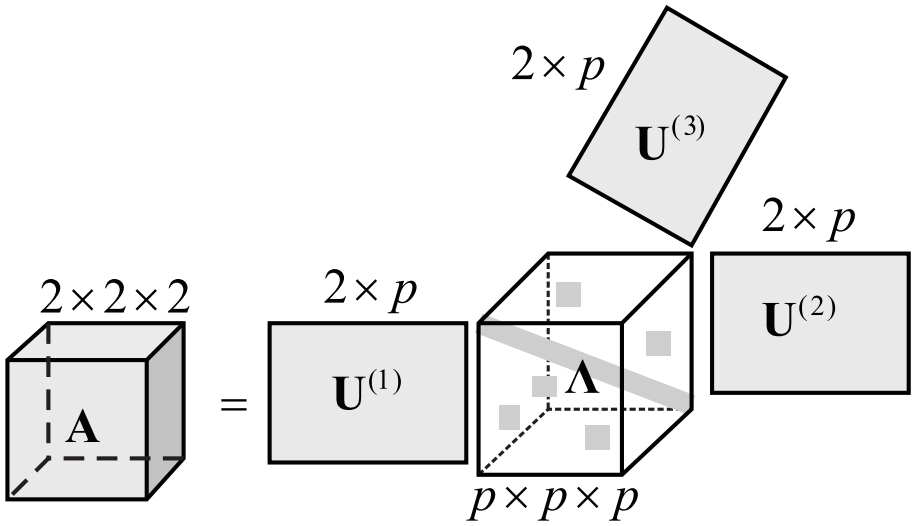


FIGURE 4.7: Decomposition of order-3 $(2, 2, 2)$-dimensional non-symmetric tensor.

by the inner-product matrix between the basis rank-1 symmetric tensor pairs and the inner-product vector between the target tensor and the basis tensors; i.e., at the optimal decomposition, $\boldsymbol{\lambda} = \mathbf{B}^{-1}\mathbf{c}(\boldsymbol{\Theta})$, where $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(p)}]$. Here the matrix $\mathbf{B}$ (invariant with respect to $\boldsymbol{\theta}^{(i)}$, since the pairwise angles between the basis vectors leading to the basis rank-1 symmetric tensors are fixed by the frame) and the vector $\mathbf{c}$ are defined elementwise as follows, assuming Frobenius tensor inner product as in (4.6):

$$\begin{aligned} B_{ij} &= \langle \mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(p)}, \mathbf{u}_j^{(1)} \circ \mathbf{u}_j^{(2)} \circ \cdots \circ \mathbf{u}_j^{(p)} \rangle_F \\ c_i(\boldsymbol{\Theta}) &= \langle \mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(p)}, \mathbf{A} \rangle_F \end{aligned} \tag{4.14}$$

where $i, j = 1, \ldots, r$.

4.3.3 Order-p $(n_1, n_2, \ldots, n_p)$-Dimensional Non-Symmetric Tensors

The number of free elements of a non-symmetric $(n_1, n_2, \ldots, n_p)$-dimensional order-p tensor is given by: $m_f(\mathbf{n}, p) = n_1 n_2 \cdots n_p$, where $\mathbf{n} = [n_1, n_2, \ldots, n_p]$. Based on the two special non-symmetric cases examined above we conclude that the decomposition of any non-symmetric tensor can consist of some fixed frames of vectors rotated along each way in their dimensionality spaces and any angle between pairwise vectors must be a constant and depends on the order p. As in the case of non-symmetric matrices, we need $q = q_1 + q_2 + \cdots + q_p$ rotation angles. Due to cross influence of rotation

matrices:

$$q_i = \binom{n_i}{2} - \binom{\tilde{n}_i}{2}; \tilde{n}_i = n_i - \prod_{j=1, j\neq i}^{p} n_j. \tag{4.15}$$

For the case of non-redundant decomposition the rank of the the tensor is evaluated as a difference between the number of free elements of the tensor and the number of required rotational angles. So the rank of the non-symmetric tensor is

$$r = n_1 n_2 \cdots n_p - \sum_{i=1}^{p} q_i, \tag{4.16}$$

and the rank of the symmetric tensor is

$$r = \binom{n+p-1}{p} - \binom{n}{2}. \tag{4.17}$$

To obtain the decomposition numerically, we construct p frames each of $m_r(n_i, p)$ initial vectors $\mathbf{F}^{(i)}, i = 1, \ldots, p$ and optimize the rotation angles $\theta^{(i)}$ such that the Frobenius norm of the error tensor is minimized (to zero). In the spirit of block coordinate descent and fixed point algorithms, for a given candidate frame orientation, the linear combination coefficients are always obtained using (4.14) and $\boldsymbol{\lambda} = \mathbf{B}^{-1}\mathbf{c}(\boldsymbol{\Theta})$, basis initial frames are expressed as in Section 4.2.2. As in the previous case we need to choose r vectors on the basis of an order-p $(r_1, r_2, \ldots, r_p)$-dimensional permutation tensor $\mathbf{Q}$, where $r_i = m_r(n_i, p),\ i = 1, \ldots, p$. So at each iteration, basis vectors are expressed as:

$$\mathbf{U}^{(i)} = \left(\prod_{k=1}^{q} \mathbf{R}(\theta_k^{(i)}) \right) \mathbf{F}^{(i)}. \tag{4.18}$$

So any order-p $(n_1, \ldots, n_p)$-dimensional non-symmetric tensor $\mathbf{A}$ decomposable by r factors can be written, with respect to an order-p $(r_1, \ldots, r_p)$-dimensional permutation tensor $\mathbf{Q}$, as a finite sum of rank-1 tensors:

$$\mathbf{A} = \sum_{i=1}^{r} \lambda_i \mathbf{u}^{(1)}_{q_i^{(1)}} \circ \cdots \circ \mathbf{u}^{(p)}_{q_i^{(p)}}, \tag{4.19}$$

where parameters $q_i^{(j)},\ i = 1, \ldots, r,\ j = 1, \ldots, p$ denote non-zero entries of permutation tensor $\mathbf{Q}$, Figure 4.8. In the case of symmetry of the tensor $\mathbf{A}$ parameter $q_i^{(j)}$ transforms to the linear index i, the tensor $\mathbf{\Lambda}$ becomes cubical and diagonal, and vectors $\mathbf{u}_i^{(j)},\ j = 1, \ldots, p$ are equal to each other with respect to j, i.e., $\mathbf{u}_i^{(1)} = \mathbf{u}_i^{(2)} = \cdots = \mathbf{u}_i^{(p)}$. This equation is common for both symmetric and non-symmetric any-order any-dimensional tensors.

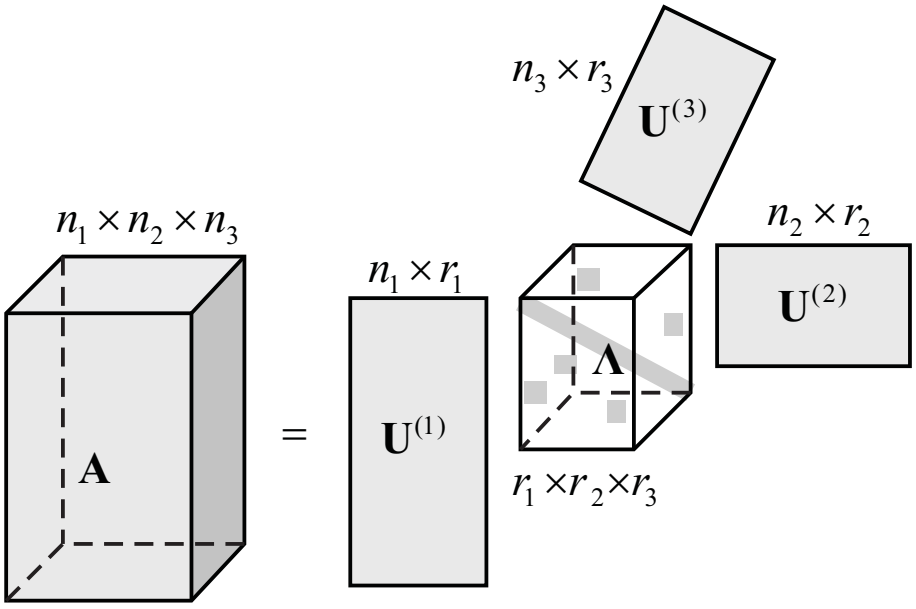


FIGURE 4.8: Decomposition of order-3 (n_1, n_2, n_3)-dimensional non-symmetric tensor.

4.4 Marginal Cases: Decomposition of Scalar and Vector

Up to this section we have investigated all possible cases of higher-order multidimensional tensors, so the order of a tensor $\mathbf{A}$, $p \geq 2$, and the minimal values of dimensions in such a tensor, $n_i \geq 2,\ i = 1, \ldots, p$. To check whether some theoretical approach is universal we need to investigate it in all possible cases of data representation without any restrictions. Let's return to our initial terms and definitions which describe the tensor and operations:

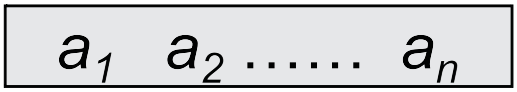
- A tensor - multidimensional or p-way array of scalars.
- The order of a tensor is the number of non-singleton dimensions.
- We make decomposition when we need to analyze order-p data with $p \geq 2$.

This list of basic definitions has two limitations: "non-singleton" and "with $p \geq 2$" in the second and the third rows, respectively. Without them any tensor can be presented in the infinite possible forms. For instance, an order-2 (n_1, n_2)-dimensional tensor $\mathbf{A}$ can be considered as an order-3 $(n_1, n_2, 1)$-dimensional tensor. The second restriction bounds the set of analyzed data, i.e., we do not decompose vectors and scalars. Without limitations we have the following list:

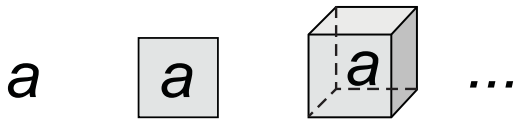
- A tensor - multidimensional or p-way array of scalars.
- The order of a tensor is the number of dimensions.
- We make decomposition when we need to analyze order-p data.

Below we will consider decomposition of a vector and a scalar.

Let's decompose an order-2 $(1, n)$-dimensional vector $\mathbf{a}$, Figure 4.9(a). From the conception of non-redundant tensor decomposition we know that the cardinal number, c, of data set must be equal to the number of variables which describe decomposition of such a data set. The number of variables consists of the sum of the numbers of rotation angles, $q_i,\ i = 1, \ldots, p$, and the rank of a tensor, r. In this case, $c = n$ and $q_1 = \binom{1}{2} - \binom{1-n}{2} = 0$, $q_2 = \binom{n}{2} - \binom{n-1}{2} = n-1$. So the rank of a vector is always equal to 1, $r = c - q_1 - q_2$. As a conclusion we can say that non-redundant decomposition



(a) an order-2 $(1, n)$-dimensional vector $\mathbf{a}$.



(b) a scalars a as an order-1, -2, -3,... tensor.

FIGURE 4.9: Marginal cases of tensor decompositions.

of a vector simply transforms data set to the hyper-spherical coordinate system, see Appendix A.

Resuming the above, if there are no limitations on the main definitions, a scalar can be presented as a 1-dimensional vector (order-1 tensor) or a $(1, 1)$-dimensional matrix (order-2 tensor), or $(1, 1, 1)$-dimensional cubic (order-3 tensor), etc, Figure 4.9(b). For any case of representations the number of rotation angles for such tensors is always equal to 0. So the rank of a scalar, $r = 1$, and non-redundant decomposition maps scalars without alternations.

On the basis of this analysis we can conclude that proposed non-redundant decomposition preserves cardinal properties for any cases of dimensions and orders. In case of singleton dimensions we do not decompose data but just transform them to the hyper-spherical space.

4.5 Non-Negative Non-Redundant Tensor Decomposition

Non-negative decomposition allows us to analyze data in their "native" form, e.g. non-negative factorization of a set of images results the sum of rank-1 tensors so the result of the sum does not nullify any parts of the factors. In the Section 3.4 we described non-negative form of classical CP model which can be applied to non-negative matrix factorization as well. One of the most awkwardness part of that algorithm is the operation of taking non-negative value, $[u]_+ = max(0, u)$. On each iteration of that approach we have to lose some data and try to get new ones without negative entries. It is similar to fitting the model with adjusted random walk. In this section we present our approach and its analysis for non-redundant tensor decomposition transformed to non-negative form. The main idea of it is the same as for the described non-redundant one but with one restriction : all linear coefficients λ as well as all entries of vector frame $\mathbf{U}$ must be non-negative, i.e., stand in positive sector of coordinate system. We do not employ the operator $[u]_+$ and perform optimization procedure as it was described above.

Let an order-p $(n_1, \ldots, n_p)$-dimensional non-negative tensor $\mathbf{A}$ is given, i.e., $a_{i_1,\ldots,i_p} \geq 0$, which is decomposable as a rank-r tensor with non-negative values in $\mathbf{U}^{(i)}$, $i = 1, \ldots, p$, and λ. With respect to an order-p $(r_1, \ldots, r_p)$-dimensional permutation tensor, $\mathbf{Q}$, we can write an approximate decomposition of $\mathbf{A}$ as in (4.19):

$$\mathbf{A} = \sum_{i=1}^{r} \lambda_i \mathbf{u}^{(1)}_{q_i^{(1)}} \circ \cdots \circ \mathbf{u}^{(p)}_{q_i^{(p)}} + \mathbf{E}, \tag{4.20}$$

where the tensor $\mathbf{E}$ denotes decompose error. The tensor $\mathbf{E}$ can contain any possible factors of $\mathbf{A}$ with negative inclusions.

As in the case of non-redundant decomposition, we will start to examine an order-p 2-dimensional non-negative symmetric tensor and then we will extend our approach to the general case of the non-negative tensor.

Let $\mathbf{A}$ be an order-3 2-dimensional non-negative symmetric tensor. From the non-redundant decomposition of such a tensor we know that all neighbor vectors are separated by $\pi/3$ radian. The range of such a vector frame is from 0 to $(\pi - \pi/3)$ radians and solution is a rotation angle θ that lies in the range $[0, \pi/3)$ radians, Figure 4.10(left). We see that such the vector frame, in confidence with its negative

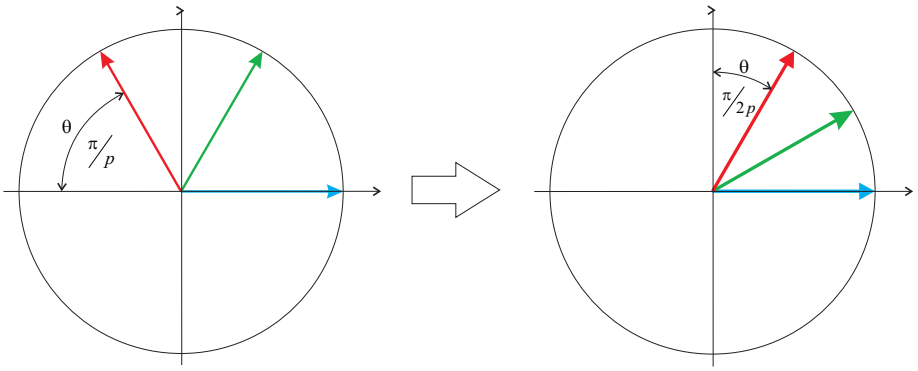


FIGURE 4.10: Frame vector reorganization for the non-negative non-redundant decomposition.

part, $-\mathbf{F}$, uniformly covers a unit circle, i.e., all range of $\mathbb{R}$. To restrict our solutions to non-negative form, $\mathbb{R}_+$, we propose to bound the range of the vector frame, $\mathbf{F}$, in half, i.e., from 0 to $(\pi/2 - \pi/6)$, Figure 4.10(right). Thereby we have a vector frame consists of 3 vectors and invariant to rotation angle $pi/6$ and the rotation angle θ lies in the range $[0, \pi/6)$ radians. New non-negative vector frame is the half of the frame for the general model of an order-6 2-dimensional tensor. Suppose we need to implement the non-negative decomposition of an order-3 rank-2 tensor $\mathbf{A}$. Involving the idea from the previous example we need to build the vector frame for an order-4 3-dimensional tensor and utilize the vectors from positive sector only. For this case of the tensor the vector frame consists of 12 vectors and full hypersphere does of 24. The 3-dimensional hypersphere has 8 sectors with unique sign of coordinate coefficients. So in the positive sector we have 3 required non-negative vectors.

Generalizing these two types of tensors we can conclude that to get the vector frame, $\mathbf{F}$, for an order-p n-dimensional non-negative symmetric tensor we utilize a $2^{(n-1)}$-th of the vectors for an order-$(2p)$ n-dimensional common symmetric tensor. Such an approach always gives us the exact number of required vectors, r, in positive sector of n-dimensional space and neighbor vectors are separated by $\pi/(2p)$ radian.

Till the vector frame stay in positive sector any projection of $\mathbf{A}$ on the vectors $\mathbf{u}_i, i = 1, \ldots, r, < \mathbf{A}, \mathbf{u}_i^{\circ p} >$ is also non-negative value. But to get linear coefficients λ we also utilize the inverse matrix $\mathbf{B} = (\mathbf{F}\mathbf{F}^T)^{\cdot p}$, where some of entries are negative values, so the final values of λ can be negative.

Actually, Monte Carlo experiments (which randomly generated each entry of the non-negative tensor from the absolute value of a normal distribution with mean zero and standard deviation one) discovered that in the case when the tensor $\mathbf{A}$ in not

absolutely factorizable in non-negative form we have non-zero value of the residual $\mathbf{E}$ in (4.20).

The non-negative non-redundant decomposition of an order-p $(n_1, \ldots, n_p)$-dimensional tensor is a combination of the expressed non-negative symmetric case and idea of non-symmetric decomposition which is presented in the Section 4.3.3.

Optimization for such a problem is implemented in the same way as for general case of the tensor and is presented in the following section. Performing non-negative decomposition we involve both vector frame $\mathbf{F}$ and its inversion $-\mathbf{F}$ such that at any moment we always have the required number of vectors, r, in positive sector of n-dimensional space.

4.6 Gradient Models and Numerical Results

In this section we present the gradient model of the Frobenius norm of the decomposition error that depends on rotation matrices in Cartesian space and gradient based optimization of the vector frame for symmetric tensors defined in hyperspherical coordinate system.

4.6.1 Squared Frobenius Norm of Decomposition Error

Here we focus on the Frobenius norm of the error (simply referred to as the error from now on) between the actual tensor $\mathbf{A}$ and its current decomposition estimate $\mathbf{T}$: $e = \|\mathbf{A}-\mathbf{T}\|_F^2$. The decomposition of the tensor is achieved by iteratively minimizing the squared error (SE):

$$e^2 = \langle \mathbf{A}-\mathbf{T}, \mathbf{A}-\mathbf{T}\rangle = \sum_{l_1=1}^{n_1}\cdots\sum_{l_p=1}^{n_p}(\mathbf{A}_{l_{1\cdots p}} - \mathbf{T}_{l_{1\cdots p}})^2. \tag{4.21}$$

Due to the higher order polynomials involved, we do not have a closed form solution for the rotation angles that yield $e(\boldsymbol{\theta}) = 0$; therefore we will employ an iterative numerical optimization methods. For simpler notation, SE in (4.21) can be reformulated in matrix form using the usual vectorization operator $vec(\cdot)$ that returns a vector whose elements are taken *column*-wise starting from the right-side operand. So a target tensor $\mathbf{T}$ in vector form is $\mathbf{t} = vec(\mathbf{T})$ and the vectorized i^{th} rank-1 non-symmetric basis tensor obtained from the corresponding basis vector is $\mathbf{x}_i = vec(\mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(p)})$. Collecting $\mathbf{x}_i$ in the columns of $\mathbf{X}$ and defining $\mathbf{e} = \mathbf{t} - \mathbf{X}\boldsymbol{\lambda}$ and $\boldsymbol{\lambda} = \mathbf{B}^{-1}\mathbf{X}^T\mathbf{t}$, SE takes the form $e^2 = \mathbf{e}^T\mathbf{e}$. The vectorized error tensor is equivalently:

$$\mathbf{e} = (\mathbf{I} - \mathbf{X}\mathbf{B}^{-1}\mathbf{X}^T)\mathbf{t} \tag{4.22}$$

and SE with $\mathbf{B} = \mathbf{X}^T\mathbf{X}$ and $\mathbf{X}\mathbf{B}^{-1}\mathbf{X}^T = \mathbf{I}$ (after some algebra to simplify the expression):

$$e^2 = \mathbf{t}^T(\mathbf{I} - \mathbf{X}\mathbf{B}^{-1}\mathbf{X}^T)\mathbf{t} \tag{4.23}$$

The gradient of SE with respect to the rotation angles $\boldsymbol{\theta}^{(m)}$ is:

$$\nabla e^2(\theta_k^{(m)}) = -\mathbf{t}^T([\partial\mathbf{X}/\partial\theta_k^{(m)}]\mathbf{B}^{-1}\mathbf{X}^T + \mathbf{X}\mathbf{B}^{-1}[\partial\mathbf{X}^T/\partial\theta_k^{(m)}])\mathbf{t} \tag{4.24}$$

The k^{th} column of the Jacobian matrix $\mathbf{J}$ for $\boldsymbol{\theta}^{(m)}$ is:

$$\mathbf{j}_k(\theta^{(m)}) = \partial \mathbf{e}/\partial \theta_k^{(m)} = -([\partial \mathbf{X}/\partial \theta_k^{(m)}]\mathbf{B}^{-1}\mathbf{X}^T + \mathbf{X}\mathbf{B}^{-1}[\partial \mathbf{X}^T/\partial \theta_k^{(m)}])\mathbf{t} \tag{4.25}$$

The Hessian matrix for SE consists of the following entries:

$$\mathbf{H}_{kl}^{(m)}(\theta^{(m)}) = -\mathbf{t}^T \left\{ \begin{array}{c} \frac{\partial^2 \mathbf{X}}{\partial \theta_k^{(m)} \partial \theta_l^{(m)}}\mathbf{B}^{-1}\mathbf{X}^T + \frac{\partial \mathbf{X}}{\partial \theta_k^{(m)}}\mathbf{B}^{-1}\frac{\partial \mathbf{X}^T}{\partial \theta_l^{(m)}} + \\ \frac{\partial \mathbf{X}}{\partial \theta_l^{(m)}}\mathbf{B}^{-1}\frac{\partial \mathbf{X}^T}{\partial \theta_k^{(m)}} + \mathbf{X}\mathbf{B}^{-1}\frac{\partial^2 \mathbf{X}^T}{\partial \theta_k^{(m)} \partial \theta_l^{(m)}} \end{array} \right\} \mathbf{t}. \tag{4.26}$$

The i^{th} column of $\mathbf{X}$ is obtained by vectorizing the rotated i^{th} frame vectors $\{\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \ldots, \mathbf{f}_i^{(p)}\}$ according to

$$\mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(p)} = (\mathbf{R}(\boldsymbol{\theta}^{(1)})\mathbf{f}_i^{(1)}) \circ (\mathbf{R}(\boldsymbol{\theta}^{(2)})\mathbf{f}_i^{(2)}) \circ \cdots \circ (\mathbf{R}(\boldsymbol{\theta}^{(p)})\mathbf{f}_i^{(p)}), \tag{4.27}$$

which for symmetric tensor is $\mathbf{u}_i^{\circ p} = (\mathbf{R}(\theta)\mathbf{f}_i)^{\circ p}$.

Consequently, we need to evaluate

$$\partial \mathbf{x}_k/\partial \theta_l^{(m)} = vec(\partial(\mathbf{u}_k^{(1)} \circ \mathbf{u}_k^{(2)} \circ \cdots \circ \mathbf{u}_k^{(p)})/\partial \theta_l^{(m)}) \tag{4.28}$$

using

$$\partial \mathbf{R}(\boldsymbol{\theta}^{(m)})/\partial \theta_i^{(m)} = \mathbf{R}(\theta_1^{(m)})\mathbf{R}(\theta_2^{(m)}) \cdots [\partial \mathbf{R}(\theta_i^{(m)})/\partial \theta_i^{(m)}] \cdots \mathbf{R}(\theta_q^{(m)}) \tag{4.29}$$

to obtain

$$\partial(\mathbf{R}(\boldsymbol{\theta}^{(m)})\mathbf{f}_k^{(m)})/\partial \theta_i^{(m)} = [\partial \mathbf{R}(\boldsymbol{\theta}^{(m)})/\partial \theta_i^{(m)}]\mathbf{f}_k^{(m)} \tag{4.30}$$

and then the chain rule to get:

$$\begin{array}{c} \partial(\mathbf{u}_k^{(1)} \circ \mathbf{u}_k^{(2)} \circ \cdots \circ \mathbf{u}_k^{(p)})/\partial \theta_i^{(m)} = \\ \partial(\mathbf{R}(\boldsymbol{\theta}^{(1)})\mathbf{f}_k^{(1)} \circ \cdots \circ \mathbf{R}(\boldsymbol{\theta}^{(m)})\mathbf{f}_k^{(m)} \circ \cdots \circ \mathbf{R}(\boldsymbol{\theta}^{(p)})\mathbf{f}_k^{(p)})/\partial \theta_i^{(m)} = \\ \mathbf{R}(\boldsymbol{\theta}^{(1)})\mathbf{f}_k^{(1)} \circ \cdots \circ (\partial(\mathbf{R}(\boldsymbol{\theta}^{(m)})\mathbf{f}_k^{(m)})/\partial \theta_i^{(m)}) \circ \cdots \circ \mathbf{R}(\boldsymbol{\theta}^{(p)})\mathbf{f}_k^{(p)} \end{array} \tag{4.31}$$

Similarly, using the chain rule appropriately, second partial derivatives can be obtained to construct the Hessian matrix. On the next step we need to evaluate the rotation matrix $\mathbf{R}(\theta^{(m)})$ and its partial derivative with respect to angle $\theta_k^{(m)}$. In Appendix C we describe two widely used approaches as well as their properties for implementation of rotation matrices: Euler-type and skew-symmetric exponential matrices.

4.6.2 Optimization Algorithms

In this part we briefly describe the algorithm that we developed for minimization of the SE in (4.23). Optimization algorithms for the widely known Tucker and CP tensor decomposition models are described in the Chapter 3. The most widely used method for CP and Tucker decompositions is the iterative alternating least squares algorithm. This algorithm allows CP and Tucker models to evaluate their decomposition components (each basis vector) in alternating/random order. We cannot use this algorithm because our proposed frame of basis vectors is geometrically constrained and one needs to iterate all vectors simultaneously as they are not independent from each other.

