

HIGHER ORDER TENSOR DECOMPOSITIONS: FROM INTUITION TO IMPLEMENTATION AND APPLICATION

Ilya Kisil

Communications & Signal Processing Group
Department of Electrical & Electronic Engineering
Imperial College London

A thesis submitted in fulfilment of the requirements for the degree of Doctor of
Philosophy of Imperial College London

November 2021

Statement of Originality

I declare that the work presented in this thesis is the outcome of my research under the supervision of Prof. Danilo P. Mandic. Ideas and results from the work of others are appropriately acknowledged with the standard referencing practices employed in this discipline. The material in this thesis has not been submitted for any degree at any other academic or professional institution.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Abstract

The exponentially increasing availability of Big Data and streaming services comes as a direct consequence of the rapid development and widespread use of multi-sensor technology. The quest to make sense from such large volumes and varieties of data has both highlighted