For symmetric tensors we evaluated the performance of four standard iterative-descent algorithms.

Jacobi rotation: Jacobi rotation updates one Givens angle at each iteration using line search or other optimization technique. On the k^{th} iteration, we update angle θ_i along j^{th} way of s tensor using gradient descent $\theta^{(j)}_{k+1,i} = \theta^{(j)}_{k,i} - \alpha \partial e^2 / \partial \theta^{(j)}_{k,i}$, where the stepsize α satisfies Wolfe's conditions.

Steepest descent: At each iteration, all Givens angles are updated along the direction of negative gradient: $\theta^{(j)}_{k+1} = \theta^{(j)}_{k} - \alpha \nabla e^2(\theta^{(j)}_{k})$.

Gauss-Newton algorithm: Inverse-Hessian provides directional correction in quadratic cost surface regions at the cost of significant increase in computational complexity: $\theta^{(j)}_{k+1} = \theta^{(j)}_{k} - \alpha [\mathbf{H}(\theta^{(j)}_{k})]^{-1} \nabla e^2(\theta^{(j)}_{k})$.

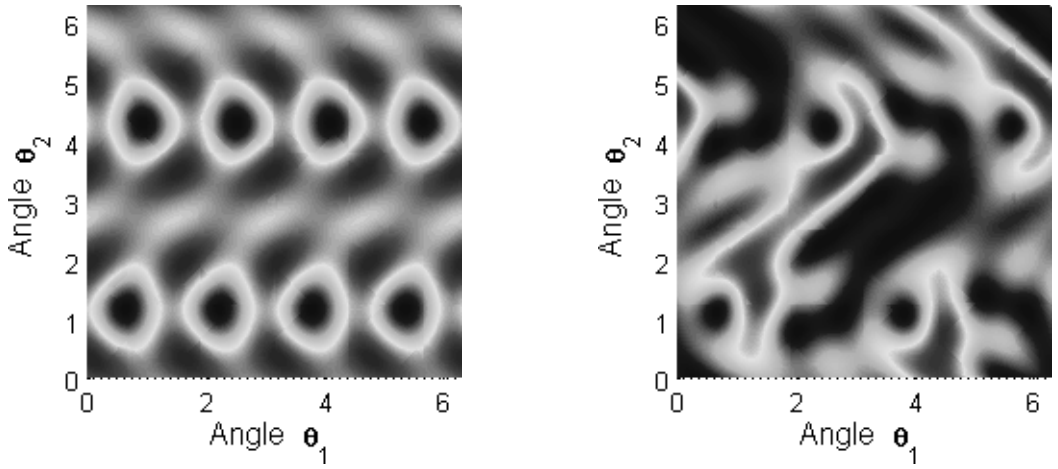
Levenberg-Marquardt algorithm: The Jacobian of the error vector is used to approximate the Hessian to yield convergence rate similar to Gauss-Newton, but at a computational cost comparable to steepest descent, where $[\mathbf{J}(\theta^{(j)}_{k})]^{-1}$ is pseudo-inverse of $\mathbf{J}(\theta^{(j)}_{k})$: $\theta^{(j)}_{k+1} = \theta^{(j)}_{k} - \alpha [\mathbf{J}(\theta^{(j)}_{k})]^{-T} \mathbf{e}(\theta^{(j)}_{k})$.

4.6.3 Numerical Results of Decomposition of Symmetric Tensors

In this section, we evaluate standard iterative descent techniques to determine the proposed tensor decomposition solution by optimizing the Givens angles representation of the rotation matrix of the frame of vectors. Evaluation of optimization algorithms was performed on rank-1 and full-rank tensors with specific structure as well as in a Monte Carlo fashion using tensors generated randomly. Decompositions were approximated to a prespecified accuracy and the numbers of iterations are compared. First, we demonstrate the nonlinear nature of the optimization problem at hand using two simple symmetric tensors: (i) a tensor of ones, which is a rank-1 tensor since it is the p-way outer product of a vector of ones; (ii) a symmetric full-rank n-dimensional order-p tensor $(\mathbf{A})_{l_1 \ldots l_p} = 1 + \sum_{i=1}^{p} (l_i - 1)$.

The visual illustration of the error surface along vector frame in the cross-section of rotation angles θ_1 and θ_2 for the 3-dimensional order-2 (matrix) and order-3 symmetric tensors of integer entries (type-ii) are provided in Figure 4.11. The surface complexity compared to the matrix case is evident. We need to note that an error of decomposition of order-2 any-dimensional or any-order 2-dimensional tensor along any rotation angle θ_i is periodic (due to the nature of trigonometric function). Table 4.1 and Table 4.2 contains the number of iterations that the considered algorithms need to achieve a SE less than 10^{-12} (in Matlab, the minimum SE is on the order of 10^{-30}, but optimization to that level of numerical accuracy takes very long). Table 4.1 shows the results of decomposition of the tensor-of-ones (type-i) and Table 4.2 shows the results of decomposition for the symmetric tensor of integers (type-ii) with all algorithms starting from the same initial estimates. In these tables, "T" indicates that optimization was terminated at iteration 300, the preset maximum number of iterations - and convergence to the desired level had not been achieved. The number of iterations for Jacobi rotations and gradient descent grows exponentially due to the ratio of $\max(\lambda)/\min(\lambda)$.

Next, we present results from a Monte Carlo experiment in which the decompositions were optimized using iterative random search (similar to stochastic annealing) in order to demonstrate that error levels comparable to the minimum possible numerical accuracy are attainable. In Figure 4.12, we show the average normalized SE (per tensor entry) for randomly generated tensors of orders 2 to 5 for dimensions 2 to 5. Errors of decomposition grow exponentially due to the number of elements in high

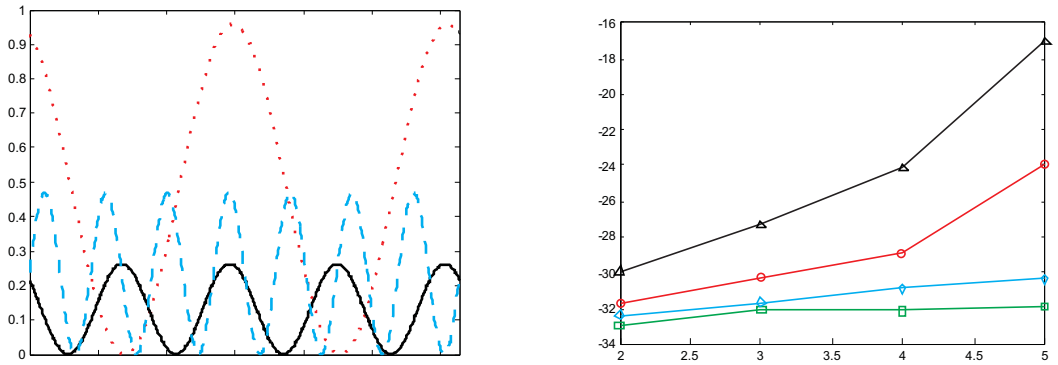


(a) symmetric matrix.

(b) symmetric tensor.

FIGURE 4.11: Error surfaces along angles θ_1 and θ_2 for 3-dimensional (left) order-2 and (right) order-3 tensors.

order high dimensional tensors growing in a combinatorial fashion combined with fast numerical degradation due to ill-conditioning.



(a) Frobenius norm squared of the reconstruction error for tensors of order 2 (dot), 4 (solid), and 7 (dash) versus $[0, \pi)$.

(b) Monte Carlo average log_{10} of normalized SE for the proposed eigendecomposition technique for random tensors of orders 2 to 5 for dimensions 2 (square), 3 (diamond), 4 (circle), 5 (triangle).

FIGURE 4.12: Normalized SE for the proposed tensor decomposition.

Table 4.3 summarizes the results of decompositions for 1000 symmetric full-rank

real-valued tensors with all algorithms starting from random initial estimates for parameters. Experiments reveal that for the given optimality criterion, the Levenberg-Marquardt algorithm is preferable in convergence speed and computational complexity considerations.

TABLE 4.1: Number of iterations to decompose rank-1 tensors.

	Jacobi rotation			Stp. Desc.			Gauss-Newton			Lev.-Marq.		
$n\backslash p$	2	3	4	2	3	4	2	3	4	2	3	4
2	1	1	1	1	1	1	1	1	1	1	1	1
3	8	51	11	9	11	3	11	3	3	7	3	3
4	21	65	63	18	52	28	10	5	5	12	3	4

TABLE 4.2: Number of iterations to decompose full-rank tensors.

	Jacobi rotation			Stp. Desc.			Gauss-Newton			Lev.-Marq.		
$n\backslash p$	2	3	4	2	3	4	2	3	4	2	3	4
2	1	1	1	1	1	1	1	1	1	1	1	1
3	9	T	T	T	T	T	8	15	13	6	7	10
4	T	T	T	T	T	T	24	33	29	11	13	12

TABLE 4.3: Median number of iterations when decomposing 1000 random full-rank tensors.

$n\backslash p$	Jacobi rotation	Stp. Desc.	Gauss-Newton	Lev.-Marq.
3 2	75	53	5	4
3 3	50	71	6	5
3 4	155	118	10	5
4 2	T	295	19	7
4 3	T	T	34	13
4 4	T	T	36	15

Due to the relatively complex optimization criterion (p^{th} order polynomials squared), the Levenberg-Marquardt approach has been identified to be preferable considering computational load and convergence speed. In the next section will evaluate presented non-redundant decomposition to non-symmetric arbitrary non-cube tensors.

4.6.4 Numerical Evaluation of Upper Bound on Tensor Rank

In this section we evaluate the proposed gradient model of the Frobenius norm of the decomposition error to show numerically that the the upper bound of non-symmetric tensor rank satisfies the proposed decomposition model.

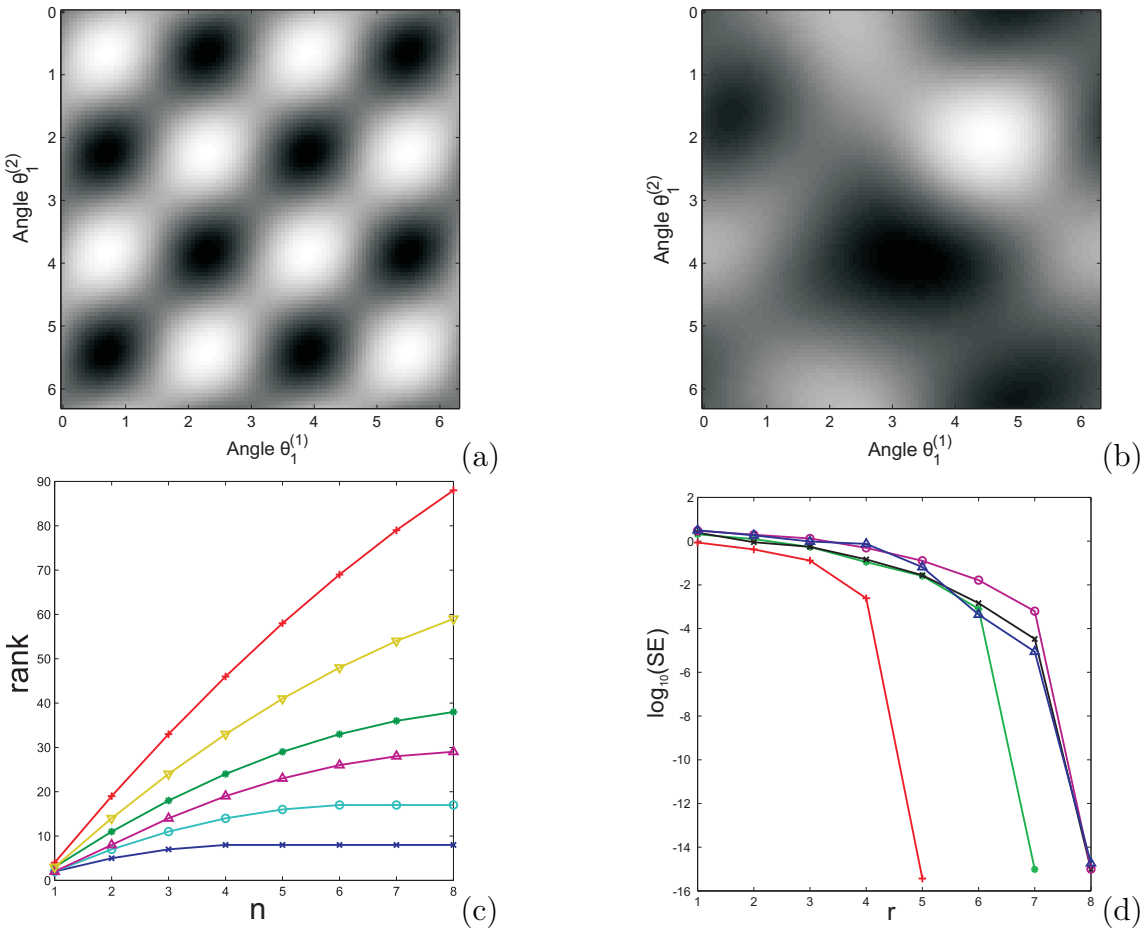


FIGURE 4.13: Error surfaces for (a) 3-dimensional order-2 and (b) order-3 non-symmetric tensors; (c) ranks of tensors: $(2,2,n)$'x', $(2,3,n)$'o', $(2,4,n)$'$\triangle$', $(3,3,n)$'$*$', $(3,4,n)$'$\triangledown$', and $(4,4,n)$'+'; (d) decomposition SE for $(2,2,n)$-dimensional tensor: n=2'+', 3'$*$', 4'$\triangle$', 5'o', 6'x'.

Evaluation of the optimization algorithms was performed on full-rank tensors with entries generated randomly from the standard normal distribution. A visual illustration of the error surface along vector frame in the cross-section of rotation angles $\theta_1^{(1)}$ and $\theta_1^{(2)}$ for the 3-dimensional order-2 (matrix) case in Figure 4.13(a) and order-3 case in Figure 4.13(b) non-symmetric tensors of random entries are provided. The surface complexity compared to the matrix case is evident. We need to note that an error of decomposition of order-2 any dimensional or any order 2-dimensional tensor along any rotation angle $\theta_j^{(i)}$ is periodic (due to the nature of trigonometric

functions). In Figure 4.13(c) we present ranks of order-3 tensors for different dimensions. Figure 4.13(d) shows the normalized SE ($\|T - A\|_F^2 / \|T\|_F^2$) for decomposition of (2,2,d)-dimensional tensor, where $d = 2, \ldots, 6$.

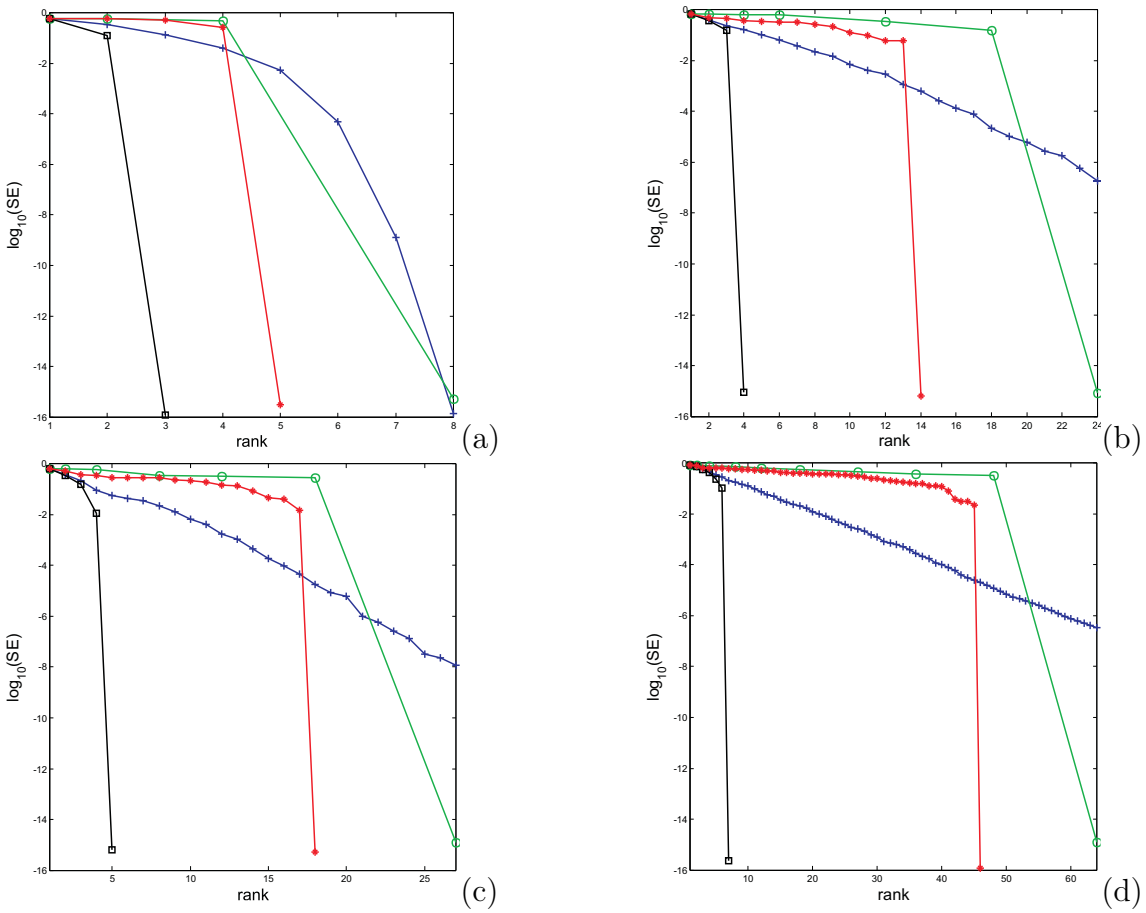


FIGURE 4.14: Non-symmetric tensor decomposition SE and corresponding tensor rank for rank-r CP '□', incremental rank-1 CP '+', Tucker '∘', and proposed '∗' models; a) (2,2,2)-dim. tensor; b) (2,3,4)-dim. tensor; c) (3,3,3)-dim. tensor; d) (4,4,4)-dim. tensor.

Figure 4.14 shows how decomposition results for non-symmetric order-3 different-dimensional tensors depend on rank for best rank-r CP[2] '□', incremental recursive rank-1 CP[3] '+', Tucker[4] '∘', and proposed '∗' models. Clearly, when proposed model uses frames of bases with the correct number of vectors, the decomposition error drops to almost zero (with numerical error remaining). Here on the basis of our definition of the tensor rank, for the data in Figure 4.14 we have that in the case of

[2]All parameters of such a decomposition model are adjusted independently and simultaneously.

[3]By incremental rank-1 CP we mean iterative reduction of error by successive best-rank-1 approximation of residual error after deflation of tensor using previously found rank-1 tensor. This is suboptimal compared to doing best rank-r approximation using CP model. In the case of an order-2 tensor (matrix) both best rank-r and incremental rank-1 CP models proved identical results.

[4]The rank for such the tensor model is evaluated as a product of dimensions of its core tensor.

(a), the rank is equal to 5, (b) - 14, (c) - 18, (d) - 46, and decomposition errors are zero. Proposed approach gives us the upper bound of CP-rank for tensors.

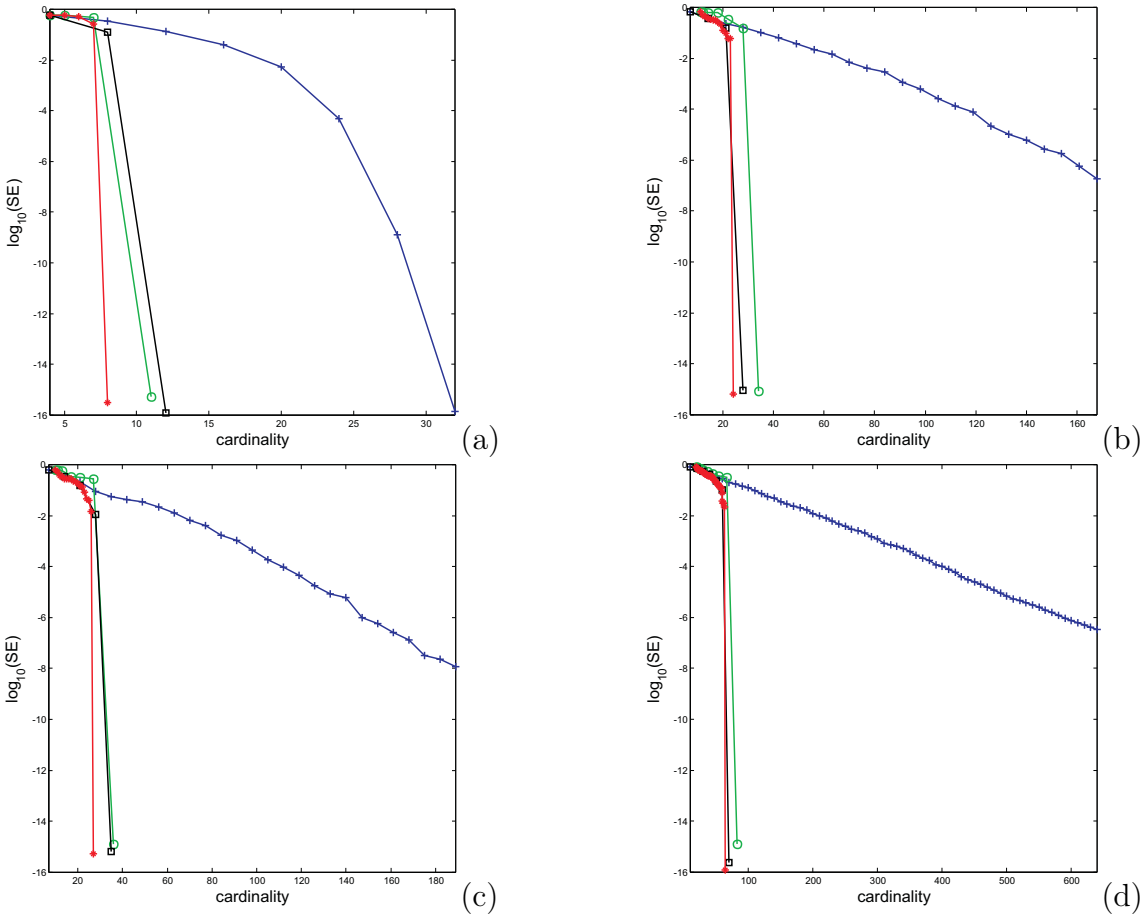


FIGURE 4.15: Non-symmetric tensor decomposition SE and corresponding cardinality for rank-r CP '$\square$', incremental rank-1 CP '+', Tucker '$\circ$', and proposed '$*$' models; a) (2,2,2)-dim. tensor; b) (2,3,4)-dim. tensor; c) (3,3,3)-dim. tensor; d) (4,4,4)-dim. tensor.

Figure 4.15 shows how decomposition results for the same non-symmetric order-3 different-dimensional tensors from Figure 4.14 depend on cardinality for best rank-r CP '$\square$', incremental recursive rank-1 CP '+', Tucker '$\circ$', and proposed '$*$' models. Evaluation of the tensor cardinality was performed in the estimated ranks of the tensors. Clearly, the proposed non-redundant tensor model utilizes the least numbers of parameters to reparameterize tensors. Here on the basis of our definition of the tensor rank and rotated vector frames, we have that in the case of (a), the cardinal number is equal to 8, (b) - 24, (c) - 27, (d) - 64, i.e., the product of tensor's dimensions. Proposed approach gives us the lower bound of cardinality. The rank-r CP tensor model decomposes data sets with lower tensor rank but utilizes more parameters

than cardinality of tensors so it is redundant decomposition. The incremental rank-1 CP and Tucker models are redundant as well.

In the Algorithms 4 we present the pseudocode for non-redundant tensor decomposition. Matlab functions can be downloaded from website [67].

Algorithm 4 Non-redundant decomposition of arbitrary tensor.

In: $\mathbf{T}$ of size $n_1 \times n_2 \times \cdots \times n_p$.
Out: $\mathbf{A}$ of size $n_1 \times n_2 \times \cdots \times n_p$, an estimate of the rank-r decomposition of $\mathbf{T}$.

Based on values of p and $n_i, i = 1, \ldots, p$, initiate variables:
a) determine the rank of $\mathbf{T}$, r.
b) construct vector frames $\mathbf{F}^{(i)}, i = 1, \ldots, p$, for each side of tensor $\mathbf{T}$.
c) choose a combination of r vectors from $\mathbf{F}^{(i)}, i = 1, \ldots, p$. Combinations must satisfy the permutation tensor $\mathbf{Q}$.
d) build the matrix of vectorized rank-1 tensors, $\mathbf{X}$, from $\mathbf{F}^{(i)}, i = 1, \ldots, p$, based on $\mathbf{P}$.
e) calculate the matrix of inner product of rank-1 tensors $\mathbf{B} = \mathbf{X}^T\mathbf{X}$.
f) determine the number of rotation angles in $\theta^{(i)}$ for each $\mathbf{F}^{(i)}$,$i = 1, \ldots, p$ and assign them random values.
while SE has not converged to zero **do**
  Evaluate rotated vector frames $\mathbf{U}^{(i)} = \mathbf{R}(\theta^{(i)})\mathbf{F}^{(i)}, i = 1, \ldots, p$.
  Build the matrix of vectorized rank-1 tensors, $\mathbf{X}$, from $\mathbf{U}^{(i)}, i = 1, \ldots, p$, based on $\mathbf{P}$.
  Evaluate the gradient and Jacobian of SE, $e^2 = \mathbf{t}^T(\mathbf{I} - \mathbf{X}\mathbf{B}^{-1}\mathbf{X}^T)\mathbf{t}$.
  Adjust rotation angles $\theta^{(i)},\ i = 1, \ldots, p$ to minimize the SE.
end while
Set $\mathbf{U}^{(i)} = \mathbf{R}(\theta^{(i)})\mathbf{F}^{(i)}, i = 1, \ldots, p$, build matrix $\mathbf{X}$.
Set $\lambda = \mathbf{B}^{-1}\mathbf{X}^T\mathbf{t}$ and $vec(\mathbf{A}) = \mathbf{X}\lambda$.

4.6.5 Squared Frobenius Norm of Vector Frame

We can describe a vector frame $\mathbf{F}$, where $\mathbf{F}$ is (n, r)-dimensional matrix, as points on the hypersphere of unit radius. Thus all diagonal entries in $\mathbf{B} = (\mathbf{F}^T\mathbf{F})^{\cdot p} = \mathbf{Y}^T\mathbf{Y}$, where i^{th} column of (n^p, r)-dimensional matrix $\mathbf{Y}$ is obtained as $\mathbf{Y}_{:i} = vec(\mathbf{f}_i^{(1)} \circ \mathbf{f}_i^{(2)} \circ \cdots \circ \mathbf{f}_i^{(p)})$, equal 1 and all non-diagonal entries evaluate distances between points on hypersphere. The operator $\mathbf{A}^{\cdot p}$ means element-by-element powers of $\mathbf{A}$. In general case, for maximization of distances between vectors for an order-p n-dimensional tensor we need to minimize absolute values of non-diagonal elements in $\mathbf{B}$ that can be written as a solution of an optimization problem: $e^2 = \langle \mathbf{B} - \mathbf{I}, \mathbf{B} - \mathbf{I} \rangle$. The

vectorized error $\mathbf{e} = vec(\mathbf{B} - \mathbf{I}) = \mathbf{i} - \mathbf{b}$ and after some algebra SE is:

$$e^2 = n - 2\mathbf{i}^T\mathbf{b} + \mathbf{b}^T\mathbf{b}. \tag{4.32}$$

Without any constrains, except unit vector length, on the structure of vector frame $\mathbf{F}$ in hyperspherical coordinate system we can precisely define the positions of vectors by their hyperspherical angles. Thus SE depends on $(n-1, r)$-dimensional parameter matrix $\mathbf{\Psi}$, where each column ψ_i defines position of i^{th} vector in n-dimensional space. As described above, to solve this problem using the gradient and Jacobian matrix we can employ the Levenberg-Marquardt algorithm. The gradient of SE with respect to the hyperspherical angles $\psi_{i,j}$ is:

$$\nabla e^2(\psi_{i,j}) = 2(\mathbf{b}^T - \mathbf{i}^T)\partial\mathbf{b}/\partial\psi_{i,j}, \tag{4.33}$$

where $\partial\mathbf{B}/\partial\psi_{i,j} = (\partial\mathbf{Y}^T/\partial\psi_{i,j})\mathbf{Y} + \mathbf{Y}^T(\partial\mathbf{Y}/\partial\psi_{i,j})$ and $(\partial\mathbf{Y}/\partial\psi_{i,j})$ can be obtained by chain rule as in previous subsection. The k^{th} column of the Jacobian matrix $\mathbf{J}$ for $\psi_{i,j}$, with one-to-one reparametrization between k and pair of (i, j), is:

$$\mathbf{j}_k(\psi_{i,j}) = \partial\mathbf{b}/\partial\psi_{i,j}. \tag{4.34}$$

Chapter 5

Applications of Tensors

In this chapter we describe utilization of proposed tensor decomposition model for the density approximation problem, which is involved into the tasks of clustering and principal curve estimation, generalized and robust nonlinear PCA, solving of nonlinear equations. The most of the multidimensional nonlinear problems can be defined by the symmetric tensor. Knowledge of decomposition model of such a tensor allows us to analyze, compress, factorize, predict examined data.

5.1 Density Estimation

This section has been prepared based on the publications of Prof. Rafail Abramov from The University of Illinois at Chicago. Particularly, we refer the reader to his publication [68] to learn more comprehensive explanations on multidimensional maximum entropy problem.

In this section we describe one of the state-of-the-art moment-constrained approaches for the task of multidimensional density estimation and present our solution for such a problem. Existing methods mostly solve the maximum entropy moment problem. Proposed model is based on maximization of the loglikelihod of density which is defined by the higher-order tensor. We involve the proposed decomposition model of such a tensor in optimization procedure. As described in the Chapter 1 any moments for any-dimensional system can be defined as corresponding symmetric tensor. Many existing algorithms developed for the maximum entropy problem are multidimensional moment-constrained approaches and, due to complexity, practically capable to solve such a problem for the $\{2, 3, 4\}$-dimensional data space with

correspondent moment constraints up to order-$\{8, 6, 4\}$, respectively. Similar restrictions are intrinsic for our proposal as well.

The moment-constrained maximum entropy problem yields an estimation of a probability density with highest uncertainty among all densities satisfying supplied moment constraints. The moment constrained maximum entropy problem arises in physics [69], econometrics [70], geophysics [71], etc. The approximation of the probability density itself is obtained by maximizing the Shannon entropy under the constraints established by measured moments [72]. A standard formalism transforms the constrained maximum entropy problem into the unconstrained minimization problem of the dual objective function. To read more details on theoretical aspects we refer to the latest review [68].

5.1.1 Maximum Entropy Problems with Constraints

Before reviewing the moment-constrained entropy problem, we need to define the Shannon entropy for such a problem. Let's have some knowledge of a probability density $g(\mathbf{x})$, where $\mathbf{x}$ denotes a vector data space, in form of linear constraints [68]:

$$\begin{aligned} f_0 = F_0(g) = \int g(\mathbf{x})d\mathbf{x} = 1, \\ f_i = F_i(g) = \int C_i(\mathbf{x})g(\mathbf{x})d\mathbf{x} = 1,\ 1 \leq i \leq L, \end{aligned} \tag{5.1}$$

where $C_i(\mathbf{x})$ is such a function integral of which (5.1) is finite and linearly independent. Then we can formulate two maximum entropy problems [68] which satisfy (5.1) as:

1. Find the optimal probability density g_S^* which maximizes the Shannon entropy

$$S(g) = -\int g(\mathbf{x}) \ln g(\mathbf{x})d\mathbf{x} \tag{5.2}$$

   (i.e., such that $S(g_S^*) = \max_g S(g)$));

2. Find the optimal probability density g_P^* which minimizes the relative entropy

$$P(g, \Pi) = \int g(\mathbf{x}) \ln[\frac{g(\mathbf{x})}{\Pi(\mathbf{x})}]d\mathbf{x} \tag{5.3}$$

   (i.e., such that $P(g_P^*, \Pi) = \min_g \Pi(g, \Pi)$), where $\Pi(\mathbf{x})$ is some known probability density.

A standard optimization approach allows to write these problems for the Lagrange multipliers in the unconstrained form [72]. In particular, the constrained optimization problems in (5.2) and (5.3) are reduced to the unconstrained minimization [68]:

$$\begin{aligned} L_S(\lambda) &= \int exp(\textstyle\sum_{i=0}^{L} \lambda_i C_i(\mathbf{x}))d\mathbf{x} - \sum_{i=0}^{L} \lambda_i f_i, \\ L_P(\theta) &= \int \Pi(\mathbf{x}) exp(\textstyle\sum_{i=0}^{L} \theta_i C_i(\mathbf{x}))d\mathbf{x} - \sum_{i=0}^{L} \theta_i f_i, \end{aligned} \tag{5.4}$$

over their sets of Lagrange multipliers λ and θ, respectively. The Hessians matrices of (5.4) are positive definite so we have convex optimization problems with unique optima [68]. Then the optimal probability densities g_S^* and g_P^* are

$$g_S^*(\mathbf{x}) = exp(\sum_{i=0}^{L} \lambda_i C_i(\mathbf{x})), \tag{5.5}$$

$$g_P^*(\mathbf{x}) = \Pi(\mathbf{x}) exp(\sum_{i=0}^{L} \theta_i C_i(\mathbf{x})). \tag{5.6}$$

If the gradients in (5.4) are equal to zeros then the constraints in (5.1) are automatically satisfied by (5.5). We need to note that, since $C_0(x) = 1$ in (5.5), λ_0 and θ_0 can be found explicitly from the other Lagrange multipliers [68] as:

$$\lambda_0 = -\ln \int exp(\sum_{i=1}^{L} \lambda_i C_i(\mathbf{x}))d\mathbf{x} \text{ end } \theta_0 = -\ln \int \Pi(\mathbf{x}) exp(\sum_{i=1}^{L} \theta_i C_i(\mathbf{x}))d\mathbf{x}. \tag{5.7}$$

However, in the following explanation we will treat λ_0 and θ_0 as generic Lagrange multipliers [68].

5.1.2 Moments and Data Preprocessing

Here we introduce a concise written form of an arbitrary moment [68]. Let's have $\mathbf{x} \in \mathbb{R}^n$, any monomial (the product of some powers of $\mathbf{x}^{th}$ components) of $\mathbf{x}$ can be written as:

$$\mathbf{x}^{\mathbf{i}} = \prod_{k=1}^{n} x_k^{i_k}, \ \mathbf{i} \in \mathbb{I}^n, \tag{5.8}$$

where the monomial order $|\mathbf{i}|$ is the total power of all vector components, i.e., $|\mathbf{i}| = \sum_{k=1}^{n} i_k$. Using the above notation, for a probability density g we write a set of

arbitrary moment constraints up to the total power m as [68]

$$m_{\mathbf{i}} = \mu_{\mathbf{i}}(g) = \int \mathbf{x}^{\mathbf{i}} g(\mathbf{x}) d\mathbf{x}, \ |\mathbf{i}| = 0, \ldots, m. \tag{5.9}$$

Before proceeding with the numerical algorithm for optimization, it is desirable to simplify the initial problem as much as possible via linear transformations of coordinate system $\mathbb{R}^n$. This preprocessing removes many implementation problems and improves convergence of optimization algorithms. To see more about transformation procedure between the Lagrange multipliers and moments we refer the reader to [68].

Initial step in many preprocessing algorithm is shifting the mean of the data so the target probability density g^* is shifted to zero [68]. Let $\mathbf{m}$ be the mean state of g^*, and $\tilde{g}^*$ be the shifted probability density:

$$\tilde{g}^*(\mathbf{x}) = g^*(\mathbf{x} + \mathbf{m}) \tag{5.10}$$

so that the mean of the $\mathbf{x}$ in $\tilde{g}^*$ is equal to zero.

The next step of preprocessing is rotating procedure and stretching the coordinate system so that the covariance matrix of $\mathbf{x}$ becomes the identity one [68]. We perform this as:

$$\mathbf{x}_{old} = E\Lambda^{1/2}\mathbf{x}_{new}, \tag{5.11}$$

where E is the matrix of eigenvectors, and Λ is the diagonal matrix of eigenvalues for the constraint covariance matrix.

But, performing defined operations is not enough since the higher power moment constraints differ from the low power moment constraints by several orders of magnitude, which influences negatively on convergence of the algorithm [68]. For instance, the moments of a simple 1-dimensional Normal distribution, $N(0, 1)$ grow extremely fast with its order increasing, p, and can be simply defined as $\mu_p(g_N) = (p-1)!!$, where p is even [68]. To partially improve this problem we can transform initial data set with some scale factor α so that

$$g^*_\alpha(\mathbf{x}) = \alpha^n \hat{g}^*(\alpha\mathbf{x}) \text{ and } \mu_{\mathbf{i}}(g^*_\alpha) = \alpha^{-|\alpha|}\mu_{\mathbf{i}}(\hat{g}^*), \tag{5.12}$$

where α can be chosen to minimize the difference in magnitude between different moments and usually expressed as [68] $\alpha = \sqrt[2p]{(2p-1)!!}$.

5.1.3 Maximum Entropy Problems Based on Moment Constraints

For practical applications it is common for the constraints in (5.1) to be various products of powers of data space coordinates (moments) [68]. With the set of moment constraints in (5.9), the unconstrained optimization problems in (5.4) become

$$L_S(\lambda) = \int exp(\sum_{|\mathbf{i}|=0}^{p} \lambda_{\mathbf{i}} \mathbf{x}^{\mathbf{i}}) d\mathbf{x} - \sum_{|\mathbf{i}|=0}^{p} \lambda_{\mathbf{i}} m_{\mathbf{i}}, \tag{5.13}$$

$$L_P(\theta) = \int \Pi(\mathbf{x}) exp(\sum_{|\mathbf{i}|=0}^{p} \theta_{\mathbf{i}} \mathbf{x}^{\mathbf{i}}) d\mathbf{x} - \sum_{|\mathbf{i}|=0}^{p} \theta_{\mathbf{i}} m_{\mathbf{i}}, \tag{5.14}$$

with corresponding optimal probability densities

$$g_S^*(\mathbf{x}) = exp(\sum_{i=0}^{M} \lambda_i \mathbf{x}^{\mathbf{i}}), \tag{5.15}$$

$$g_P^*(\mathbf{x}) = \Pi(\mathbf{x}) exp(\sum_{i=0}^{M} \theta_i \mathbf{x}^{\mathbf{i}}). \tag{5.16}$$

These probability densities differ from each other due to the factor $\Pi(\mathbf{x})$. In particular, if $\Pi(\mathbf{x})$ itself is of the form (5.15) then the optimization problem in (5.14) can be reduced to (5.13) [68]. Let Π be

$$\Pi(\mathbf{x}) = exp(\sum_{|\mathbf{i}|=0}^{p} \beta_{\mathbf{i}} \mathbf{x}^{\mathbf{i}}). \tag{5.17}$$

This choice of Π is natural if Π is the optimal probability density in (5.13) with different constraints [68]. Then the optimization problem in (5.14) can be written as

$$L_P(\theta) = \int exp(\sum_{|\mathbf{i}|=0}^{p} (\theta_{\mathbf{i}} + \beta_{\mathbf{i}}) \mathbf{x}^{\mathbf{i}}) d\mathbf{x} - \sum_{|\mathbf{i}|=0}^{p} \theta_{\mathbf{i}} m_{\mathbf{i}}. \tag{5.18}$$

Changing variables $\xi_{\mathbf{i}} = \theta_{\mathbf{i}} + \beta_{\mathbf{i}}$ we obtain for (5.14)

$$L_P(\xi) = \int exp(\sum_{|\mathbf{i}|=0}^{p} \xi_{\mathbf{i}} \mathbf{x}^{\mathbf{i}}) d\mathbf{x} - \sum_{|\mathbf{i}|=0}^{p} \xi_{\mathbf{i}} m_{\mathbf{i}} + \sum_{|\mathbf{i}|=0}^{p} \beta_{\mathbf{i}} m_{\mathbf{i}}, \tag{5.19}$$

where the last sum is constant and does not influent on (5.19) [68]. We see that $L_P(\xi)$ and $L_S(\lambda)$ for (5.13) differ just by a constant. Once the optimal sets of Lagrange multipliers for $\rho_S(\mathbf{x})$ and $\rho_P(\mathbf{x})$ are found, the corresponding Shannon and relative entropies can be computed as [68]

$$S(\rho_S^*) = -\sum_{|\mathbf{i}|=0}^{M} \lambda_{\mathbf{i}} m_{\mathbf{i}} \text{ end } P(\rho_P^*, \Pi) = \sum_{|\mathbf{i}|=0}^{M} (\xi_{\mathbf{i}} - \beta_{\mathbf{i}}) m_{\mathbf{i}}. \tag{5.20}$$

5.1.3.1 Polynomial Basis and its Reorthonormalization

For the optimization problem (5.13), which is convex, we can apply some an iterative technique, e.g. the Newton method [73], but due to different sensitivity of $L(\lambda)$ to changes in different multipliers $\lambda_{\mathbf{i}}$ we meet with numerical difficulties. It appears because the exponential function in (5.13) is more sensitive to changes in higher powers of $\mathbf{x}$, so an optimization process over the Lagrange multipliers can fail even when the order p is equal to 6 or 4 [68].

Thus, to resolve this problem in (5.13), we can replace the set of monomials $\mathbf{x}^{\mathbf{i}}$ with a different basis, in which (5.13) has similar sensitivity [68]. A simple solution is to replace basis monomials $\mathbf{x}^{\mathbf{i}}$ with a set of order-p polynomials $f_j(\mathbf{x})$,

$$\{\mathbf{x}^{\mathbf{i}}, \lambda_{\mathbf{i}}\},\ \mathbf{i} \in \mathbb{I}^n, 0 \leq |i| \leq p \rightarrow \{f_j(\mathbf{x}), \gamma_j\},\ 1 \leq j \leq k,\ k = \frac{(p+n)!}{p!n!} \tag{5.21}$$

where γ_j are the Lagrange multipliers in new basis [68]. Since each basis polynomial $\mathbf{f}_j(\mathbf{x})$ is the order-p, the cost function should have similar sensitivity to changes in different γ_j [68]. So new objective function in polynomial coordinates is

$$L(\gamma) = \int exp(\sum_{j=1}^{k} \gamma_j f_j(\mathbf{x}))d\mathbf{x} - \sum_{j=1}^{k} \gamma_j f_j(\mu), \tag{5.22}$$

where $f_j(\mu)$ denotes the polynomial $f_j(\mathbf{x})$, where all the powers of $\mathbf{x}^{\mathbf{i}}$ are replaced with corresponding constraints $\mu_{\mathbf{i}}$ from (5.9) [68]. The corresponding optimal probability density function now is

$$g^*(\mathbf{x}, \gamma) = exp(\sum_{j=1}^{k} \gamma_j f_j(\mathbf{x})). \tag{5.23}$$

Below we show how to adapt described polynomial system to the Newton method [68]. The Newton iteration is:

$$\gamma^{n+1} = \gamma^n - \zeta_n (H^{-1} \nabla L)|_{\gamma^n}, \tag{5.24}$$

where γ^n and ζ_n denote the vectors of Lagrange multipliers and a stepping distance parameter at the n^{th} Newton iteration, respectively. The entries of the gradient, $\nabla\Lambda$, and Hessian, $\mathbf{H}$, are

$$(\nabla L)_k = \frac{\partial L}{\partial \gamma_k} = Q_\gamma(f_k) - f_k(\mu), \tag{5.25}$$

$$(H)_{kl} = \frac{\partial^2 \Lambda}{\partial \gamma_k \partial \gamma_l} = Q_\gamma(f_k f_l), \tag{5.26}$$

where the $Q_\gamma(g)$ for some function $g(\mathbf{x})$ is computed as

$$Q_\gamma(g) = \int s(\mathbf{x}) g^*(\mathbf{x}, \gamma) d\mathbf{x}. \tag{5.27}$$

The optimization problem in new coordinates remains convex [68]:

$$\mathbf{v}^T H \mathbf{v} = v_k Q(f_k f_l) v_l = Q(v_k f_k f_l v_l) = Q((v_k f_k)^2) \geq 0. \tag{5.28}$$

To improve numerical stability in (5.25), we can require the orthogonal properties for the f_k for each step of optimization algorithm [68]:

$$Q(p_k p_l) = \delta_{pl}, \tag{5.29}$$

where δ_{pl} is the usual Kronecker delta-symbol, so

$$(H)_{kl} = (H^{-1})_{kl} = \delta_{kl}. \tag{5.30}$$

With (5.30), the Newton method in (5.25) is the same as the steepest descent method:

$$\gamma^{n=1} = \gamma^n - \zeta_n (\nabla)|_{\gamma^n}. \tag{5.31}$$

The quadratures Q in (5.27) are weighted by the p.d.f. in (5.23) meanings of which change between different points of the optimization process, and thus the orthogonal

relations in (5.29) is not hold permanently. So we need to apply some orthonormalization procedure, e.g. the Gram-Schmidt one, to keep the f_k orthogonal in the sense of (5.29) [68].

According to the classical Gram-Schmidt method, k arbitrary linearly independent polynomials a_j are converted into the orthonormal (in the sense of (5.29)) set of polynomials f_j as [68]

$$f_j = \frac{a_j - \sum_{l=1}^{j-1} Q(a_j f_l) f_l}{Q([a_j - \sum_{l=1}^{j-1} Q(a_j f_l) f_l]^2)^{(1/2)}}, \; 1 \leq j \leq k. \tag{5.32}$$

Poor numerical stability of the classical Gram-Schmidt reorthonormalization is known and common numerically stable tools to reorthonormalize a set of Euclidean vectors usually involve the Householder reflections or Givens rotations [48]. The recent work [74] demonstrates the modified Gram-Schmidt algorithm that yields errors which are small multiples of the machine round-off error [68].

Algorithm 5 Modified Gram-Schmidt algorithm.

for $j = 1, \ldots, k$ **do**
  for l=1,. . . ,r **do**
    for m=1,. . . ,j-1 **do**
      $a_j = a_j - Q(a_j f_j) f_j$.
    end for
    $f_j = Q(a_j^2)^{-1/2} a_j$.
  end for
end for

5.1.3.2 Optimization Algorithm

The described optimization algorithm, which involves polynomial basis, performs the Gram-Schmidt reorthonormalization at each step in (5.31) [68]. To reduce the computational expense we can perform several descent iterations between the Gram-Schmidt reorthonormalizations. In this case we can employ some quasi-Newton methods which is not vulnerable to curvature anisotropy, e.g. the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [73]. It needs the initial state of the Hessian, H_0, which is an identity matrix after polynomial reorthonormalization, and requires computation of the gradient of (5.22) [68]. The structure of the BFGS approach [68] for some i^{th} itaration is:

1. Find the direction of descent as $\mathbf{H}_i \mathbf{d}_i = -(\nabla L)_i$.

2. Perform a line search and find the next iterate: $\gamma_{i+1} = \gamma_i + \zeta_i \mathbf{d}_i$.

3. At the new iterate γ_{i+1}, compute the gradient $(\nabla L)_{i+1}$.

4. Apply the Sherman-Morrison formula [73] to get pseudo-Hessian, $\mathbf{H}_i^{-1}$, and compute the descent direction, $\mathbf{d}_i = -\mathbf{H}_i^{-1}(\nabla\lambda)_i$.

The schematic description of the moment-constrained approach [68] is presented in the Algorithm 6. On practice, the values of n and p are limited for the algorithm due to its computational speed.

Algorithm 6 Moment-constrained density estimation.

Perform precondition constraints: zero mean and diagonal covariance matrix of $\mathbf{x}$.
Create k p-order random linearly independent polynomials f_j, $j = 1, \ldots, k$.
Initiate a starting values of the Lagrange multipliers γ_j.
repeat
  Reorthogonalize the polynomials f_j according to (5.29) with respect to g and reevaluate the set of γ_j.
  Compute the gradient L.
  Perform BFGS steps to get the minimum.
until minimum is not reached
Recompute the optimal γ_j into the set of standard Lagrange multipliers $\lambda_{\mathbf{i}}$ for the monomial basis $\mathbf{x}^{\mathbf{i}}$.

5.1.4 Maximum Entropy Problems Based on Symmetric Tensors

A generalized exponential function for multivariate family in n-dimensional space can be defined as

$$g(\mathbf{x}) = e^{-(q(\mathbf{x}))^2}, \tag{5.33}$$

where vector $\mathbf{x}$ consists of $n+1$ elements: $\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$. The polynomial $q(\mathbf{x})$, in terms of tensor operations, can be presented as

$$q(\mathbf{x}) = \langle \mathbf{A}, \mathbf{x}^{\circ p} \rangle = \sum_{l_1=0}^{n} \ldots \sum_{l_p=0}^{n} \mathbf{A}_{l_{1\ldots p}} \cdot x_{l_1} \cdot \ldots \cdot x_{l_p}, \tag{5.34}$$

where $x_0 = 1$. If the symmetric tensor $\mathbf{A}$ containing these polynomial coefficients is decomposed into the desired form, then the polynomial can be written as

$$\langle \mathbf{A}, \mathbf{x}^{\circ p} \rangle = \sum_{l=1}^{r} \lambda_l (\mathbf{u}_l^T \mathbf{x})^p, \tag{5.35}$$

and a generalized exponential density in factorized form is:

$$g(\mathbf{x}) = e^{-(\sum_{l=1}^{r} \lambda_l (\mathbf{u}_l^T \mathbf{x})^p)^2}. \tag{5.36}$$

As in the case of moment-constrained approach we propose to estimate the distribution of data points by minimization of negative loglikelihood of expression:

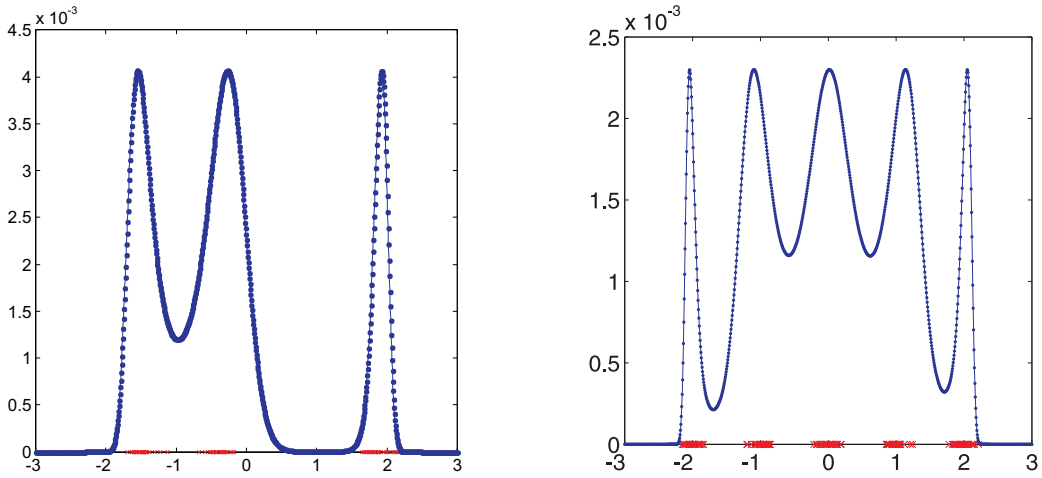
$$LL = -log \prod_{i=1}^{k} \frac{e^{-(q(\mathbf{d}_i))^2}}{\sum_{j=1}^{m} e^{-(q(\mathbf{t}_j))^2}}, \tag{5.37}$$

where data points $\mathbf{d}_i$ are initial given data set and $\mathbf{t}_j$ is data set for density estimation.

On Figure 5.1 we demonstrate the computational ability of the maximum entropy algorithm solving the problem of the density estimation for 1 and 2-dimensional data sets. Figure 5.1(a) and Figure 5.1(b) present 1-dimensional toy problems for multimodal distributions, where red dots define $\mathbf{d}_i$ and $\mathbf{y}_j$ are linearly equally spaced points between -3 and 3 in which we estimate the distribution of $\mathbf{d}_i$.

Figure 5.2 presents a result of density estimation for the "Faith geyser eruption" [75], which is at least bimodal, where axes denote interval and duration of eruption, respectively.

Algorithms 7 presents pseudocode for density estimation based on non-redundant symmetric tensor decomposition. The future work on this approach will be intended to the ability of the developed algorithm to converge for the problems with higher dimensions and orders, due to the number of iterations as well as the time to performance for a single iteration needed to converge for a higher-order maximum entropy problem increases significantly.



(a) (b)

FIGURE 5.1: Density estimation by the model of symmetric tensor decomposition: (a) univariate 3-modal asymmetric density (3-order tensor); (b) univariate 5-modal density (5-order tensor).

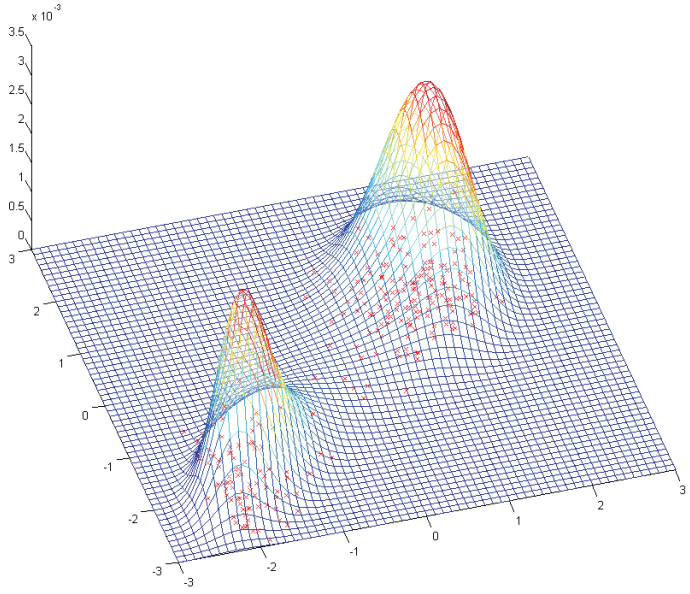


FIGURE 5.2: Density estimation of the Faith geyser eruption (interval and duration) by the 4-order model.

5.2 Principal Curve Fitting and Clustering

This section has been prepared based on the publications of Dr. Umut Ozertem from the Yahoo! Search Sciences (USA). Particularly, we refer the reader to his

Algorithm 7 Density estimation based on tensor decomposition.

In: $\mathbf{D}$ is $(n+1,k)$-dimensional matrix of n-dimensional data, where each column $\mathbf{d}_i = [1, d_1, \ldots, d_n]$.
Out: Density estimation via generalized exponential function $g(\mathbf{x}) = e^{-(q(\mathbf{x}))^2}$, where $\mathbf{x} = [1, x_1, \ldots, x_n]$.
Based on prior information initiate variables:
a) assign the order, p, for the polynomial $q(\mathbf{x}) = \sum_{l=1}^{r} \lambda_l (\mathbf{u}_l^T \mathbf{x})^p$.
b) determine the rank, r, and the number of rotation angles, θ, for vector frame $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$.
c) generate a set of data points $\mathbf{T}$ such that $\mathbf{D} \in \mathbf{T}$.
d) assign random values for $\lambda_l, \ldots, \lambda_r$ and θ.
while LL is not converged to minimum **do**
  Evaluate $p_{\mathbf{d}_i} = -(q(\mathbf{d}_i))^2, i = 1, \ldots, k$.
  Evaluate $p_{\mathbf{t}_i} = exp(-(q(\mathbf{t}_i))^2), i = 1, \ldots, m$.
  Normalize $p_{\mathbf{t}_i} = \frac{p_{\mathbf{t}_i}}{s_{\mathbf{t}}}$, where $s_{\mathbf{t}} = \sum_{i=1}^{m} p_{t_i}$.
  Loglikelihood $LL = -\sum_{i=1}^{k} p_{\mathbf{d}_i} + k \log s_{\mathbf{t}}$.
  Evaluate the gradient and Jacobian of LL.
  Adjust rotation angles θ and λ to minimize the LL.
end while
Set $g(\mathbf{x}) = e^{-(q(\mathbf{x}))^2}$, where $q(\mathbf{x}) = \sum_{l=1}^{r} \lambda_l (\mathbf{u}_l^T \mathbf{x})^p$.

publication [76] to learn more comprehensive explanations on principal curve fitting and clustering.

Data mining is a process of discovering patterns in data and relations among them. It covers aspects of classification, segmentation, clustering, smoothing, representation of discovered results, ets. The representation may be done by the way of statistical indicators or through visualization by images, trees or graphs [77]. Discovered data is any information that can be numerically interpreted, measured, and compared. For instance, we present visual scenes of nature in form of images for the purpose of collecting, analyzing, transmitting, and processing. Segmentation, clustering, and smoothing do not have distinguished boundaries and can use similar or equal approaches. In common case, segmentation is a process of data partitioning onto closed regions with taking into account similarity characteristics of the data parts (a size of parts can be from one datum to whole data set) [78]; clustering can be described as a task of segmentation without boundary on cohesiveness [79]; smoothing is a process which removes short-term variations, or "noise" to discover the important underlying form of the data [76]. In this section we apply the proposed model of symmetric tensor decomposition to the tasks of clustering and principal curve fitting. Experimental results are presented for each approach.

5.2.1 Principal Curve Fitting

One of the first definition of principal curve is "self-consistent smooth curve which passes through the "middle" of a n-dimensional probability distribution or data cloud" [80]. The recently proposed another definition for principal curves is "a point in the data feature space is on the principal curve if and only if the gradient of the p.d.f. is parallel to one of the eigenvectors of the Hessian and the remaining eigenvectors have negative eigenvalues" [81], which describes the principal curve in terms of the gradient and the Hessian of the data probability density and yields constrained maximum likelihood type algorithms. We utilize this definition for our model of tensor decomposition.

Let's consider the simple illustration in Figure 5.3. To find the principal curve, we start with the density estimation of the data (red dots) and estimate probability density (mesh), e.g., by the means of kernel density estimation (KDE). The principal curve is the ridge of the p.d.f. (green dort). This is exactly where the gradient of the p.d.f. is parallel to one of the eigenvectors of the Hessian of the p.d.f.

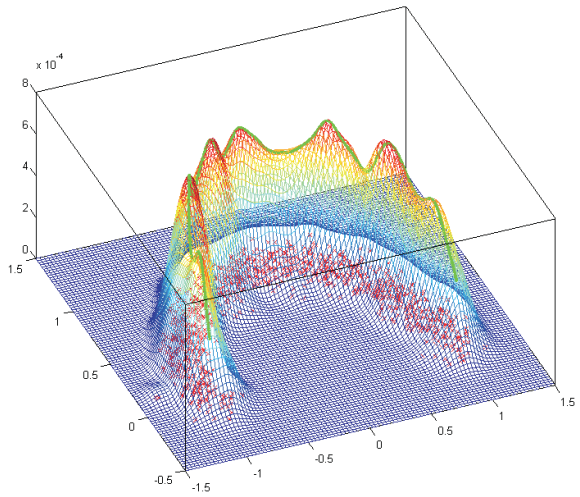


FIGURE 5.3: Curve dataset (red dots), KDE of the curve dataset (mesh) and its principal curve (green dots).

Principal curve definition in the terms of the data p.d.f. allows us to leave all regularization constraints to the density estimation step, which were described in the previous Chapter.

To utilize principal curve projections, one should start with translating the signal into a suitable feature space. Assuming that the observed signal $\mathbf{x}_i$ corrupted by

additive noise, where vector $\mathbf{x} = [1, x_1, \ldots, x_n]$ combines the n-dimensional samples of the noisy signal $[x_1, \ldots, x_n]$.

For unimodal additive noise distributions, e.g., Gaussian noise, the noiseless signal will most likely be a smoother signal passing through the middle of the observed noised data. Figure 5.4 shows an illustrative smooth signal with its noisy version as well as the tensor density estimate, where $\mathbf{x} = [1, x_1, x_2]^T$ and the order of the tensor $\mathbf{T}$, $p = 2$. The tensor density estimate, which involves symmetric tensor model, is given by

$$p(\mathbf{x}) = e^{-<\mathbf{T}, \mathbf{x}^{\circ p}>^2}, \text{ where } \mathbf{T} = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i^{\circ p}. \tag{5.38}$$

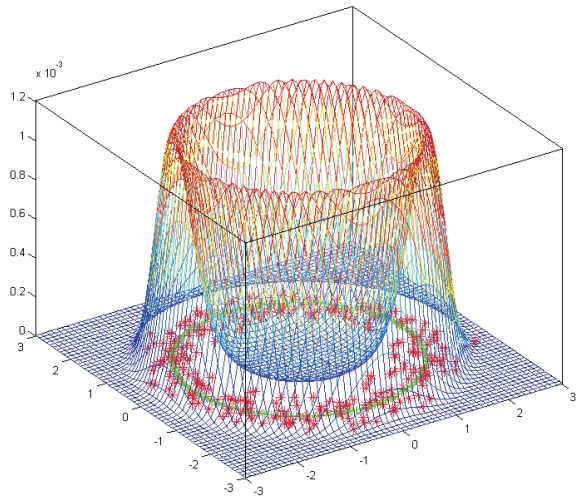


FIGURE 5.4: Noisy data set (red dots), underlying signal (green dots), and its density estimation by a symmetric tensor model.

As it was proposed in [81], principal curve projection can be achieved by a likelihood maximization in a constrained space. Below we describe the main properties and an algorithm which helps us to achieve the principal curve based on the tensor model. Let's have a density estimate as in (5.38) then to get its principal curve we need to analyze its gradient, $\mathbf{g}(\mathbf{x})$, and Hessian matrix, $\mathbf{H}(\mathbf{x})$. Due to monotonicity of *log* and *exp* functions we can operate with $\log p(\mathbf{x})$ preserving directions of the first and second derivatives. So the transformation $p(\mathbf{x}) \overset{\nabla}{\rightarrow} \mathbf{g}(\mathbf{x}) \overset{\nabla^T \nabla}{\rightarrow} \mathbf{H}(\mathbf{x})$ is equivalent to $\log p(\mathbf{x}) \overset{\nabla}{\rightarrow} \mathbf{g}_{\log p}(\mathbf{x}) \overset{\nabla^T \nabla}{\rightarrow} \mathbf{H}_{\log p}(\mathbf{x})$, where the *log* of Hessian matrix can be factorized as $\mathbf{H}_{\log p} = \sum_{i=1}^{n} \gamma_i(\mathbf{x}) \mathbf{v}_i(\mathbf{x}) \mathbf{v}_i^T(\mathbf{x})$.

In terms of these definitions, the n-dimensional principal surface is the set of data $\mathbf{x} \in \mathbb{R}^n$, which satisfies [76]:

- $\frac{\mathbf{g}(\mathbf{x})^T\mathbf{H}(\mathbf{x})\mathbf{g}(\mathbf{x})}{\|\mathbf{g}(\mathbf{x})\|\|\mathbf{H}(\mathbf{x})\mathbf{g}(\mathbf{x})\|} = \pm 1$.
- $\gamma_i(\mathbf{x}) < 0$ for all except one $i = 1, \ldots, n$.

The gradient and Hessian matrix of $\log p(\mathbf{x})$ in the tensor terms are:

$$\begin{aligned} \mathbf{g}_{\log p}(\mathbf{x}) &= -2 < \mathbf{T}, \mathbf{x}^{\circ p} > \frac{\partial < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}^T}, \\ \mathbf{H}_{\log p}(\mathbf{x}) &= -2\frac{\partial < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}^T}\frac{\partial < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}} - 2 < \mathbf{T}, \mathbf{x}^{\circ p} > \frac{\partial^2 < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}^T \partial \mathbf{x}}. \end{aligned} \tag{5.39}$$

The first and second derivatives of Frobenius norm of two rank-1 tensors are

$$< \mathbf{u}^{\circ p}, \mathbf{x}^{\circ p} >= (\mathbf{u}^T\mathbf{x})^p \overset{\nabla}{\rightarrow} p(\mathbf{u}^T\mathbf{x})^{p-1}\mathbf{u} \overset{\nabla^T\nabla}{\rightarrow} p(p-1)(\mathbf{u}^T\mathbf{x})^{p-2}\mathbf{u}\mathbf{u}^T, \tag{5.40}$$

and corresponding derivatives for the full rank tensors defined by:

$$\begin{aligned} \frac{\partial < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}^T} &= \sum_{i=1}^r \lambda_i \frac{\partial (\mathbf{u}_i^T\mathbf{x})^p}{\partial \mathbf{x}^T} = p\sum_{i=1}^r \lambda_i (\mathbf{u}_i^T\mathbf{x})^{p-1}\mathbf{u}_i, \\ \frac{\partial^2 < \mathbf{T}, \mathbf{x}^{\circ p} >}{\partial \mathbf{x}^T \partial \mathbf{x}} &= p(p-1)\sum_{i=1}^r \lambda_i (\mathbf{u}_i^T\mathbf{x})^{p-2}\mathbf{u}_i\mathbf{u}_i^T. \end{aligned} \tag{5.41}$$

The Algorithm 8 presents the overall approach. For short let's denote $\mathbf{g}_{\mathbf{x}_i} = g_{\log p(\mathbf{x}_i)}(\mathbf{x}_i)$ and $\mathbf{H}_{\mathbf{x}_i} = \mathbf{H}_{\log p(\mathbf{x}_i)}(\mathbf{x}_i)$. In order to be acquaint with properties and convergence proof we refer the reader to [82].

Algorithm 8 Principal curve estimate.

Estimate p.d.f., $p(\mathbf{x})$, on the basis of $\mathbf{x}_i,\ i = 1, \ldots, k$.
for each data point $\mathbf{x}_i$ **do**
  repeat
    Evaluate local gradient $\mathbf{g}_{\mathbf{x}_i}$ and Hessian matrix $\mathbf{H}_{\mathbf{x}_i}$.
    Move $\mathbf{x}_i$ along ascent direction $\mathbf{g}_{\mathbf{x}_i}^T\mathbf{H}_{\mathbf{x}_i}$.
  until $\frac{\mathbf{g}_{\mathbf{x}_i}^T\mathbf{H}_{\mathbf{x}_i}\mathbf{g}_{\mathbf{x}_i}}{\|\mathbf{g}_{\mathbf{x}_i}\|\|\mathbf{H}_{\mathbf{x}_i}\mathbf{g}_{\mathbf{x}_i}\|} \approx \pm 1$
end for

Figure 5.11 presents an example of density estimation by an order-3 symmetric tensor for hyperbolic data distribution. Red dots on the top of the surface denote the principal curve of data (blue dots).

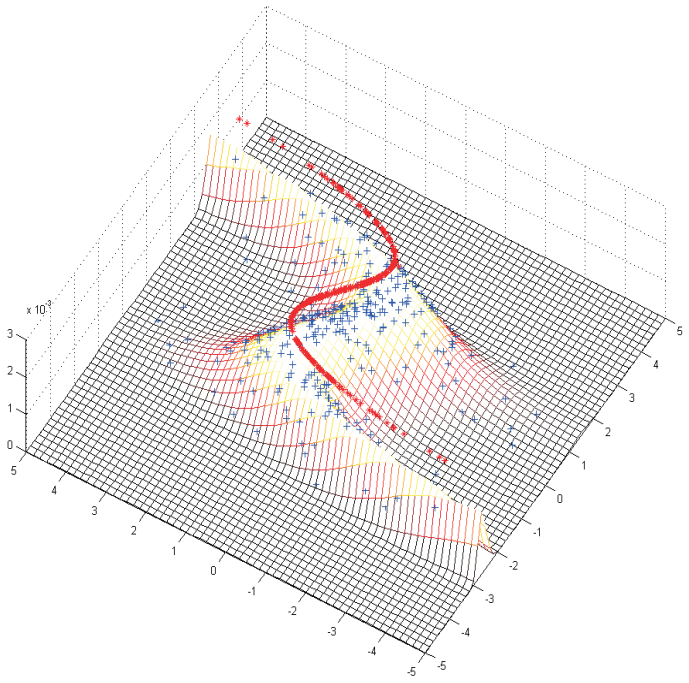


FIGURE 5.5: Hyperbolic data distribution and corresponding principal curve fitting (3-dimensional order-3 tensor $\mathbf{T}$).

5.2.2 Clustering

Mean shift [83] is a very commonly used non-parametric iterative feature-space technique that maps the data points to the local maxima of the probability density, where the gradient is equal to zero and all eigenvalues of the Hessian are non-positive [84]. Application domains widely include clustering in image processing, computer vision [78]. Here we briefly describe mean-shift algorithm and extend it by proposed tensor density estimate.

Let's have some data set $\mathbf{x}_i,\ i = 1, \ldots, k$, its initial estimate $\mathbf{x}$, and kernel function $K(\mathbf{x}, \mathbf{y})$. This function determines the weights of points $\mathbf{x}_i$, which are close to $\mathbf{x}$, and used for re-estimation of their means. Generally the Gaussian kernel is used for such an estimate, $K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/\sigma}$, then the weighted mean of local density is defined as

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in N(\mathbf{x})} K(\mathbf{x}_i, \mathbf{x})\mathbf{x}_i}{\sum_{\mathbf{x}_i \in N(\mathbf{x})} K(\mathbf{x}_i, \mathbf{x})}, \tag{5.42}$$

where $N(\mathbf{x})$ defines the neighbor data set of $\mathbf{x}$. At the end, the mean-shift algorithm sets $\mathbf{x} \leftarrow \mathbf{m}(\mathbf{x})$ and repeats the estimation until $\mathbf{m}(\mathbf{x})$ converges to some stable state.

Selection of the Gaussian kernel bandwidth is a significant stage in the implementation that severely influents on the performance of the algorithm. It is known that an

appropriate window, or kernel, size, corresponds to an appropriate density estimate. Literature on density estimation is rather rich on the domain of the kernel width selection, especially for data sets of low to moderate dimensionality, and methods in the literature are extended from simple heuristics to more technically approaches like maximum likelihood [79]. All above mentioned techniques are general purposed kernel bandwidth selection methods that blindly approach the data. Particularly for this problem, one can also select the kernel bandwidth by considering the actual physical meaning of the data. The Renyi entropy estimate is directly connected to Parzen windowing [85]. Silverman's rule [75] is considered to be one of the simplest and is given by

$$\sigma = \sigma_{\mathbf{x}}[\frac{4}{(2n+1)k}]^{\frac{1}{n+4}}, \tag{5.43}$$

where $\sigma_{\mathbf{x}}^2 = n^{-1}\sum_i \Sigma_{x_{ii}}$, and $\Sigma_{x_{ii}}$ are the diagonal elements of the sample covariance matrix. Unless otherwise stated, the window size is determined using this rule.

For a tensor density function (5.38) the mean-shift is transformed to

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{k} \mathbf{x}_i e^{-<\mathbf{T},\mathbf{x}_i^{\circ p}>^2 - \|\mathbf{x}-\mathbf{x}_i\|^2/\sigma}}{\sum_{i=1}^{k} e^{-<\mathbf{T},\mathbf{x}_i^{\circ p}>^2 - \|\mathbf{x}-\mathbf{x}_i\|^2/\sigma}}. \tag{5.44}$$

An example of clustering problem is presented on Figure 5.11(c), where we have a mixture of normal distributed points [78]. Estimated density (by an order-4 symmetric tensor) was applied using mean-shift algorithm to converge the data points to their local maxima.

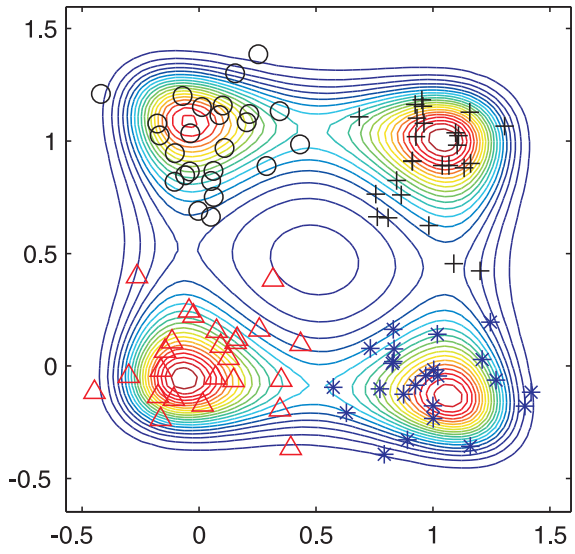


FIGURE 5.6: Density estimation and corresponding mean-shift clustering (3-dimensional order-4 symmetric tensor $\mathbf{T}$).

5.3 Generalized PCA

This section has been prepared based on the publications of Prof. Rene Vidal from The Johns Hopkins University (MD\USA). Particularly, we refer the reader to his publication [86] to learn more comprehensive explanations on Generalized PCA.

In this section we describe an approach to the problem of linear subspaces mixture estimation in data set, which is known as Generalized Principal Component Analysis (GPCA) problem [86]. The GPCA is equivalent to factoring a homogeneous polynomial degree of which is the number of subspaces and factors of which represent normal vectors to each subspace, where any homogeneous polynomial can be presented by a rank-r symmetric tensor or a rank-1 nonsymmetric one. We describe an approach for an estimate of the number of subspaces. As a solution of the GPCA problem in the presence of noise or data nonlinearity, we can utilize constrained nonlinear least squares optimization as well as kernel PCA, which transforms data to linear form. Application of the GPCA can be widely used to solve the computer vision problems, e.g., when we need to analyze the mixture of objects motion, data compression, regression, etc.

The standard PCA problem estimate a linear subspace $S \subset \mathbb{R}^n$ of unknown dimension $m < n$ from k data points $\mathbf{x}_j \in S,\ j = 1, \ldots, k$, and can be solved by applying the SVD to the data set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k] \in \mathbb{R}^{n \times k}$ [86]. In probabilistic PCA [87] the subspace is estimated in Maximum Likelihood sense using a probabilistic

generative model, where we assume that data points within each subspace are drawn from an unknown probability distribution. The membership of the data points is estimated by multinomial density distribution. Nonlinear PCA (NLPCA) [88] is considered in the following Section. For this method a subspace is estimated after applying a nonlinear transformation to the data.

Let a set of data points $\mathbf{X} = \{\mathbf{x}_j \in \mathbb{R}^n\},\ j = 1, \ldots, k$ is given and drawn from $p > 1$ linear subspaces $\{S_i \subset \mathbb{R}^n\},\ i = 1, \ldots, p$ of dimension $m,\ 0 < m < n$. By identifying the subspaces S_i without prior knowledge about belonging of data $\mathbf{x}_j$ we mean the following [86]:

1. Identify the number of subspaces and their dimensions;
2. Identify a basis for each subspace S_i or its normal vector;
3. Group or segment given k data points into the subspace(s);

In the GPCA the data points $\mathbf{x}_j,\ j = 1, \ldots, k$ are drawn from p m-dimensional linear subspaces of $\mathbb{R}^n, S_i,\ i = 1, \ldots, p$, as shown on Figure 5.7 for $p = 3, m = \{1, 1, 2\}$, and $n = 3$ [86]. In this case, the problem consists in identifying of each subspace without any prior knowledge of the number of subspace and belongings of data points to a subspace.

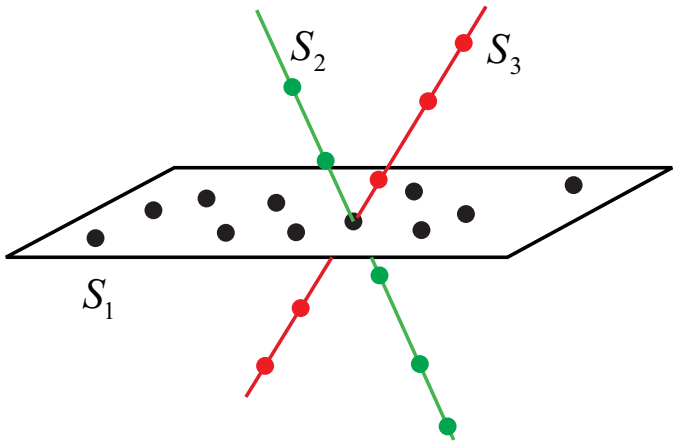


FIGURE 5.7: GPCA for 3 linear subspaces $S_1 \in \mathbb{R}^2, S_2 \in \mathbb{R}^1, S_3 \in \mathbb{R}^1$ in 3-dimensional data space, with equivocal interpretation of S_2 and S_3, which can be considered as one 2-dimensional data set.

In the noiseless data, GPCA interprets the number of subspaces as degree of the certain polynomial, that is multi-dimensional higher-order tensor, and the normals

to each subspace become the factors of such a tensor. We will show how to obtain the number of subspaces p from the rank of a certain matrix which depends on the data in presence of noise and without it. For the given n and p, the estimation of the subspaces $S_i \subset \mathbb{R}^n$ is essentially equivalent to tensor factorization problem.

In the following explanation we will present a solution for the special case of known dimension of subspaces, $m = n - 1$. The general case, where $0 < m < n$, can be obtained by consecutive applying of GPCA to a separated subspace of higher dimension, $m + 1$. It turns out that the general case can be always reduced to the special case, as long as all the subspaces $S_i, i = 1, \ldots, p$ have the same dimension m [86].

5.3.1 GPCA Subspaces

The GPCA problem for $m = n - 1$ is algebraically equivalent to factoring of a homogeneous polynomial of degree p in n dimensional space into a product of p polynomials of degree 1 [86]. Any $(n-1)$-dimensional space $S_i \subset \mathbb{R}^n$ can be represented by a nonzero normal vector $\mathbf{b}_i \in \mathbb{R}^n$ as

$$S_i = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{b}_i^T \mathbf{x} = b_{i1}x_1 + b_{i2}x_2 + \cdots + b_{i1}x_n = 0\}. \tag{5.45}$$

For clarity, we need to point out one of the most important structure properties for vector $\mathbf{x}$. One of its elements must be always equal to 1 (usually we denote 1^{th} entry as in the case of density estimation). It does not define any coordinate position in the data space but it is used as a bias of liner subspace. Otherwise, all our linear subspaces must be pass through the origin. So in fact for the GPCA problem a n-dimensional vector $\mathbf{x}$ defines one data point in $(n-1)$-dimensional data space. Since we consider distinct subspaces S_i, normal vectors $\mathbf{b}_i,\ i = 1, \ldots, p$, are pairwise linearly independent. Let's imagine that we have a point $\mathbf{x} \subset \mathbb{R}^n$ lying on one of the subspaces S_i. Such a point must satisfy the equation [86]:

$$(\mathbf{b}_1^T \mathbf{x} = 0) \cup (\mathbf{b}_2^T \mathbf{x} = 0) \cup \cdots \cup (\mathbf{b}_p^T \mathbf{x} = 0), \tag{5.46}$$

which is equivalent to the order-p homogeneous polynomial:

$$f_p(\mathbf{x}) = \prod_{i=1}^{p} (\mathbf{b}_i^T \mathbf{x}) = 0. \tag{5.47}$$

Now the problem of identifying S_i is equivalent to the problem of finding the vectors $\mathbf{b}_i,\ i = 1, \ldots, p$ from the nonlinear equation (5.47) [86].

As we know from the previous explanation on symmetric tensor, any order-p n-dimensional homogenous polynomial can be defined by an order-p n-dimensional tensor $\mathbf{A}$, where $\mathbf{A} = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i^{\circ p}$. So in the tensor form the equation (5.47) is transformed to

$$f_p(\mathbf{x}) =< \mathbf{A}, \mathbf{x}^{\circ p} >= \sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x})^p = 0. \tag{5.48}$$

Now the problem of identifying S_i is then equivalent to the factorization problem of the tensor $\mathbf{A}$. The vectors $\mathbf{b}_i,\ i = 1, \ldots, p$, are factors which decompose symmetric rank-r tensor $\mathbf{A}$ as a non-symmetric rank-1 tensor $\mathbf{B}$, where $\mathbf{B} = \mathbf{b}_1 \circ \mathbf{b}_2 \circ \cdots \circ \mathbf{b}_p$ and vectors $\mathbf{b}_i,\ i = 1, \ldots, p$ are different and independent. The trick is to evaluate the difference between tensors $\mathbf{A}$ and $\mathbf{B}$ because the sum of element-wise square errors $e^2 = \|\mathbf{A} - \mathbf{B}\|_F^2$ will be always greater than zero. To produce such a decomposition we need to evaluate the square difference between the sums of tensor entries under permutation of their indices. Let's order-2 2-dimensional tensors $\mathbf{A}$ and $\mathbf{B}$ are given by

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}. \tag{5.49}$$

Then the square error of the sums of tensor entries under permutation of their indices is

$$e^2 = (a_{11} - b_{11})^2 + ((a_{12} + a_{21}) - (b_{12} + b_{21}))^2 + (a_{22} - b_{22})^2. \tag{5.50}$$

The general square error for such a decomposition is evaluated by

$$e^2 = \sum_{i_c} \left(\sum_{i_p} (\mathbf{A})_{i_p} - \sum_{i_p} (\mathbf{B})_{i_p} \right)^2, \tag{5.51}$$

where index i_c denotes all unique combinations of tensor indexes and index i_p denotes all possible combinations in i_c.

Let $\mathbf{g}(n, p) = \mathbf{g}[x_1, \ldots, x_n]$ be some homogeneous polynomial of degree p in n variables, where $\mathbf{g}(n, p)$ is generated by the set of monomials $\mathbf{x}^p = x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$, with $0 \leq p_j \leq n,\ j = 1, \ldots, n$, and $p_1 + p_2 + \cdots + p_n = p$. So the number of unique monomials is equal to the number of free elements in a n-dimensional order-p symmetric tensor $\mathbf{A}$ [86], i.e.,

$$m(n, p) = \binom{p + n - 1}{p}, \tag{5.52}$$

where the dimension of $\mathbf{g}(n,p)$ is equal to $m(n,p)$. Therefore, we can transform data space from $\mathbb{R}^n$ into $\mathbb{R}^{m(n,p)}$ [86].

Here we describe Veronese map which will be utilized to the problem of subspaces estimation in absence of noise. Let p and n are given, the Veronese map of degree p, $v_p : \mathbb{R}^n \to \mathbb{R}^{m(n,p)}$, is defined as:

$$v_p : [x_1, \ldots, x_n]^T \to [\ldots, \mathbf{x}^{\mathbf{p}}, \ldots]^T, \tag{5.53}$$

where $\mathbf{x}^p$ is a monomial of the form $x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$ [86]. So (5.47) becomes the linear expression with vector of coefficients $\mathbf{a} \in \mathbb{R}^{m(n,p)}$ which defines all unique entries in an order-p n-dimensional tensor $\mathbf{A}$:

$$f_p(\mathbf{x}) = v_p^T(\mathbf{x})\mathbf{a} = \sum (\mathbf{A})_{p_1,\ldots,p_n} x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n} = 0, \tag{5.54}$$

where $a_{\mathbf{p}} \in \mathbb{R}$ represents the coefficient of the monomial $\mathbf{x}^{\mathbf{p}}$ [86].

5.3.2 The Number of Subspaces with Absence of Noise

Applying (5.54) to a given data set $\mathbf{x}_j,\ j = 1, \ldots, k$ such that $k \geq m(n,p) - 1$ gives us the system of linear equations on the vector of coefficients $\mathbf{a}$ [86]

$$\mathbf{L}_p\mathbf{a} = \begin{cases} v_p^T(\mathbf{x}_1)\mathbf{a} = 0 \\ v_p^T(\mathbf{x}_2)\mathbf{a} = 0 \\ \quad\vdots \\ v_p^T(\mathbf{x}_k)\mathbf{a} = 0 \end{cases} \tag{5.55}$$

Since this linear system depends only on the number of subspaces, p, we need to know p in advance in order to estimate $\mathbf{a}$. So the estimation of the number of subspaces p is very much related to the conditions under which the solution for $\mathbf{a}$ is unique [86]. Let's assume that a collection of $k \geq m(n,p) - 1$ data points $\mathbf{x}_j,\ j = 1, \ldots, k$ on p different $(n-1)$-dimensional subspaces of $\mathbb{R}^m$ is given. Let $\mathbf{L}_i \in \mathbb{R}^{k \times m(n,i)}$ be the matrix defined above, but computed with the Veronese map $v_i(x)$ of degree i. If the data points are in general positions and at least $m-1$ points correspond to each subspace then [86]:

$$rank(\mathbf{L}_i) \begin{cases} > m(n,i) - 1, & i < p, \\ = m(n,i) - 1, & i = p, \\ < m(n,i) - 1, & i > p. \end{cases} \tag{5.56}$$

Therefore, the number p of subspaces is given by [86]:

$$p = min\{i : rank(\mathbf{L}_i) = m(n,i) - 1\}. \tag{5.57}$$

So there is no polynomial of degree $i < p$ that is satisfied by all the data, hence $rank(\mathbf{L}_i) = m(n,i)$ for $i < p$. Conversely, there are multiple polynomials of degree $i > p$, namely any multiple of $f_p(\mathbf{x})$, which are satisfied by all the data, hence $rank(\mathbf{L}_i) < m(n,i) - 1$ for $i > p$. Thus the case $i = p$ is the only one in which system has a unique solution, namely the coefficients $\mathbf{a}$ of the polynomial $f_p(\mathbf{x})$ [86].

5.3.3 The Number of Subspaces with Presence of Noise

Data without noise exist just in theory. Here we describe a kernel PCA which is based on the concept of maximum entropy preservation [85] and can be utilized for estimation of the number of subspaces in data as well as transformation of nonlinear subspaces into linear ones. The method, named kernel maximum entropy is based on Renyi's quadratic entropy estimated via Parzen windowing, which was described in the previous Section, 5.43. The data transformation is obtained using eigenvectors of the data affinity matrix [27]. These eigenvectors are in general not the same as those ones used in kernel PCA. The Renyi quadratic entropy is given [89] by

$$H_2(\mathbf{x}) = -log \int f^2(\mathbf{x}) d\mathbf{x}, \tag{5.58}$$

where $f(\mathbf{x})$ is the density associated with the random variable $\mathbf{x}$ and n-dimensional data set $\mathbf{x}_i,\ i = 1, \ldots, k$ is generated from $f(\mathbf{x})$. A non-parametric estimator for $H_2(\mathbf{x})$ is obtained by replacing the actual p.d.f. by its Parzen window estimator as

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} W_\sigma(\mathbf{x}, \mathbf{x}_i),\ W_\sigma(\mathbf{x}, \mathbf{x}_i) = \frac{1}{(2\pi\sigma^2)^{n/2}} exp\{-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\}. \tag{5.59}$$

Alongside with Gaussian window we can utilize any appropriate one, but it must be a density itself. So we have

$$\begin{gathered}\hat{H}_2(\mathbf{x}) = -\log \int \hat{f}^2(\mathbf{x}) dx = -log \tfrac{1}{k^2} \textstyle\sum_{i=1}^{k} \sum_{i=1}^{k} \int W_\sigma(\mathbf{x}, \mathbf{x}_i) W_\sigma(\mathbf{x}, \mathbf{x}_j) dx = \\ -\log \tfrac{1}{k^2} \textstyle\sum_{i=1}^{k} \sum_{i=1}^{k} W_{\sqrt{2}\sigma}(\mathbf{x}_i, \mathbf{x}_j),\end{gathered} \tag{5.60}$$

where in the last step we employ the convolution theorem for Gaussian functions.

For notational simplicity, we denote $W_{\sqrt{2}\sigma}(\mathbf{x}_i, \mathbf{x}_j)$ as a value of kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ and present the kernel matrix $\mathbf{K_x}$, such that element (i, j) of $\mathbf{K_x}$ is equal to $k(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, \ldots, k$. So the Renyi quadratic entropy may be expressed compactly in the terms of the kernel matrix as $\hat{H}_2(\mathbf{x}) = -log\frac{1}{k^2}\mathbf{1}^T\mathbf{K_x}\mathbf{1}$, where $\mathbf{1}$ is a $(k \times 1)$ ones-vector. Since the logarithm is a monotonic function, we can focus on the quantity $V(\mathbf{x}) = \frac{1}{k^2}\mathbf{1}^T\mathbf{K}\mathbf{1}$.

The maximum entropy method is mapping $\mathbf{X} \to \mathbf{Y}$, such that the entropy associated with $\mathbf{Y}$ is maximally similar to the entropy of $\mathbf{X}$. Since we are concerned with Renyi's entropy, therefore such a data mapping results in a $V(\mathbf{y}) = \frac{1}{k^2}\mathbf{1}^T\mathbf{K_y}\mathbf{1}$, in the terms of the $\mathbf{Y}$ data set, should be as close as possible to $V(\mathbf{x}) = \frac{1}{k^2}\mathbf{1}^T\mathbf{K_x}\mathbf{1}$. This means that the kernel matrix $\mathbf{K_y}$ must be maximally similar to $\mathbf{K_x}$ in some sense. According to the fact that our input data space is the kernel matrix, we do not actually want to obtain $\mathbf{Y}$.

Let the kernel matrix $\mathbf{K_x}$ be decomposed as $\mathbf{EDE}^T$ and $\mathbf{\Phi_x}$ be such a matrix each column of which represents an approximation of the corresponding kernel feature space data point in the set $\{\mathbf{\Phi}(\mathbf{x}_1), \mathbf{\Phi}(\mathbf{x}_2), \ldots, \mathbf{\Phi}(\mathbf{x}_k)\}$. So an approximation which preserves inner-products is given by $\mathbf{\Phi_x} = \mathbf{D}^{1/2}\mathbf{E}^T$, then $\mathbf{K_x} = \mathbf{\Phi_x}^T\mathbf{\Phi_x} = \mathbf{EDE}^T$. Now we can define a dimensionality reduction in the kernel space, obtaining the m-dimensional $\mathbf{\Phi_y}$ from $\mathbf{\Phi_x}$, yielding $\mathbf{K_y} = \mathbf{\Phi_y}^T\mathbf{\Phi_y}$ such that $V(\mathbf{y}) \approx V(\mathbf{x})$. According to [90], the value $V(\mathbf{x})$ is given by

$$V(\mathbf{x}) = \frac{1}{k^2}\sum_{i=1}^{k}\lambda_i(\mathbf{1}^T\mathbf{e}_i)^2 = \frac{1}{k^2}\sum_{i=1}^{k}\lambda_i\gamma_i^2, \tag{5.61}$$

where $\mathbf{e}_i$ is the eigenvector corresponding to the i^{th} column of $\mathbf{K_x}$, and $\mathbf{1}^T\mathbf{e}_i = \gamma_i$ and the products $\lambda_i\gamma_i^2$ have been sorted in decreasing order. If we approximate $V(\mathbf{x})$ using only m terms of its sum, we must use the m first terms in order to achieve minimum approximation error. This invokes using of the m largest $\lambda_i\gamma_i^2$. Let's define the data set $\mathbf{\Phi}_y = \mathbf{D}^{1/2}\mathbf{E}^T$, using the m eigenvalues and eigenvectors of $\mathbf{K_x}$ corresponding to the m largest products $\lambda_i\gamma_i^2$. Hence, $\mathbf{K_y} = \mathbf{\Phi_y}^T\mathbf{\Phi_y} = \mathbf{E}_m\mathbf{D}_m^{1/2}\mathbf{D}_m^{1/2}\mathbf{E}_m^T = \mathbf{E}_m\mathbf{D}_m\mathbf{E}_m^T$, and

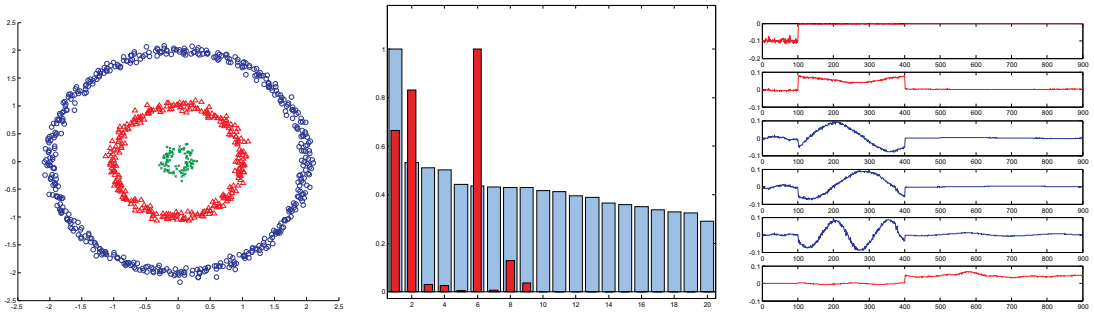
$$V(\mathbf{y}) = \frac{1}{k^2}\sum_{i=1}^{m}\lambda_i\gamma_i^2 = \frac{1}{k^2}\mathbf{1}^T\mathbf{K_y}\mathbf{1}, \tag{5.62}$$

the best approximation to the entropy estimate $V(\mathbf{x})$ using m eigenvalues and eigenvectors. So we refer to the mapping $\mathbf{\Phi_y} = \mathbf{D}_m^{1/2}\mathbf{E}_m^T$ as a maximum entropy data

transformation in a kernel feature space. We need to mention that this is not the same as the PCA dimensionality reduction, which is defined as $\mathbf{\Phi}_{PCA} = \mathbf{D}_m^{1/2}\mathbf{E}_m^T$, using the eigenvalues and eigenvectors corresponding to the m largest eigenvalues of $\mathbf{K_x}$. So the kernel maximum entropy procedure is given by

$$\begin{array}{ccc} \mathbf{K_x} = \mathbf{E}\mathbf{D}\mathbf{E}^T & \rightarrow & \mathbf{\Phi_x} = \mathbf{D}^{1/2}\mathbf{E}^T \\ \updownarrow & & \updownarrow \\ \mathbf{K_y} = \mathbf{E}_m\mathbf{D}_m\mathbf{E}_m^T & \leftarrow & \mathbf{\Phi_y} = \mathbf{D}_m^{1/2}\mathbf{E}_m^T \end{array} \tag{5.63}$$

Figure 5.8(a) shows a ring-shaped data set consisting of 3 clusters. The original data set is ordered as a sequence of data points: 100 (green), 300 (red), and 500 (blue). The two bar graphics on Figure 5.8(b) present the 20 largest normalized eigenvalues λ_i (blue) and entropy terms $\lambda_i\gamma_i^2$ (red) corresponding to the largest eigenvalues. We need to point out that the entropy terms corresponding to the 1^{st}, 2^{nd}, and 6^{th} eigenvalues are significantly larger than the rest and define the exact number of clusters in the data set that is impossible by standard KPCA approach. Figure 5.8(c) shows the first six eigenvectors, where the vectors with maximum entropy (red) carry information about the cluster structure with their blockwise appearance.

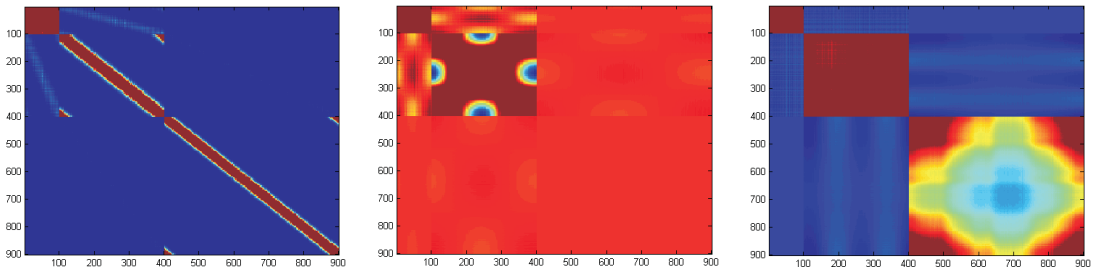


(a) (b) (c)

FIGURE 5.8: Results of data analysis using standard and entropy KPCA: (a) initial data set; (b) normalized eigenvalues of standard KPCA (blue) and maximum entropy KPCA (red); (c) first 6 eigenvectors of $\mathbf{K_x}$.

The affinity matrices $\mathbf{K_x}$ for the described data set are shown on Figure 5.9(a). In ideal situation, the task of any clustering approach is to transform such a matrix into a blockwise one. Figure 5.9(b) shows the KPCA approximation of $\mathbf{K_x}$, obtained from the first three eigenvectors, which do not provide us blockwise structure. In

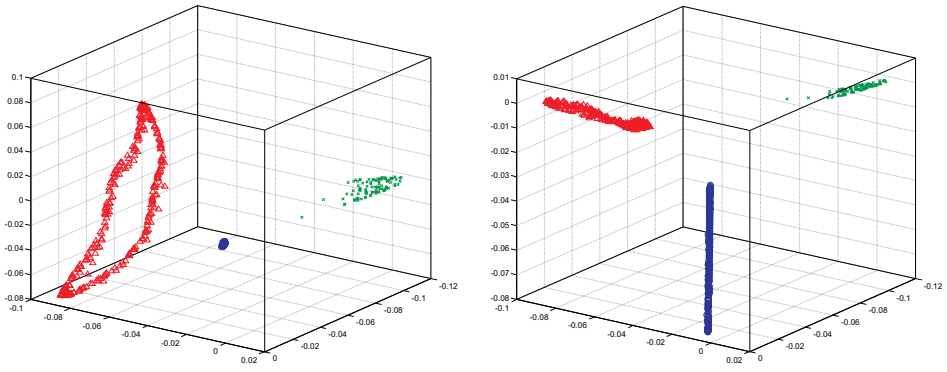
contrast, the entropy KPCA approximation $\mathbf{K_y}$, Figure 5.9(c), shows the blockwise appearance.



(a) (b) (c)

FIGURE 5.9: Affinity matrices: (a) initial data set; (b) standard KPCA; (c) maximum entropy KPCA.

On Figure 5.10 the data transformation is shown via (a) standard KPCA and (b) maximum entropy KPCA. We note that in the case of entropy KPCA the clusters are located along different lines radially from the origin. These lines are almost orthogonal to each other, which is suitable for GPCA model. The results of data transformation by standard KPCA is significantly different and the mean vectors of the clusters are not spread angularly.



(a) (b)

FIGURE 5.10: Data transformation by: (a) standard KPCA and (b) maximum entropy KPCA.

5.3.4 Estimation of the Subspaces with Absence of Noise

The linear system (5.54) allows us to determine the number of subspaces, p, and the vector of coefficients, $\mathbf{c}$, directly from the data set $\mathbf{x}_j,\ j = 1, \ldots, k$. The rest of the problem now is to recover the normal vectors $\mathbf{b}_i,\ j = 1, \ldots, p$ from $\mathbf{a}$. From (5.47) and (5.54) we have

$$f_p(\mathbf{x}) = \sum a_{\mathbf{p}} \mathbf{x}^{\mathbf{p}} =< \mathbf{A}, \mathbf{x}^{\circ p} >= \prod_{i=1}^{p} \left(\sum_{j=1}^{n} b_{ij} x_j \right). \tag{5.64}$$

Therefore, recovering $\mathbf{b}_i,\ j = 1, \ldots, p$ from $\mathbf{A}$ is equivalent to factoring a given homogeneous polynomial $f_p(\mathbf{x}) \in \mathbf{g}(n, p)$, into p distinct polynomials in $\mathbf{g}(n, 1)$ [86]. We can interpret $\mathbf{A}$ as a symmetric tensor representation of the non-symmetric part of the tensor $\mathbf{b}_1 \circ \cdots \circ \mathbf{b}_p$. In this case estimating $\mathbf{b}_i,\ j = 1, \ldots, p$ from $\mathbf{A}$ is equivalent to factoring such a symmetric tensor. We utilize the proposed model of rank-r symmetric tensor decomposition to obtain coefficients in $\mathbf{A}$ and then we apply the model of rank-1 non-symmetric tensor to get a set of vectors $\mathbf{b}_1 \circ \cdots \circ \mathbf{b}_p$. The reader can learn an initial approach to solve such a problem, which is based on consecutive reducing of data dimensionality, in [91].

The linear subspace estimation can be defined as a solution of a nonlinear least square problem:

$$\min_{\mathbf{b}_1, \ldots, \mathbf{b}_p} e^2 = \min_{\mathbf{b}_1, \ldots, \mathbf{b}_p} \sum_{j=1}^{k} \left(\prod_{i=1}^{p} (\mathbf{b}_i^T \mathbf{x}_j) \right)^2, \tag{5.65}$$

where $\mathbf{b}_i$ is a vector of linear coefficients of i^{th} subspace in $\mathbb{R}^n$, and a vector $\mathbf{x}_j = [1, x_1, \ldots, x_n]^T$. Parameter p defines the number of linear subspaces and k - the number of data points. In tensor form we can present this problem as

$$\min_{\lambda_1, \ldots, \lambda_r, \boldsymbol{\Theta}} e^2 = \min_{\lambda_1, \ldots, \lambda_r, \boldsymbol{\Theta}} \sum_{j=1}^{k} \left(\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x}_j)^p \right)^2. \tag{5.66}$$

To minimize such cost function we need to find appropriate values of the linear coefficients, $\lambda_i,\ i = 1, \ldots, r$, and rotation Givens angles, $\boldsymbol{\Theta}$, which define the position of vector frame $\mathbf{U}$ in $\mathbb{R}^n$ data space. From the proposed method of tensor decomposition we know that linear coefficients are estimated for each unique value of $\boldsymbol{\Theta}$.

Let's take partial derivatives of e^2 with respect to $\lambda_i,\ i = 1, \ldots, r$:

$$\begin{cases} \frac{\partial e^2}{\partial \lambda_1} = 2 \sum_{j=1}^{k} \left(\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x}_j)^p \right) (\mathbf{u}_1^T \mathbf{x}_j)^p = 0 \\ \vdots \\ \frac{\partial e^2}{\partial \lambda_r} = 2 \sum_{j=1}^{k} \left(\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x}_j)^p \right) (\mathbf{u}_r^T \mathbf{x}_j)^p = 0 \end{cases} \tag{5.67}$$

Now we have a system of r linear equations with r unknown:

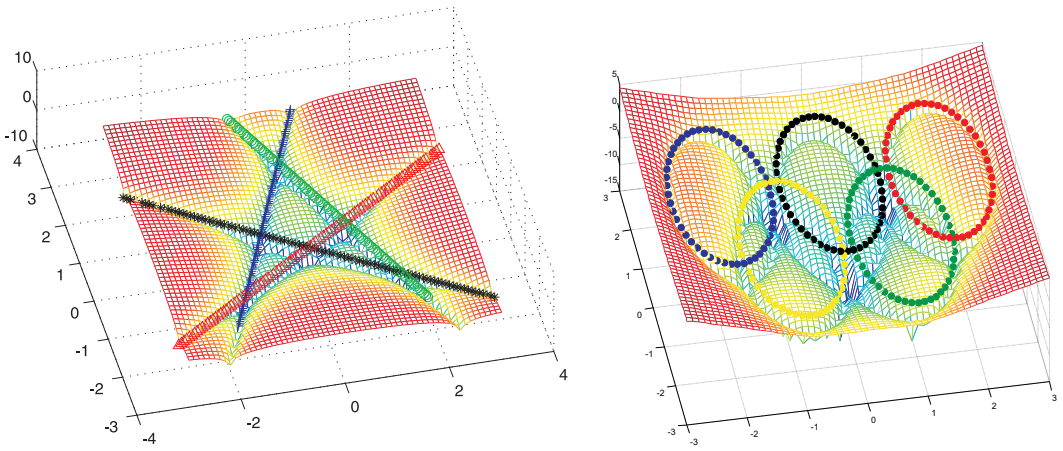
$$\begin{gathered} \mathbf{M} \lambda^T = 0, \text{ s.t. } \lambda_i \neq 0, \\ \text{where } m_{l_1 l_2} = \sum_{j=1}^{k} (\mathbf{u}_{l_1}^T \mathbf{x}_j)^p (\mathbf{u}_{l_2}^T \mathbf{x}_j)^p, \text{ and } i, l_1, l_2 = 1, \ldots, r. \end{gathered} \tag{5.68}$$

As an appropriate solution for such linear system we can utilize the Eigendecomposition to the matrix $\mathbf{M}$ and assign the eigenvector which corresponds to minimal eigenvalue as a solution of λ^T. Now to minimize the cost function e^2 we need to estimate the Givens angles, $\mathbf{\Theta}$. We perform this operation on the basis of the gradient symmetric tensor model, see Chapter 4. When the symmetric full-rank tensor $\mathbf{A}$ is given we need to factorize it as a non-symmetric rank-1 tensor $\mathbf{B}$, (5.51). The equation (5.51) is a multidimensional order-2 polynomial the root of which can be found, for instance, by the Newton's method. Note that the roots of the equation (5.51) are normal vectors to subspaces $S_i,\ i = 1, \ldots, p$ and can be normalized as soon as the solution is reached. Once we have normalized vectors $\mathbf{b}_i,\ i = 1, \ldots, p$ we can segment data space $\mathbb{R}^n$ onto p linear subspaces $S_i \in \mathbb{R}^{(n-1)}$ via distances to the subspaces $(\mathbf{b}_i^T \mathbf{x})$.

Figure 5.11(a) shows an example of an application of a symmetric tensor to GPCA problem. The data points (marked by different colors) denote result of the estimation of intersected linear subspaces and the surface on the figure denotes error for all data space. Figure 5.11(b) presents a segmentation problem when data points were expanded to the second order data space, i.e. $\mathbf{x} = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2]^T$.

5.3.5 Estimation of the Subspaces with Presence of Noise

In the previous subsection, we described a "linear" approach for estimating a set of subspaces from data points $\mathbf{x}_j,\ j = 1, \ldots, k$ which lied on those subspaces. Essentially, proposed approach solves the normal vectors $\mathbf{b}_i,\ i = 1, \ldots, p$ from the set of nonlinear equations $\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x}_j)^p = 0,\ j = 1, \ldots, k$. From an optimization point of view, proposed algorithm gives a "linear" solution to the nonlinear least squares problem



(a) (b)

FIGURE 5.11: GPCA and estimated subspaces: (a) linear subspaces and (b) non-linear ones, where each data are mapped to an order-2 space.

$$\min_{\lambda_1,\ldots,\lambda_r,\boldsymbol{\Theta}} e^2 = \min_{\lambda_1,\ldots,\lambda_r,\boldsymbol{\Theta}} \sum_{j=1}^{k} \left(\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \mathbf{x}_j)^p \right)^2 . \tag{5.69}$$

where a solution is $\mathbb{S}^{n-1}$ unit sphere in $\mathbb{R}^n$.

In this section, we derive derive an optimal algorithm for subspaces reconstruction for the case when the data points are corrupted with i.i.d. zero-mean Gaussian noise. We show that the optimal solution can be obtained by minimizing a function similar to (5.69) in manner of fixed-point iteration. Since our approach is based on tensor representation proposed algorithm depends on Givens rotational angles and involves final rank-1 decomposition. Fixed-point manner provides us final smoothed data in form of linear subspaces.

Let $\mathbf{x}_j,\ j = 1,\ldots,k$ be the given collection of noisy data points. We would like to find a collection of subspaces $S_i,\ i = 1,\ldots,p$ such that the corresponding noise free data points $\tilde{\mathbf{x}}_j,\ j = 1,\ldots,k$ lie on those subspaces. It leads to solving of the constrained nonlinear least squares optimization problem

$$\begin{aligned} &\min \textstyle\sum_{j=1}^{k} \|\tilde{\mathbf{x}}_j - \mathbf{x}_j\|^2 \\ &\text{s.t.}\ \ \left(\textstyle\sum_{i=1}^{r} \lambda_i (\mathbf{u}_i^T \tilde{\mathbf{x}}_j)^p\right)^2 = 0,\ j = 1,\ldots,k. \end{aligned} \tag{5.70}$$

Using Lagrange multipliers γ_j for each constraint, the above optimization problem turns out to be equivalent to minimizing the Lagrangian function

$$\sum_{j=1}^{k}\left(\|\tilde{\mathbf{x}}_j-\mathbf{x}_j\|_2^2+\gamma_j\left(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p\right)^2\right). \tag{5.71}$$

After taking partial derivatives with respect to γ_j we obtain

$$\left(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p\right)^2=0, \tag{5.72}$$

this leads to the minimization problem of linear subspace estimate with absence of noise as in (5.66). So we have the system of linear equations (5.67) which can be solved as in (5.68).

Now we can take partial derivatives of (5.71) with respect to $\tilde{\mathbf{x}}_j$ so we obtain

$$2(\tilde{\mathbf{x}}_j-\mathbf{x}_j)+2p\gamma_j(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p)\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^{p-1}\mathbf{u}_i^T=0 \tag{5.73}$$

from which we can find a solution for γ_j as

$$\gamma_j=\frac{(\mathbf{x}_j-\tilde{\mathbf{x}}_j)}{p(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p)\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^{p-1}\mathbf{u}_i^T}= \frac{\sum_{i=1}^{r}\lambda_i[(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^{p-1}(\mathbf{u}_i^T\mathbf{x}_j)-(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p]}{p(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^p)\|\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_j)^{p-1}\mathbf{u}_i^T\|_2^2}. \tag{5.74}$$

Replacing obtained γ_j value in (5.73) gives us fixed-point algorithm

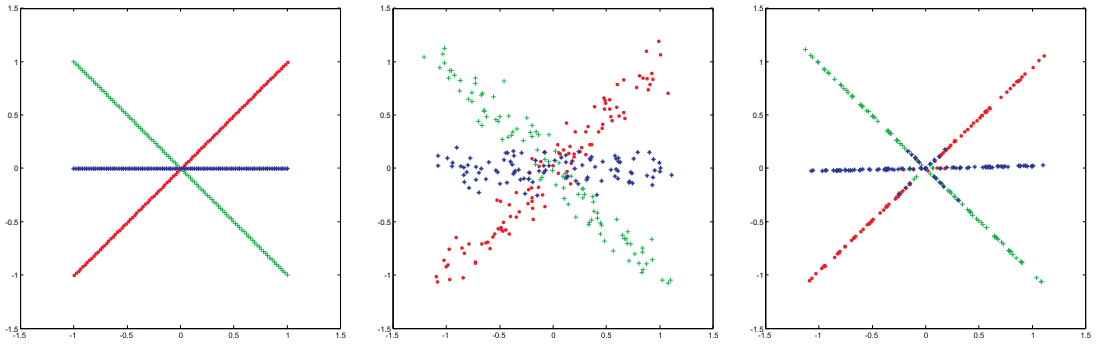
$$\tilde{\mathbf{x}}_{j,q+1}=\mathbf{x}_j-p\gamma_j(\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_{j,q})^p)\sum_{i=1}^{r}\lambda_i(\mathbf{u}_i^T\tilde{\mathbf{x}}_{j,q})^{p-1}\mathbf{u}_i^T, \tag{5.75}$$

where $q=[1,\ldots,\text{until converge}]$ and the value of γ_j is, in essence, the weight factor which, to reduce computational cost, can be replaced on some *exp* functions of the iterate number. Thus the objective function (5.71) is minimized as corresponding noise free data points $\tilde{\mathbf{x}}$ forms linear subspaces. The initial values for $\tilde{\mathbf{x}}$ can be equal to the noised data $\mathbf{x}$. To perform better convergence in (5.75) we can reassigned the nosied data set, $\mathbf{x}$, by the $\tilde{\mathbf{x}}$ after each iteration . Optimization can be implemented as in Algorithm 9.

Figure 5.12 shows an example of estimation of linear subspaces in presence of noise by an order-3 tensor. On Figure 5.12(a) we present initial data which consist of tree linear subspaces denoted by different colors and symbols, each subsapce consists of 100 data points, and on Figure 5.12(b) – noised data from (a) with noise level $\sigma = 0.1$. The result of estimate of linear subspaces is on Figure 5.12(c), note that in the intersection regions some data points converged to different subspaces that can happen due to noise and data intermixing and can not be eluminated without any prior information on data belonging. Overall we have rather good results of linear subspaces estimate based on proposed algorithm.

Algorithm 9 Generalized principal component analysis.

Initiate noise free data points $\tilde{\mathbf{x}}_j = \mathbf{x}_j,\ j = 1, \ldots, k$.
repeat
  Optimize Givens angles $\boldsymbol{\Theta}$ and find optimal λ^T for current states of $\tilde{\mathbf{x}}_j$.
  Evaluate Lagrange multipliers $\gamma_j,\ j = 1, \ldots, r$ via (5.74).
  Perform fixed-point iterations for $\tilde{\mathbf{x}}_j\ j = 1, \ldots, k$ as in (5.75).
  After each m^{th} itatere reassign data set, $\tilde{\mathbf{x}}$=$\mathbf{x}$.
until $(\sum_{j=1}^{k} < \mathbf{A}, \tilde{\mathbf{x}}_j >= 0)$
Estimate normal vectors $\mathbf{b}_i, i = 1, \ldots, p$.



(a) (b) (c)

FIGURE 5.12: GPCA and estimated subspaces in presence of noise: (a) initial linear subspaces, (b) noised linear subspaces from (a) with $\sigma = 0.1$, (c) estimated linear subspaces with an order-3 tensor.

5.4 Robust Nonlinear PCA

This section has been prepared based on the publications of Prof. Danijel Skocaj from The University of Ljubljana (Slovenia). Particularly, we refer the reader to his publications [92, 93] to learn more comprehensive explanations on Robust PCA.

Different methods have been proposed to perform PCA on a set of data. Major drawback of existing methods is that they are not robust, because they are based on the least squares minimization [94], and they are restricted to linear case. Robust methods, which have made the recognition stage less sensitive to outliers, are very important for real applications. In this Section we present a robust nonlinear PCA method for obtaining a consistent subspace representation in the presence of outlying data. The approach is based on tensor model and involve the robust expectation-maximisation (EM) algorithm for estimation of principal subspaces [92, 95].

5.4.1 Robust PCA Based on EM

We denote a n-dimensional data point by a vector $\mathbf{x}_i = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ and data set is then a matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \in \mathbb{R}^{n \times k}$. The eigenvectors obtained from $\mathbf{X}$ are denoted by m rows in matrix $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_k] \in \mathbb{R}^{m \times k}$, where each column $\mathbf{e}_i = [e_1, \ldots, e_m]^T \in \mathbb{R}^m$. We assume $\mathbf{X}$ as not necessary be normalized or zero-mean matrix, since the first row in $\mathbf{E}$ can be defined as a vector of ones, which can be used for data normalization. Generally, $m < n$ and we utilize a matrix of combinational coefficients $\mathbf{A} \in \mathbb{R}^{n \times m}$ to represent $\mathbf{x}_i$ with sufficient accuracy [92], where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]^T$ and vector $\mathbf{e}_i$:

$$\tilde{\mathbf{x}}_j = \mathbf{A}\mathbf{e}_j, \ j = 1, \ldots, k. \tag{5.76}$$

where $\tilde{\mathbf{x}}$ denotes the approximation of $\mathbf{x}$. The entire data set $\mathbf{X}$ can thus be represented as $\tilde{\mathbf{X}} = \mathbf{A}\mathbf{E}$ [92]. In the case of tensor representation we can rewrite above equitation as

$$\tilde{x}_{ij} = \mathbf{a}_i^T \mathbf{e}_j = < \mathbf{a}_i, \mathbf{e}_j >, \ i = 1, \ldots, n, \ j = 1, \ldots, k. \tag{5.77}$$

So in the general case of an order-p tensor representation robust PCA becomes nonlinear:

$$\tilde{x}_{ij} =< \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >= \sum_{l=1}^{r} \lambda_l^{(i)} (\mathbf{e}_j^T \mathbf{u}_l^{(i)})^p, \ i = 1, \ldots, n, \ j = 1, \ldots, k. \tag{5.78}$$

which means that for each dimensionality of the data set we utilize different rank-r order-p tensors $\mathbf{A}^{(i)}, \ i = 1, \ldots, n$. In this case the linear PCA becomes the marginal case when $p = 1$. Now we will present an algorithm based on EM and Probabilistic PCA (PPCA), which can operate with missing data. On the basis of this algorithm we will then present a robust nonlinear PCA approach.

5.4.2 EM Algorithm for PCA

Most of the algorithms for building principal subspaces are based on the eigendecomposition of the covariance matrix of the input data [92]. However, there exist other approaches. For instance, a probabilistic approach, where PCA can be considered as a limiting case of a linear Gaussian model, when the noise is small with stable characteristics. From these definitions we can derive an algorithm for calculating principal axes, which is based on the EM algorithm. This algorithm consists of two steps, E and M, which are sequentially and iteratively executed [93]:

- **E-step:** Estimate coefficients $\mathbf{A}$ using computed principal axes $\mathbf{E}$.
- **M-step:** Compute new principal axes $\mathbf{E}$ which maximize expected joint likelihood of the estimated coefficients $\mathbf{A}$ and the observed data $\mathbf{X}$.

The EM algorithm for PCA proposed in [96] is given by:

- **E-step**: $\mathbf{A} = \mathbf{X}\mathbf{E}^T(\mathbf{E}\mathbf{E}^T)^{-1}$.
- **M-step**: $\mathbf{E} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}\mathbf{X}$.

The rows of $\mathbf{A}$ span the space of the first m principal axes. These vectors are, in general, not orthogonal, but they can be orthogonalized at the end of EM recursion [92]. Described algorithm does not allow us to use nonlinearity. Both matrices are evaluated in whole.

The convergence of the algorithm can be checked by looking at a difference in the successive estimates of the average lost variance σ^2 per discarded dimension. As shown in [97], the error for the maximum likelihood solution is $\sigma^2 = \frac{1}{k-m}\sum_{j=m+1}^{k}\lambda_j$, that corresponds to the sum of the discarded eigenvalues. It can be conveniently calculated by [92]

$$\sigma^2 = \frac{1}{k-m}(VAR(\mathbf{X}) - \sum_{j=1}^{m}\lambda_j), \tag{5.79}$$

where $VAR(\mathbf{X})$ is the variance of $\mathbf{X}$ which is defined as the sum of variances in rows of $\mathbf{X}$. Since in the EM algorithm we do not explicitly calculate eigenvalues at each iteration, we can estimate σ^2 by [92]

$$\sigma^2 = \frac{1}{k-m}(VAR(\mathbf{X}) - VAR(\tilde{\mathbf{X}}). \tag{5.80}$$

In [96] it was proposed a generalized E-step for handling missing data. One can treat missing data as additional hidden states and estimate them in the E-step simultaneously with estimating the coefficients by solving a least squares problem [92]. In the following explanation we describe an algorithm for handling missing data in the EM approach, which involves tensor model.

5.4.3 PCA in the Presence of Missing Data

It is important to note that in the case when we use all data points, the M-step is equivalent to calculating the coefficients by the solution of the nonlinear system of equations (5.76). This can be easily seen by noting that $\mathbf{e}_i = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}\mathbf{x}_i$. The least squares solution of (5.76) is equivalent to the pseudo-inverse. A very similar observation is held also for the M-step. Therefore, we can perform the EM-algorithm by iteratively solving the following systems of nonlinear equations [93]:

- **E-step:** Estimate coefficients in $\mathbf{A}^{(i)},\ i = 1,\ldots,n$ in the following way: For each data point $\mathbf{x}_j,\ j = 1,\ldots,k$, solve the following system of nonlinear equations in the least squares sense:

$$x_{ij} =< \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >,\ i = 1,\ldots,n. \tag{5.81}$$

- **M-step:** Estimate principal axes in $\mathbf{E}$ in the following way: For each data point $\mathbf{x} = [x_1,\ldots,x_n]^T$, solve the following system of nonlinear equations in

the least squares sense:

$$x_{ij} =< \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >,\ j = 1, \ldots, k. \tag{5.82}$$

From these considerations we can also see how to compute the coefficients and principal axes in the case of missing data. We only need to set up these equations with the known data points and to compute the coefficients in $\mathbf{a}_i$ and $\mathbf{e}_i$. Similar solutions, but just for linear model was proposed in [92].

When dealing with data containing a considerable amount of missing data points, such a formulation results is an ill-posed problem. To alleviate this problem we impose additional application dependent constraints to the minimization process. When the data are ordered , e.g, image sequences, we can extend the algorithm to include also a smoothness prior to enforce that the missing data are changing smoothly over time. Thus, in the M-step we minimize the second derivative of the reconstructed missing data points. If $\mathbf{S}$ is a set containing all missing points, the algorithm looks as follows [92]:

- **E-step:** Estimate coefficients in $\mathbf{A}^{(i)},\ i = 1, \ldots, n$ in the following way: For each data point $\mathbf{x}_j,\ j = 1, \ldots, k$, solve the following system of nonlinear equations in the least squares sense:

$$x_{ij} =< \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >,\ i = 1, \ldots, n,\ \text{where } x_{ij} \notin \mathbf{S}. \tag{5.83}$$

- **M-step:** Estimate principal axes in E in the following way: For each data point $\mathbf{x} = [x_1, \ldots, x_n]^T$, solve the following system of nonlinear equations in the least squares sense:

$$\begin{aligned} x_{ij} &=< \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >,\ j = 1, \ldots, k,\ \text{where } x_{ij} \notin \mathbf{S}, \\ 0 &=< \mathbf{A}^{(i)}, (\mathbf{e}_j^{\circ p})'' >,\ j = 1, \ldots, k,\ \text{where } x_{ij} \in \mathbf{S}, \end{aligned} \tag{5.84}$$

  where α is the parameter which weights the influence of the smoothness constraint.

The overall algorithm minimizes the following error function:

$$\varepsilon = \sum_{j=1}^{k} \sum_{i \in \mathbf{G}_j} (x_{ij} - < \mathbf{A}^{(i)}, \mathbf{e}_j^{\circ p} >)^2 + \alpha \sum_{j=1}^{k} \sum_{i \in \mathbf{B}_j} (< \mathbf{A}^{(i)}, (\mathbf{e}_j^{\circ p})'' >)^2, \tag{5.85}$$

where $\mathbf{G}_j$ denotes a set of indices of non-missing data points in j^{th} data layer, while B_j denotes a set of indices of the corresponding missing data points.

Alternatively, the principal axes can in the M-step also be obtained by applying the standard PCA using all data points [92]. It provides that missing data points are filled-in. The question is how to optimally fill-in the values of the missing data points. Since not all points of a data set are known, some coordinates of the corresponding point in the data space are undefined. Thus, the position of the point is constrained to the subspace defined by the missing points. Let the principal subspace $\mathbf{E}$, which models the input data space, is given. We can determine the optimal location of a missing data point as a weighted value of principal subspaces. This data point is obtained on the basis of the reconstructed values by replacing missing data points. Missing data points are calculated by (5.76) using the coefficients $\mathbf{e}_i$, which were estimated in E-step of the current iteration, and the principal axes $\mathbf{a}_i$ obtained in the previous iteration [92].

- **M-step**: Estimate principal axes in $\mathbf{A}$ by applying the standard PCA on $\mathbf{X}$ with the reconstructed missing data points:

$$x_{ij} = \tilde{x}_{ij},\ i = 1, \ldots, n,\ j = 1, \ldots, k,\ x_{ij} \in \mathbf{M} \text{ where } \tilde{\mathbf{X}} \text{ as in (5.78).} \quad (5.86)$$

What still remains to be determined is how to calculate initial solutions for $\mathbf{A}^{(i)},\ i = 1, \ldots, n$. In fact, for the general case of tensor, we just start with random values. For the linear model, $p = 1$, we can calculate the principal axes $\mathbf{A}^{(i)}$ from an estimate of $\mathbf{E}$ obtained by performing SVD on the inner product matrix $\mathbf{S}$. The matrix $\mathbf{S}$ is evaluated from the non-missing points of the input set $\mathbf{X}$:

$$s_{ij} = \frac{1}{|L|} \sum_{l \in L} x_{li} x_{lj};\ L = \{l | x_{li} \notin \mathbf{M},\ x_{lj} \notin \mathbf{M}\}. \quad (5.87)$$

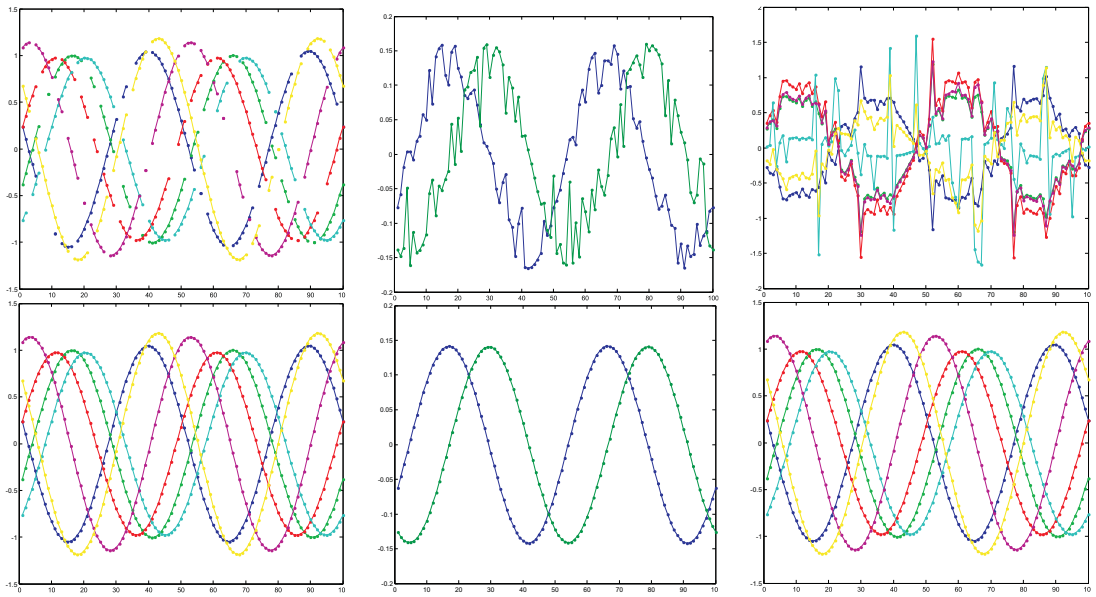
To summarize, the nonlinear algorithm for missing data points is [92]:

Let's consider a set of 1-dimensional vectors which is formed of 6 shifted harmonic (sinus) functions with different magnitudes [92]. We randomly removed 20% of the elements, as depicted in the top row of Figure 5.13(a). Now, the goal is to find the optimal principal axes representing these vectors which contain missing data by applying Algorithm 10. In this experiment, the estimation of the principal axes in the M-step of the EM algorithm was performed by applying the standard

Algorithm 10 Restoration of the missing data points in NLPCA.

Generate randomly symmetric tensors $\mathbf{A}^{(i)},\ i = 1, \ldots, n$.
Perform M-step according to (5.84) obtaining an estimate for $\mathbf{E}$.
repeat
  Perform E-step according to (5.83).
  Perform M-step according to (5.84).
  Replace missing data points in $\mathbf{X}$ by reconstruction using (5.78).
  Calculate σ^2.
until $\sigma^2 \to 0$

PCA on the input data with the reconstructed missing data points. Figure 5.13 illustrates the performance of the proposed method. First, the initial values of the principal axes are obtained from the covariance matrix estimated from the non-missing values of the input vectors. Then, based on the E-step, we calculate the coefficients and the reconstructed signals. The missing data points are then replaced by their reconstructed values from the previous iteration. The initialization, the last, stable, iteration of the algorithm are shown in the first and second rows, respectively. We can note how the functions representing estimated principal axes are getting smoother after every iterations and obtain a perfect reconstruction of the input vectors [92].



(a) (b) (c)

FIGURE 5.13: A 1-D example with missing data points: (a) input data at each iteration, (b) estimated principal axes, (c) reconstructed vectors.

5.4.4 Robust PCA

The basic idea of our robust PCA algorithm is to determine the outliers and treat them as missing data points, and then use the algorithm described above to calculate PCA without these outliers [92]. Being based on the estimated principal axes and coefficients we can again determine the outliers and repeat the process. The crucial question is how to determine the outliers. We describe two approaches, one of which gives a global threshold and another gives a local threshold [92]:

- **Global threshold:** Subject to the number of principal components which we choose to represent the data, m, we expect that an average error is equal to the discarded variance, which is $\sigma^2 = \frac{1}{n}\sum_{i=m+1}^{n}\lambda_i$. If we divide obtained value σ^2 on the number of data points, k, we can get an average expected error per data point, $\sigma^2_{point} = \sigma^2/k$. Now we can treat all those data points as outliers whose reconstruction errors are larger than biased value of σ^2_{point}. The problem with this method consists in assuming the same variability across the data set, which is in general not true.
- **Local threshold:** In [94] it was proposed to compute a local threshold for each point based on the Median Absolute Deviation (MAD). They compute σ^2 for each point as a median value over region with center in the data point.

According to this we can outline our robust nonlinear PCA algorithm as follows [92]: Usually, for leaner data and model only a few iterations of this algorithm are

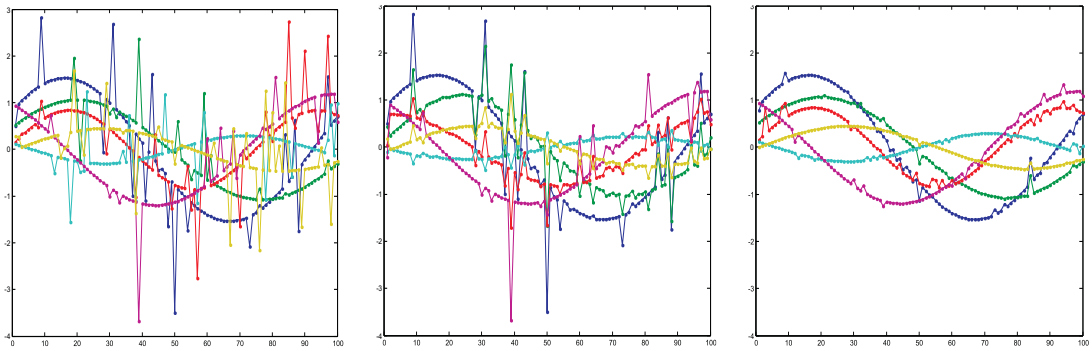
Algorithm 11 Robust nonlinear PCA.

repeat
Compute principal axes and coefficients using robust nonlinear PCA on $\mathbf{X}$.
Detect outliers using either global or local threshold.
Treat outliers as missing data points and perform NLPCA by using Algorithm 10.
Replace missing data points in $\mathbf{X}$ by reconstruction using estimated $\mathbf{E}$ and $\mathbf{A}^{(i)}$, $i = 1, \ldots, n$.
until the outlier set is empty

sufficient for convergence.

Figure 5.14 illustrates the performance of the algorithm robunst nonlinear PCA. Now, a set of 1-D vectors is formed from 6 shifted sinus functions with different magnitudes and random frequencies. We randomly replaced 20% of the elements

with Gaussian noise, Figure 5.14(a). Figure 5.14(b) and (c) depict the reconstruction after the using standard PCA and proposed robust nonlinear PCA algorithm of treating noised data, where $p = 2$. One can observe, how initially noisy signals become more regular. We need to note that due to nonlinearity of the data, the case of differen frequencies, it is impossible to reconstruct pure signal by standard PCA because it is leaner operator.



(a) (b) (c)

FIGURE 5.14: A 1-D example with noisy signals: (a) input data, (b) reconstruction after using the standard PCA algorithm, (c) reconstruction after the using tensor robust PCA algorithm.

5.5 Tensor Methods for Systems of Nonlinear Equations

This section has been prepared based on the publications of Prof. Brett Bader from The Sandia National Laboratories (CA\USA). Particularly, we refer the reader to his publication [98] to learn more comprehensive explanations on solving large-scale systems of nonlinear equations.

In this section we describe iterative tensor methods for solving systems of nonlinear equations. Systems of nonlinear equations arise in many practical tasks, e.g., solution of partial differential or non-polynomial equations. Tensor approach is based on higher-order Taylor approximation of the system of nonlinear equation, that is required in each tensor iteration, while Newton's method is the first order approximation [98]. Newton's method is a state-of-the-art approach which is widely used in practice but is weak for large-scale problems because of its high linear algebra costs and large memory requirements. The diagram on Figure 5.15 shows classification of solutions for system of nonlinear equations [99].

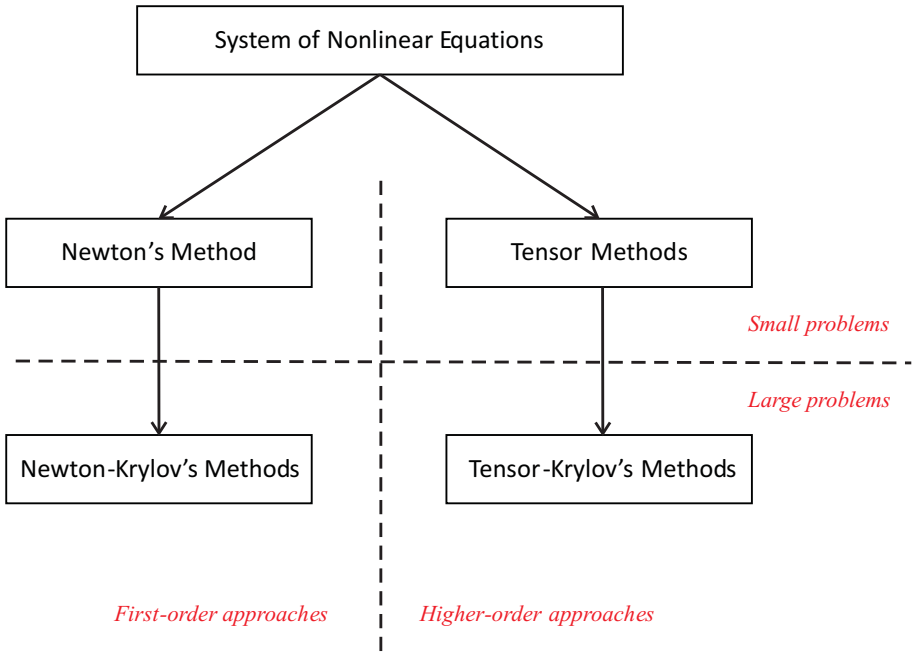


FIGURE 5.15: Existing optimization methods for systems of nonlinear equations solving.

Direct tensor methods for nonlinear equations have performed especially well on small, dense problems, for instance, when the Jacobian matrix at the solution is

singular or ill-conditioned, which may occur when the current solution approaches to optimal value. For large space problems we can utilize modified method, called Newton-Krylov's method, which involves approximate solution for a local linear model, or tensor-Krylov method [98]. Here we concentrate on small space problem and compare algorithms of Newton's and tensor models as well as their performances. The test results evidently show that that tensor methods are generally more robust and efficient than Newton's one for some classical optimization problems.

To acquaint with well discussed tensor methods for large nonlinear equations we refer the reader to [100], where author uses direct methods for solving the linear systems of equations that arise in each tensor iteration. Unfortunately, for very large systems of nonlinear equations, the slow asymptotic performance and large storage requirements of direct methods make them impractical. For these systems robust and fast iterative solvers such as Krylov's algorithms must be used. Different tensor-Krylov's approaches are presented and analyzed in [98]. One of the main problems of tensor methods is that the Jacobian matrix must be available explicitly in each iteration. Tensor-Krylov's methods do not almost require matrix storage. It is one of the main advantages of these methods.

Now we can define the system of nonlinear equations as [98]

$$\text{for } \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n, \text{ find } \mathbf{x}_* \in \mathbb{R}^n \text{ such that } \mathbf{f}(\mathbf{x}_*) = 0, \tag{5.88}$$

where $\mathbf{f}(\mathbf{x})$ is at least once continuously differentiable.

Below we present and compare Newton's and tensor approach for the problem of a system of nonlinear equations.

5.5.1 Newton's method

Standard methods in numerical optimization are called such methods, for which solutions of (5.88) are based on linear local models [98]. Most popular among these methods is Newton's one, which bases each of its iterations on a linear local model $\mathbf{m}(\mathbf{x}_c + \mathbf{d})$ of the function $\mathbf{f}(\mathbf{x})$ around the current state $\mathbf{x}_c$ as [98]:

$$\mathbf{m}(\mathbf{x}_c + \mathbf{d}) = \mathbf{f}(\mathbf{x}_c) + \mathbf{J}(\mathbf{x}_c)\mathbf{d}, \tag{5.89}$$

where $\mathbf{x}_c$ means the current iterate of a nonlinear solver, $\mathbf{d} \in \mathbb{R}^n$ is the step and $\mathbf{J}(\mathbf{x}_c) \in \mathbb{R}^{n\times n}$ is either the current Jacobian matrix of $\mathbf{f}(\mathbf{x})$ or its approximation. A root of this local model provides the Newton step [98]

$$\mathbf{d} = -\mathbf{J}^{-1}(\mathbf{x}_c)\mathbf{f}(\mathbf{x}_c), \tag{5.90}$$

which is used to reach the next iterate point. Thus, Newton's method is defined when $\mathbf{J}(\mathbf{x}_c)$ is nonsingular and consists of updating the current point with the Newton step

$$\mathbf{x}_* = \mathbf{x}_c + \mathbf{d}. \tag{5.91}$$

In practice, for the case of large-scale problems, it is not suitable to implement the Newton'sĄmethod using direct factorizations of $J(\mathbf{x}_k)$ due to large arithmetic and storage costs [98].

5.5.2 Tensor method

Tensor methods solve (5.88) by including more information in local model. According to this generated steps are of better quality than for Newton's method and reaching the solution is faster [98]. Especially it happens for problems where the Jacobian is singular or ill-conditioned at its solution. Common local tensor model bases each iteration on a quadratic model of the nonlinear function [98],

$$\mathbf{m}(\mathbf{x}_c + \mathbf{d}) = \mathbf{f}(\mathbf{x}_c) + \mathbf{J}(\mathbf{x}_c)\mathbf{d} + \frac{1}{2}\mathbf{T}_c \times_2 \mathbf{d} \times_3 \mathbf{d}, \tag{5.92}$$

where $\mathbf{T}_c \in \mathbb{R}^{n\times n\times n}$ is an order-3 tensor (that is why these methods get their name), which includes second-order information of $\mathbf{f}(\mathbf{x}_c)$. Usually, $\mathbf{T}_c$ is selected so that the model interpolates r previous function values in the recent history of iterates, which makes $\mathbf{T}_c$ a rank-r tensor. Thus no second derivative information is evaluated in forming the tensor term $\mathbf{T}_c$. In practice, for most approaches r is equal to 1 or 2, but, as was shown in our tensor analysis, to describe any system in whole we need to involve full rank tensor. From here and below, for simplicity, we denote the current state $\mathbf{f}(\mathbf{x}_c)$ as $\mathbf{f}$ and $\mathbf{J}(\mathbf{x}_c)$ as $\mathbf{J}$.

Let's define the tensor term $\mathbf{T}_c$ by such a way that the local model $\mathbf{m}(\mathbf{x}_c + \mathbf{d})$ interpolates values of the function $\mathbf{f}(\mathbf{x})$ at past iterates [98]. The model should

satisfy [98]

$$\mathbf{f}(\mathbf{x}_{-k}) = \mathbf{f} + \mathbf{J}\mathbf{s}_k + \frac{1}{2}\mathbf{T}_c \times_2 \mathbf{s}_k \times_3 \mathbf{s}_k,\ k = 1, \ldots, r, \tag{5.93}$$

where

$$\mathbf{s}_k = \mathbf{x}_{-k} - \mathbf{x}_c,\ k = 1, \ldots, r. \tag{5.94}$$

Generally, it is required that the past points $\mathbf{x}_{-1}, \ldots, \mathbf{x}_{-r}$ should be selected so that the set of directions $\mathbf{s}_k$ from $\mathbf{x}_c$ to the selected are linearly independent [98]. That is why in practice r is small. The procedure of finding linearly independent directions is easily implemented easily using Gram-Schmidt algorithm. After selecting the linearly independent past directions, we form the tensor term. In [101] authors define $\mathbf{T}_c$ as a tensor that satisfies the interpolation conditions [98]:

$$\begin{gathered} \min_{\mathbf{T}_c} ||T_c||_F, \\ \text{subject to } \mathbf{T}_c \times_2 \mathbf{s}_k \times_3 \mathbf{s}_k = 2(\mathbf{f}(\mathbf{x}_{-k}) - \mathbf{f} - \mathbf{J}\mathbf{s}_k). \end{gathered} \tag{5.95}$$

The solution of above equation is the sum of r rank-1 tensors [98] whose horizontal faces are symmetric,

$$\mathbf{T}_c = \sum_{k=1}^{r} \mathbf{a}_k \circ \mathbf{s}_k \circ \mathbf{s}_k, \tag{5.96}$$

where $\mathbf{a}_k$ is the k^{th} column of matrix $\mathbf{A} \in R^{n \times p}$ and defined by $\mathbf{A} = \mathbf{Z}\mathbf{M}^{-1}$, $\mathbf{Z}$ is an $(n \times p)$-dimensional matrix whose columns are $\mathbf{z}_j = 2(\mathbf{f}(\mathbf{x}_{-j}) - \mathbf{f} - \mathbf{J}\mathbf{s}_j)$, and $\mathbf{M}$ is a $(p \times p)$-dimensional matrix entries of which defined by $m_{i,j} = (\mathbf{s}_i^T \mathbf{s}_j)^2,\ 1 \leq i, j \leq p$.

Using the tensor term derived above, the tensor model (5.92) becomes [98]

$$\mathbf{m}(\mathbf{x}_c + \mathbf{d}) = \mathbf{f} + \mathbf{J}\mathbf{d} + \frac{1}{2}\sum_{k=1}^{r} \mathbf{a}_k (\mathbf{d}^T \mathbf{s}_k)^2. \tag{5.97}$$

The simple form of the second term in (5.97) is the key of being able to efficiently solve the tensor model. The cost of forming the tensor term and the tensor model is $O(n^2 r)$ arithmetic operations.

As was shown in [100] the solution of (5.97), where $\mathbf{m}(\mathbf{x}_c + \mathbf{d}) = 0$, can be reduced to the solution of a system of r quadratic equations in r unknowns, plus the solution of $r + 1$ systems of linear equations that all involve the same matrix, $\mathbf{J}$ possibly augmented by r dense rows and columns. We can utilize the same approach to solve the minimization problem $\min \|\mathbf{m}(\mathbf{x}_c + \mathbf{d})\|_2$.

The basic approach used in these algorithms is illustrated by the case when $\mathbf{J}$ is nonsingular and the tensor model has a root. In this case, premultiplying the tensor model by $\mathbf{s}_i^T\mathbf{J}^{-1}, i = 1,\ldots,r$, gives the r quadratic equations in the r unknowns $\beta_i = \mathbf{s}_i^T\mathbf{d}$,

$$\mathbf{s}_i^T\mathbf{J}^{-1}\mathbf{f} + \beta_i + \frac{1}{2}\sum_{k=1}^{r}\mathbf{s}_i^T\mathbf{J}^{-1}\mathbf{a}_k\beta_k^2 = 0, i = 1,\ldots,r. \tag{5.98}$$

These can be solved for $\beta_i,\ i = 1,\ldots,r$, and then the equation

$$\mathbf{f} + \mathbf{J}\mathbf{d} + \frac{1}{2}\sum_{k=1}^{r}\mathbf{a}_k\beta_k^2 = 0 \tag{5.99}$$

can be solved for $\mathbf{d}$. This process requires the calculation of $\mathbf{J}^{-1}\mathbf{F}$ and $\mathbf{J}^{-1}\mathbf{a}_k,\ k = 1,\ldots,r$ (or alternatively $\mathbf{J}^{-1}(\mathbf{F} + \frac{1}{2}\sum_{k=1}^{r}\mathbf{a}_k\beta_k^2)$ and $\mathbf{J}^{-T}\mathbf{s}_i,\ i = 1,\ldots,r$) and the solution of the system of quadratic equations (5.98). We need to note that in general case of $\mathbf{f}(\mathbf{x})$ it can be impossible to get explicit value of Jacobian matrix, $\mathbf{J}(\mathbf{x})$, from system of nonlinear equations. So we have to approximate $\mathbf{J}(\mathbf{x})$ as well. To solve the generalized minimization problem, $\min\|\mathbf{Q}\mathbf{m}(x_c + d)\|_2$, we refer the reader to [100].

Now let's focus on the case when $r = 1$ so our approach transforms to one secant update. Similar idea is used for large-scale tensor-Krylov methods [98]. In this case, the tensor model about some data point $\mathbf{x}_k$ reduces to

$$\mathbf{m}(\mathbf{x}_k + \mathbf{d}) = \mathbf{f}(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)\mathbf{d} + \frac{1}{2}\mathbf{a}_k(\mathbf{s}_k^T\mathbf{d})^2, \tag{5.100}$$

where

$$\mathbf{a}_k = \frac{2(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{f}(\mathbf{x}_k) - \mathbf{J}(\mathbf{x}_k)\mathbf{s}_k)}{(\mathbf{s}_k^T\mathbf{s}_k)^2} \text{ and } \mathbf{s}_k = \mathbf{x}_{k-1} - \mathbf{x}_k. \tag{5.101}$$

After forming the model, we use it to determine the step to the next trial point. Because (5.100) may not have a root, one can solve the minimization subproblem $min\|\mathbf{m}(\mathbf{x}_k + \mathbf{d})\|_2$ and a root or minimizer of the model using the tensor step. Due to the special form of (5.100), the solution of subproblem in the nonsingular case reduces to solving a quadratic equation followed by solving a system of $n-1$ linear equations with $n-1$ unknowns. Based on described material, we can outline rank-1 tensor method as follows [98]:

Algorithm 12 Tensor algorithm.

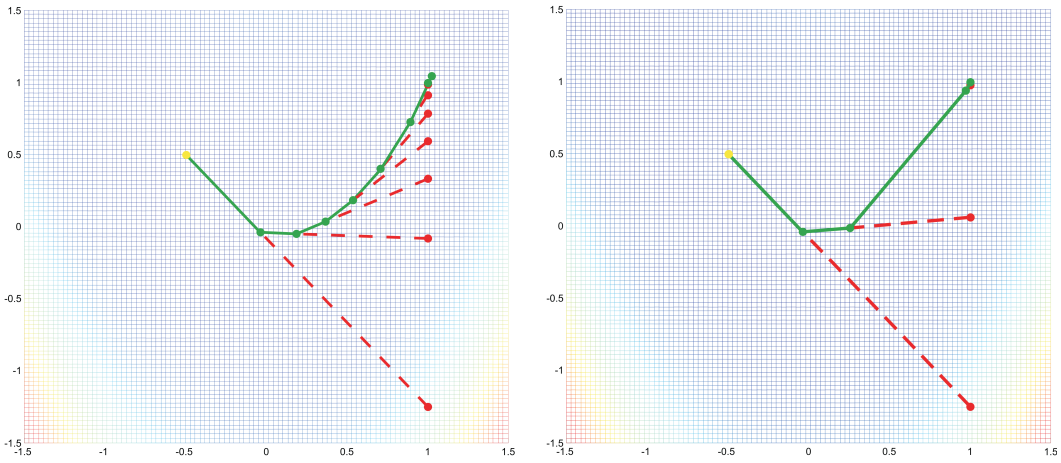
Choose initial state $\mathbf{x}_0$.
On the basis Newton' algorithm evaluate the next state, $\mathbf{x}_k, k = 1$.
repeat
  Form local tensor model as in 5.100 for $\mathbf{x}_k$.
  Find $\mathbf{d}$ that minimize $\mathbf{m}(\mathbf{x}_k + \mathbf{d})$.
  Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}$.
  If $\mathbf{x}_{k+1}$ is not acceptable, then perform linesearch $\min \|\mathbf{f}(\mathbf{x}_k + \alpha\mathbf{d})\|_2$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}$.
until the x_{k+1} is stable

5.5.3 Numerical Testing

Here we present results of testing Newton's and tensor models for minimization of modified Rosenbrock's function:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}) = 5(x_2 - x_1^2) \\ f_2(\mathbf{x}) = 1 - x_1 \end{cases} \tag{5.102}$$

Modification was performed just for squeezing the range of the function in observing area. The modified function has the same root, $(1, 1)$, as for classical definition. Figure 5.16 presents optimization results for described above (a) Newton' and (b) tensor methods, respectively [99]. To show and compare performance of the algorithm we used the same initial data point (denoted by yellow dot), $(-0.5, 0.5)$, for both optimization methods. Both subfigures have the same meaning of colored lines: red dashed lines denote estimated position of the next optimal points, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}$, and green lines denote accepted locations after linesearch procedure, $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}$. As we can see from the obtained results the tensor method needs almost 3 steps to reach the minimum whereas Newthon's method takes 8 steps and performs fluctuation around the root point due to ill-condition of Jacobian matrix near by the root. Is is evident that due to nonlinearity of tensor method we can reach functional minimum faster and more reliable than in the case of linear methods utilization.



(a) (b)

FIGURE 5.16: Minimization of modified Rosenbrock's function: (a) Newton's method, (b) tensor method.

Chapter 6

Conclusions and Future Work

This dissertation contributes to the field of tensor analysis which appears increasingly in many application domains such as signal and image processing, factor analysis, computer vision, etc., where we need to decompose higher-order data. A tensor is a multidimensional order-p array of scalars, where p denotes the number of non-singleton dimensions. Attractiveness of tensors consists in their "native" form of data representation, e.g., higher-order moments and derivatives. During the last half a century tensors have played arisingly important role in data analysis. And again I would like to acknowledge Prof. Tamara Kolda and Prof. Brett Bader from Sandia National Laboratories as well as other authors whose publications I have used in this dissertation.

6.1 Conclusions

At the beginning of this dissertation we have described importance of tensor approach and presented an overview of the state-of-the-art research works as well as notations and operations in the field of tensor analysis. After, we have analyzed existing methods of tensor decomposition and their applications.

In this dissertation we have defined a sum-of-rank-1 non-redundant tensor decomposition in terms of which we have determined an upper bound for the rank of symmetric and non-symmetric tensors and described tensors as sets of variables. The decomposition we have proposed reinterprets the orthogonality property of the singular vectors of matrices as a geometric constraint on the rank-1 matrix bases

which leads to a geometrically constrained singular vector frame. The main idea of the proposed decomposition consists in preservation of cardinality between initial data set and obtained result. As it was shown, the simplest case of tensor decomposition which preserves cardinality is SVD ($p = 2$), in this case SVD just transforms data space into parametric one.

We have figured out that widely known and used tensor models, CP and Tucker, can be considered as extensions of the SVD without preserving cardinality. This means that the number of variables, which describes the decomposition results, in higher than d.o.f. of initial data set. So decomposition like that is considered to be redundant. It happens because the algorithms of CP and Tucker models are based on properties of SVD but do not involve them in whole. We have presented a tensor decomposition that includes SVD as a particular case with all its corresponding properties. It has been numerically proved that proposed tensor model does not have redundancy.

We have presented a geometrically constrained basis vector frame that yields a set of rank-1 symmetric tensor bases that is able to attain a sum-of-rank-1 decomposition of any order, any dimensional symmetric tensor. We have also defined permutation tensors on the basis of which we have determined how to pick vectors for non-symmetric rank-1 tensor bases from vector bases for symmetric ones. The number of non-zero entries of such permutation tensors are determined by the upper-bound of rank for tensors. We have provided a differential model of non-redundant decomposition and utilize a Levenberg - Marquardt algorithm to achieve zero-error decompositions by optimizing the Givens rotation angles. This model has been successfully applied both to classical SVD-based tasks and to high-order extension of linear problems.

We have applied the proposed tensor decomposition model to the problem of density estimation as well as to derivative problems of principal curve fitting and clustering. Obtained results are better and easier computable than for moment based methods. We have shown how tensors can be utilized in tasks of data subspace separation via Generalized PCA where each factor describes separated data set. Robust Nonlinear PCA had been described in tensor terms and used for missing and noised data that have allowed us to had a deal with nonlinear data space. Solution of system of nonlinear equation via tensor methods has shown much better performance comparing with classical Newton's method. Given results provide us wide field for further research.

6.2 Future Work

In this dissertation we have been interested in the basis of non-redundant tensor model. Proposed conceptions were applied to the state-of-the-art problems in the fields of data analysis. As further important and possible research we offer the following goal list:

1. Investigate properties of the proposed geometrical vector frame and prove the invariance and uniqueness of the offered approach which can be empirically observed for small values of tensor order and the dimensionality. The most important part for such a problem is an analytic solution and simple constructing algorithm of vector frame. Because proposed vector frame always covers the unit hypersphere by even number of points (vectors from origin) it is also important to figure out an approach for odd number of points.

2. Analyze properties of permutation tensors which influence on the final structure of decomposed tensor. Like for the case of permutation matrices, we believe that there is a common transformation rule for different permutation tensors that converts linear combinational coefficients from ones to others. Strictly speaking the invariance of proposed tensor model depends on both invariance of vector frame and properties of permutation tensor.

3. Simplify implementation and improve the convergence speed of decomposition model. Due to non-orthogonality of vector frame in general case, we have to figure out all combinational coefficients simultaneously in order to decompose a tensor. A possible approach to get them separately, that means tensor orthogonalization, is to extend dimensions of decomposed tensor, like for the case of KPCA, and perform decomposition in higher dimensions.

4. Expand analysis and application of the non-negative case of non-redundant tensor decomposition, which describes natural form of data representation and does not involve nullifying operations. Such an approach is very important in the fields of image processing and computer vision.

5. Decrease complexity and apply the proposed tensor model to the problems of density estimation/classification/clustering/dimensionality reduction/compression/independent components analysis/etc. with regard to real higher-order multidimensional data. Due to complexity of tensor model, since it can be imagined as multidimensional higher-order polynomial, the most weak point

of the current tensor models is their applicability just to small data set. There are plenty of published papers in this field without real numerical results.

6. Reveal the definition, properties, and analytical expression of tensor determinant for higher-order cubical tensors. Our earlier research let us believe in existing of such an operator. We are sure that tensor determinant will play the same role in multidimensional higher-order analysis as matrix determinant does it in the case of linear systems.

7. When all previous challenges are solved we can consider the problem of tensor inversion and methodology, however it sounds unbelievable, for analytic solution of systems of nonlinear polynomial equations.

Appendix A

Hyperspherical Coordinate System

The vector frame for tensor decomposition can be easily reparameterized in hypersphere coordinate system since vectors form uniformly distributed points on the unit hypersphere. Hyperspherical coordinate system in n-dimensional Euclidean space expresses position of point $\mathbf{x}$ by $(n-1)$ angular hyperspherical coordinates $\theta_i,\ i = 1, \ldots, n-1$ and the distance from the origin ρ, which can be thought of as the radios-vector the hypersphere centred at the origin [102]. Polar and spherical coordinate systems are particular cases of hyperspherical one for 2 and 3-dimensional Euclidian spaces, respectively.

The transformation from Cartesian to hyperspherical coordinates for $\mathbf{x}$ and θ with ρ is:

$$\begin{aligned} x_1 &= \rho cos(\theta_1) \\ x_2 &= \rho sin(\theta_1) cos(\theta_2) \\ x_3 &= \rho sin(\theta_1) sin(\theta_2) cos(\theta_3) \\ &\vdots \\ x_{n-1} &= \rho sin(\theta_1) \cdots sin(\theta_{n-2}) cos(\theta_{n-1}) \\ x_n &= \rho sin(\theta_1) \cdots sin(\theta_{n-2}) sin(\theta_{n-1}) \end{aligned} \tag{A.1}$$

The transformation from hyperspherical to Cartesian coordinates for θ with ρ and $\mathbf{x}$ is:

$$\begin{aligned} \theta_1 &= \arctan \tfrac{x_2}{x_1} \\ \theta_2 &= \arctan \tfrac{x_3}{\sqrt{x_1^2+x_2^2}} \\ &\vdots \\ \theta_{n-1} &= \arctan \tfrac{x_n}{\sqrt{\sum_{i=1}^{n-1} x_i^2}} \\ \rho &= \sqrt{\textstyle\sum_{i=1}^{n} x_i^2} \end{aligned} \tag{A.2}$$

Figure A.1 illustrates relation between Cartesian and hyperspherical coordinate systems for 3-dimensional Euclidean space.

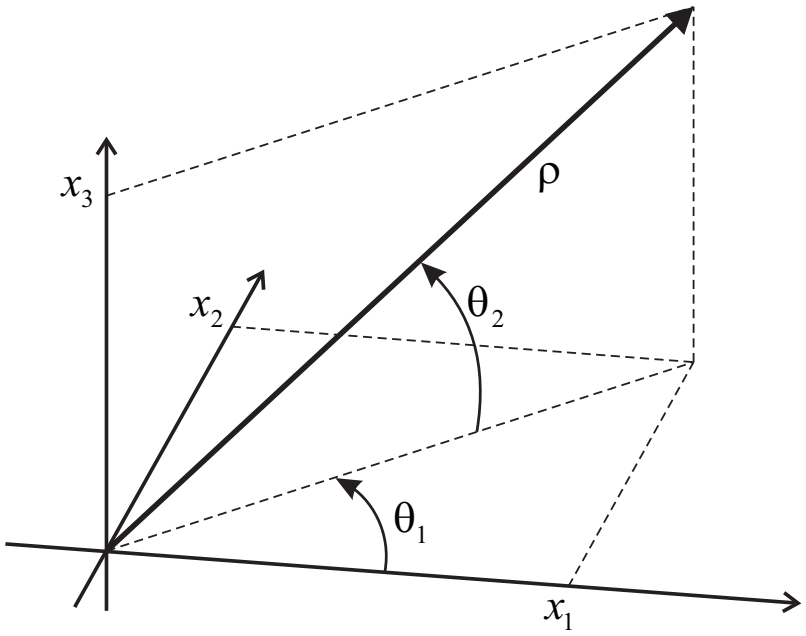


FIGURE A.1: Relation between Cartesian and hyperspherical coordinate systems.

Appendix B

Platonic Solids

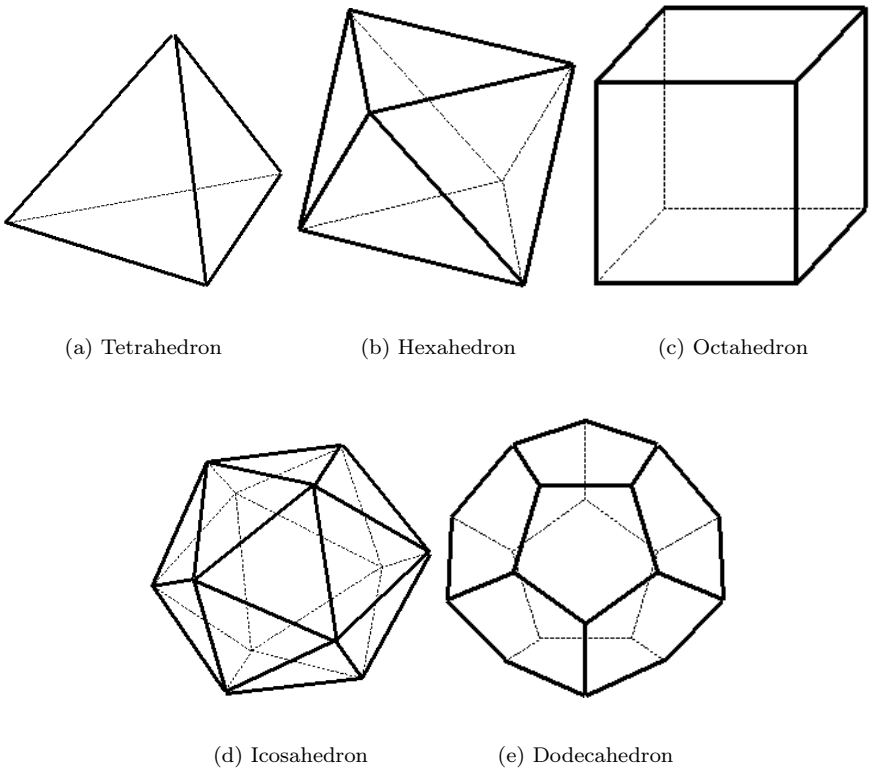
In 2-dimensional space any figure limiting a part of a plane can be defined as a set of line segments . Such a plane figure is called a polygon and consists of finite number of closed lines [103]. A regular polygon is a convex polygon with equal sides. A polyhedron is a figure in 3-dimensional space consists of set of polygons. A polyhedron is regular if all its faces are identical and regular polygons. Regular polyhedra are called "Platonic solids" and are symmetric 3-dimensional convex objects [104]. There are only five convex regular polyhedra (Tetrahedron, Cube, Octahedron, Dodecahedron, Icosahedron), Figure B.1. Below we present two common analyses that prove this limit.

Polyhedron is regular when it satisfies rules:

- It is convex.
- All faces are regular polygons.
- Each vertex has the same number of edges.

Each Platonic has its unique Schlafli symbol $\{p, q\}$ [105], where p defines the number of edges in a face and q defines the number of edges meeting at a vertex. From Schlafli symbol we can determine all other information, such as total number of vertices (V), edges (E), and faces (F). Since any edge joins two vertices and has two adjacent faces we have:

$$pF = 2E = qV \tag{B.1}$$



(a) Tetrahedron (b) Hexahedron (c) Octahedron

(d) Icosahedron (e) Dodecahedron

FIGURE B.1: Platonic solids.

After Platon, Euler discovered another relationship between these values:

$$V - E + F = 2. \tag{B.2}$$

The Schlafli symbols as well as values of V, E, F of the five Platonic solids are given in the Table B.1.

TABLE B.1: Platonic solids and their parameters.

Polyhedron	Vertices	Edges	Faces	Schlafli symbol
Tetrahedron	4	6	4	{3,3}
Cube	8	12	6	{4,3}
Octahedron	6	12	8	{3,4}
Dodecahedron	20	30	12	{5,3}
Icosahedron	12	30	20	{3,5}

Below we present two analyses of possible regular polyhedra.

Geometric approach. Each vertex of the solid must connect at least three faces. The sum of the flat angles connected to each vertex must be less than 2π. Because

each vertex connects regular polygons then maximal angle in the polygon (face) must be less than $2\pi/3$. Regular polygons with 6 and more vertexes have inner angles $\geq 2 * \pi/3$, so the face of polyhedron can be triangle, square, or pentagon.

Topological approach. Combining two above equations we have:

$$\frac{2E}{q} - E + \frac{2E}{p} = 2, \tag{B.3}$$

and after some simplification algebra:

$$\frac{1}{q} + \frac{1}{p} = \frac{1}{2} + \frac{1}{E}. \tag{B.4}$$

As E is always more than p and q we get an inequality:

$$\frac{1}{q} + \frac{1}{p} > \frac{1}{2}. \tag{B.5}$$

There are only five possibilities for Schlafli symbol $\{p, q\}$ that satisfy this inequality: $\{3, 3\}, \{4, 3\}, \{3, 4\}, \{5, 3\}, \{3, 5\}$.

Appendix C

Rotation Matrices and their Derivatives

Rotation - movement of data points in Euclidean space when at least one point of data space is stable [106]. We need to distinguish rotation, transformation, and reflection. Translation does not have any stable data point and just moves all data points along fixed direction on constant distance. Refection produces mapping of data points of Euclidean space $\mathbb{R}^n$ to itself through fixed $\mathbb{R}^m$ data space, where $m < n$. We can call any matrix $\mathbf{R}$ as a rotation matrix in n-dimensional space if $\mathbf{R}$ is square and orthogonal, and determinant of $\mathbf{R}$ is equal to 1:

$$\mathbf{R}^T = \mathbf{R}^{-1} \text{ and } det(\mathbf{R}) = 1. \tag{C.1}$$

In this appendix we outline two the most common realizations of the rotation matrix as well as their derivative forms. The first implementation is the Euler-type rotation, where any rotation is a sequence of 2-dimensional one. This operation can be very easily imagined but it has such disadvantage as complexity. The second approach is based on matrix exponential. This allows us to implement product and derivation of such a matrix very easily.

C.1 Euler-type Rotation Matrix

In real applications we meet 2 and 3-dimensional rotations, in general, rotation matrix for n-dimensional space can be defined as consecutive product of rotational

matrices in 2-dimensional spaces. A rotational matrix $\mathbf{R}$ in 2-dimensional space is:

$$\mathbf{R} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}. \tag{C.2}$$

A rotational matrix $\mathbf{R}$ in 3-dimensional space is consecutive product of 2-dimensional matrices:

$$\mathbf{R} = \begin{bmatrix} cos(\theta_{12}) & -sin(\theta_{12}) & 0 \\ sin(\theta_{12}) & cos(\theta_{12}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta_{13}) & 0 & -sin(\theta_{13}) \\ 0 & 1 & 0 \\ sin(\theta_{13}) & 0 & cos(\theta_{13}) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{23}) & -sin(\theta_{23}) \\ 0 & sin(\theta_{23}) & cos(\theta_{23}) \end{bmatrix} \tag{C.3}$$

In general case, (n, n)-dimensional rotation matrix $\mathbf{R}(\theta_{ij})$ in (i, j)-space is:

$$\mathbf{R}(\theta_{ij}) = \begin{bmatrix} 1 & 0 & & & \cdots & & & 0 \\ 0 & \ddots & & & \vdots & & & \\ & & cos(\theta_{ij}) & \cdots & 0 & \cdots & -sin(\theta_{ij}) & \\ \vdots & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots \\ & & sin(\theta_{ij}) & \cdots & 0 & \cdots & cos(\theta_{ij}) & \\ & \iddots & & & \vdots & & & \ddots \\ 0 & & & & \cdots & & & 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \vdots \\ \\ \\ 1 \end{matrix} \tag{C.4}$$

A rotational matrix $\mathbf{R}$ in n-dimensional space is consecutive product of 2-dimensional matrices:

$$\mathbf{R} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \mathbf{R}(\theta_{ij}). \tag{C.5}$$

Let's have a vector of rotation angles, $\theta_k,\ k = 1, \ldots, n(n-1)/2$, in n-dimensional space with one-to-one correspondence between k and pair of indexes , (i_k, j_k) denoting rotation plane. Then the s^{th} partial derivative of $\mathbf{R}(\theta_k)$ with respect to angle θ_k is $\partial^s \mathbf{R}(\theta)/\partial^s \theta_k = \mathbf{M}^s \mathbf{R}(\theta)$. In this case $\mathbf{M}$ is (n, n)-dimensional matrix such that $m_{i_k, j_k} = \delta_{i_k, j_k} - \delta_{j_k, i_k}$ and all odd and even partial derivatives of rotation matrix $\mathbf{R}(\theta)$ are equal to the first and the second derivatives respectively and are alternating sequences.

C.2 Skew-Symmetric Matrix Exponential

Real real skew-symmetric (n, n)-dimensional matrix $\mathbf{\Theta}$ is given, we can obtain a rotation matrix as $\mathbf{R}(\mathbf{\Theta}) = e^{\mathbf{\Theta}}$ [106]. We can define matrix exponential values using the power series of $e^{\mathbf{\Theta}}$:

$$e^{\mathbf{\Theta}} = \mathbf{I} + \mathbf{\Theta} + \frac{1}{2}\mathbf{\Theta}^2 + \frac{1}{6}\mathbf{\Theta}^3 + \cdots + \frac{1}{k!}\mathbf{\Theta}^k + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{\Theta}^k, \tag{C.6}$$

where $\mathbf{\Theta}$ is a skew-symmetric square matrix whose transpose is also its negative, $\mathbf{\Theta}^T = -\mathbf{\Theta}$. Here there is an example of a skew-symmetric $(3, 3)$-dimensional matrix $\mathbf{\Theta}$:

$$\mathbf{\Theta} = \begin{pmatrix} 0 & \theta_{12} & \theta_{13} \\ -\theta_{12} & 0 & \theta_{23} \\ -\theta_{13} & -\theta_{23} & 0 \end{pmatrix}. \tag{C.7}$$

For any skew-symmetric $\mathbf{\Theta}$, $e^{\mathbf{\Theta}}$ is always a rotation matrix. In general case of $\mathbf{\Theta}$, to find $e^{\mathbf{\Theta}}$ we can use the Pade approximation with scaling and squaring [107]. If $\mathbf{\Theta}$ is skew-symmetric and non-singular such that $\mathbf{\Theta} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ then $e^{\mathbf{\Theta}} = \mathbf{V}diag(e^{diag(\mathbf{D})})/\mathbf{V}$. As far as $\mathbf{V}$ is the orthogonal matrix then $\mathbf{V}^{-1} = \mathbf{V}^T$ and

$$\mathbf{R}(\mathbf{\Theta}) = e^{\mathbf{\Theta}} = e^{\mathbf{V}\mathbf{D}\mathbf{V}^T} = \mathbf{V}diag(e^{diag(\mathbf{D})})\mathbf{V}^T. \tag{C.8}$$

The s^{th} partial derivative of $\mathbf{R}(\mathbf{\Theta})$ with respect to angle θ_{ij} is $\partial^s\mathbf{R}(\mathbf{\Theta})/\partial^s\theta_{ij} = \mathbf{M}^s\mathbf{R}(\mathbf{\Theta})$, where, as for the case of Euler rotation, $m_{i_k,j_k} = \delta_{i_k,j_k} - \delta_{j_k,i_k}$.

Biography

Olexiy Kyrgyzov was born on 11 October 1979 in stanitsa Kanevskaya, Krasnodarskiy region, Russia. He received the Bachelor and Master Science degrees in Information Management Systems and Technologies from the Dnipropetrovsk National University (DNU), Dnipropetrovsk, Ukraine, in 2001 and 2002, respectively.

Between June 2002 and April 2006, he worked as an instructor at the Radiophysics department and a service engineer at the International department of DNU under the supervision of Prof. Valeriy Dolgov, Prof. Alexandr Akhmetshin, and Prof. Mihail Dyachenko.

Since 2006, he has been working towards his Ph.D. and been involved in investigation as a research assistant in the Computer Science and Electrical Engineering Department at Oregon Health and Science University under the supervision of Prof. Deniz Erdogmus. In 2008, together with his dissertation supervisor and other Ph.D. students, he moved to Boston (MA) and was transferred to the Department of Electrical and Computer Engineering at Northeastern university.

His current research interests broadly include nonlinear and statistical signal processing, machine learning, image processing, computer vision, and pattern recognition areas.

Dr. Kyrgyzov serves as a Reviewer for IEEE Transactions on Neural Networks, Journal of Neurocomputing (Elsevier), and IEEE International Symposium on Circuits And Systems.

Bibliography

[1] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[2] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[3] L. Minati and D. Aquino. Probing neural connectivity through diffusion tensor imaging DTI. *Cybernetics and Systems*, pages 263–268, 2006.

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

[5] Ledyard Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, September 1966.

[6] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(R1,R2,. . .,Rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000. ISSN 0895-4798.

[7] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *JOURNAL OF THE ACM*, 46(5):604–632, 1999.

[8] Tamara Kolda and Brett Bader. The TOPHITS model for higher-order web link analysis. In *Proceedings of the SIAM Data Mining Conference Workshop on Link Analysis, Counterterrorism and Security*, 2006.

[9] Morten Morup, Lars Kai Hansen, Christoph S. Herrmann, Josef Parnas, and Sidse M. Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg. *NeuroImage*, 29(3):938 – 947, 2006.

[10] Jacey-Lynn Minoi and Duncan Gillies. Sub-tensor decomposition for expression variant 3d faces recognition. In *3rd International Conference on Geometric*

Modeling and Imaging, pages 108–113. IEEE Computer Society, 2008. ISBN 978-0-7695-3270-7.

[11] Eric W. Weisstein. Taylor series. rom MathWorld–A Wolfram Web Resource., 2010. http://mathworld.wolfram.com/TaylorSeries.html.

[12] A. Papoulis and U. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw Hill, New York, 4th edition, 2002.

[13] F.L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *J. Math. and Physics*, 6(1):164–189, 1927.

[14] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *J. Math. Phys.*, 7:39–79, 1927.

[15] Raymond Cattell. Parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283, December 1944.

[16] C. J. Appellof and E. R. Davidson. Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Anal. Chem.*, 53: 2053Ű2056, 1981.

[17] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. Wiley, England, October 2004.

[18] Knuth Donald E. *The Art Of Computer Programming: Seminumerical Algorithms*, volume 2. Pearson Addision Wesley Prof, 3 edition, 1997.

[19] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun ichi Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multiway Data Analysis and Blind Source Separation*. Wiley, John & Sons, Incorporated, 1 edition, November 2009.

[20] B. W. Bader and T. G. Kolda. Matlab tensor toolbox. Website, 2010. http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/.

[21] Brett W. Bader and Tamara G. Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. *ACM Trans. Math. Softw.*, 32(4):635–653, 2006. ISSN 0098-3500.

[22] Alfred Gray, Elsa Abbena, and Simon Salamon. *Modern Differential Geometry of Curves and Surfaces with Mathematica, Third Edition (Studies in Advanced Mathematics)*. Chapman & Hall/CRC, 2006. ISBN 1584884487.

[23] Henk A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

[24] Pierre Comon, Gene Golub, Lek-Heng Lim, and Bernard Mourrain. Symmetric tensors and symmetric tensor rank, 2008.

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

[26] P. Comon. Canonical tensor decompositions. In *In Mathematics in Signal Processing*, pages 1–24. Clarendon Press, 2002.

[27] M.A.O. Vasilescu and D. Terzopoulos. Multilinear independent components analysis. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 547–553, June 2005.

[28] Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006.

[29] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part i: Lemmas for partitioned matrices. *SIAM J. Matrix Anal. Appl.*, 30(3):1022–1032, 2008. ISSN 0895-4798.

[30] J. Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Younga decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[31] Evrim Acar and Bu"lent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21 (1):6–20, 2009. ISSN 1041-4347.

[32] Rasmus Bro, Richard A. Harshman, Nicholas D. Sidiropoulos, and Margaret E. Lundy. Modeling multi-way data with linearly dependent loadings. *Journal of Chemometrics*, 23(7-8):324–340, 2005.

[33] André Almeida De, Gérard Favier, and João César Mota. The constrained block-PARAFAC decomposition. In *TRICAP'06*, Chania Grèce, 06 2006.

[34] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM J. Matrix Anal. Appl.*, 30(3):1033–1066, 2008. ISSN 0895-4798.

[35] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms—part iii: Alternating least squares algorithms. *SIAM J. Matrix Anal. Appl.*, 30(3):1067–1083, 2008. ISSN 0895-4798.

[36] C. Navasca, L. De Lathauwer, and S. Kindermann. Swamp reducing technique for tensor decomposition. In *Proc. 16th European Signal Processing Conference (EUSIPCO 2008)*, Lausanne, Switzerland, Aug. 25–29 2008.

[37] C.F. Beckmann and S.M. Smith. Tensorial extensions of independent component analysis for multisubject fmri analysis. *NeuroImage*, 25(1):294 – 311, 2005. ISSN 1053-8119.

[38] Rasmus Bro, Nicholaos D. Sidiropoulos, and Age K. Smilde. Maximum likelihood fitting using ordinary least squares algorithms. *Journal of Chemometrics*, 16(8-10):387–400, 2002.

[39] Vega L. Montoto and P. D. Wentzell. Maximum likelihood parallel factor analysis (MLPARAFAC). *Journal of Chemometrics*, 17(4):237–253, 2003.

[40] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111Ű126, 1994.

[41] Daniel D. Lee and Sebastian H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[42] Amnon Shashua and Tamir Hazan. Nonnegative tensor factorization with applications to statistics and computer vision. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 792–799, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5.

[43] Tamir Hazan, Simon Polak, and Amnon Shashua. Sparse image coding using a 3d non-negative tensor factorization. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 50–57, Washington, DC, USA, 2005. IEEE Computer Society.

[44] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *ECCV06*, pages IV: 595–608, 2006.

[45] M. Mørup, L. K. Hansen, and S. M. Arnfred. Algorithms for sparse non-negative TUCKER. *Neural Computation*, 20(8):2112–2131, Aug 2008.

[46] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Non-negative tensor factorization using alpha and beta divergences. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2007.*, volume 3, pages III–1393–III–1396, June 2007.

[47] Olexiy O. Kyrgyzov and Deniz Erdogmus. Geometric structure of sum-of-rank-1 decompositions for n-dimensional order-p symmetric tensors. In *ISCAS*, pages 1340–1343, Seattle, WA, USA, May 2008.

[48] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0-8018-5414-8.

[49] P. Comon and B. Mourrain. Decomposition of quantics in sums of powers of linear forms. *Signal Processing*, 53:93–107, 1996.

[50] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Math. Comp*, 64:763–776, 1995.

[51] H. H. Goldstine, F. J. Murray, and J. von Neumann. The Jacobi method for real symmetric matrices. *J. ACM*, 6(1):59–96, 1959.

[52] E.B. Saff and A.B. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, December 1997.

[53] L. Lovisolo and E.A.B. da Silva. Uniform distribution of points on a hyper-sphere with applications to vector bit-plane encoding. *Vision, Image and Signal Processing, IEE Proceedings*, 148(3):187–193, Jun 2001.

[54] Paul Leopardi. *Distributing points on the sphere: Partitions, separation, quadrature and energy*. Doctor of philosophy in mathematics, University of New South Wales, April 2007.

[55] Rob Womersley. Distributing points on the sphere. Website, School of Mathematics, UNSW, March 2010. http://www.maths.unsw.edu.au/about/distributing-points-sphere.

[56] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2 edition, 2001.

[57] S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate voronoi diagrams. *Proc. 34th ACM Symp. on Theory of Computing*, page 721Ũ730, 2002.

[58] J.J. Thomson. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *Philosophical Magazine*, 6(39), 1904.

[59] P. M. L. Tammes. On the origin of number and arrangement of the places of exit on the surface of pollen grains. *Recueil Travaux Botaniques Nerlandais*, 27:1–84, 1930.

[60] Fejes Toth. On the densest packing of spherical caps. *American Mathematical Monthly*, 56:330–331, 1949.

[61] Lloyd N. Trefethen Jean-Paul Berrut. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501Ű517, 2004.

[62] Ian H. Sloan and Alvise Sommariva. Approximation on the sphere using radial basis functions plus polynomials. *Advances in Computational Mathematics*, 29: 147Ű177, 2008.

[63] P. Delsarte, J.M. Goethals, and J.J. Seidel. Spherical codes and designs. *Geometriae Dedicata*, 6(3):363–388, 1977.

[64] R. S. Womersley and I. H. Sloan. How good can polynomial interpolation on the sphere be? *Advances in Computational Mathematics*, 14:195–226, 2001.

[65] Hugo Steinhaus. *Mathematical Snapshots*. Dover Publications; 3 edition, 3 edition, July 1999. ISBN 0486409147.

[66] I.N. Bronshtein, K.A. Semendyayev, G. Musiol, and H. Muehling. *Handbook of mathematics (4rd ed.)*. Springer-Verlag, 2007. ISBN 978-3-540-43491-7.

[67] Olexiy Kyrgyzov. Non-redundant tensor decomposition. Website, 2010. http://www.ece.neu.edu/~kyrgyzov/TensorDecomposition.

[68] R. Abramov. The multidimensional maximum entropy moment problem: A review on numerical methods. *Communications in Mathematical Sciences*, 8 (2):377Ů392, 2009.

[69] K. Bandyopadhyay, A.K. Bhattacharya, P. Biswas, and D.A. Drabold. Maximum entropy and the problem of moments: a stable algorithm. *Physical Review E*, 71, 2005.

[70] D. Ormoneit and H. White. An efficient algorithm to compute maximum entropy densities. *Econometric Reviews*, 18(2):127–140, 1999.

[71] Rafail V. Abramov and Andrew J. Majda. Quantifying uncertainty for non-gaussian ensembles in complex systems. *SIAM J. Sci. Comput.*, 26(2):411–447, 2005. ISSN 1064-8275.

[72] L. R. Mead and N. Papanicolaou. Maximum entropy in the problem of moments. *Journal of Mathematical Physics*, 25:2404–2417, August 1984.

[73] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2 edition, 2006.

[74] L. Giraud, J. Langou, and M. Rozloznik. On the loss of orthogonality in the gram-schmidt orthogonalization process. *Computers and Mathematics with Applications 50*, pages 1069–1075, 2005.

[75] Bernard. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1 edition, 1986.

[76] U. Ozertem, D. Erdogmus, and O. Arikan. Piecewise smooth signal denoising via principal curve projections. In *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 426 –431, 2008.

[77] Daniel T. Larose. *Data Mining Methods and Models*. Wiley-IEEE Press, 2006.

[78] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[79] Richard O. Duda and Peter E. Hart nad David G. Stork. *Pattern Classification*. Wiley, 2 edition, 2000.

[80] T. Hastie and W. Stuetzle. Principal curves. *Jour. Am. Statistical Assoc.*, 84: 502–516, 1989.

[81] Deniz Erdogmus and Umut Ozertem. Self-consistent locally defined principal surfaces. *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2:549–552, 2007.

[82] U. Ozertem and D. Erdogmus. Local conditions for critical and principal manifolds. In *IEEE International Conference on Acoustics, Speech and Signal Processing.*, pages 1893–1896, April 2008.

[83] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32 – 40, Jan 1975.

[84] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995.

[85] R. Jenssen, T. Eltoft, M. Girolami, and D. Erdogmus. Kernel maximum entropy data transformation and an enhanced spectral clustering algorithm. *Neural Information Processing Systems*, 2006.

[86] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (gpca). *IEEE Conference on Computer Vision and Pattern Recognition*, 1:621 – 628, 2003.

[87] Elzbieta Pekalska and Robert P. W. Duin. *The dissimilarity representation for pattern recognition: foundations and applications*. World Scientific, 2005.

[88] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*. John Wiley and Sons, 2001.

[89] A. Renyi. Measures of entropy and information. *Selected Papers of Alfred Renyi*, 2:565–580, 1976.

[90] M. Girolami. Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Computation*, 14(3):669–688, 2002.

[91] R. Vidal. *Generalized Principal Component Analysis (GPCA)*. PhD thesis, University of California at Berkeley, 2003.

[92] Danijel Skocaj, Horst Bischof, and Ales Leonardis1. A robust pca algorithm for building representations from panoramic images. In *In European Conference Computer Vision*, pages 761–775. Springer, 2002.

[93] D. Skočaj, A. Leonardis, and H. Bischof. Weighted and robust learning of subspace representations. *Pattern recogn.*, 40(5):1556–1569, May 2007. URL http://vicos.fri.uni-lj.si/data/publications/skocajPR07.pdf.

[94] F. De la Torre and M. Black. Robust principal component analysis for computer vision. *ICCV*, page 362Ű369, 2001.

[95] E. J. Candes, Y. Ma X. Li, , and J. Wright. Robust principal component analysis? Website, Department of Mathematics, Stanford University, April 2009. http://www-stat.stanford.edu/~candes/papers/RobustPCA.pdf.

[96] S. Roweis. EM algorithms for PCA and SPCA. *NIPS*, page 626Ű632, 1997.

[97] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. Technical report, Microsoft Research, 1999.

[98] Brett W. Bader. Tensor-krylov methods for solving large-scale systems of nonlinear equations. *SIAM J. Numer. Anal.*, 43(3):1321–1347, 2005. ISSN 0036-1429.

[99] Brett Bader. Tensor-krylov methods for solving large-scale systems of nonlinear equations. Contributed talk at CASC Workshop, 2003.

[100] Ali Bouaricha and Robert B. Schnabel. Tensor methods for large sparse systems of nonlinear equations. *Mathematical Programming*, 82(3):377–400, 2005.

[101] Robert B. Schnabel and Paul D. Frank. Tensor methods for nonlinear equations. *SIAM Journal on Numerical Analysis*, 21(5):815–843, 1984.

[102] Lambert M. Surhone, Miriam T. Timpledon, and Susan F. Marseken. *Spherical Coordinate System: Vector Fields in Cylindrical and Spherical Coordinates, Sphere, N-sphere, Cartesian Coordinate System, Cylindrical Coordinate System*. Betascript Publishers, 2009.

[103] H. S. M. Coxeter. *Introduction to Geometry*. Wiley, New York, 2 edition, 1969.

[104] Alexey Stakhov. Platonic solids. Website, 2010. http://www.goldenmuseum.com/0213Solids_engl.html.

[105] H. S. M. Coxeter. *Regular polytopes*. Courier Dover Publications, 1973.

[106] Jean Gallier and Dianna Xu. Computing exponentials of skew symmetric matrices and logarithms of orthogonal matrices. *International Journal of Robotics and Automation*, 18:10–20, 2000.

[107] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005. ISSN 0895-4798.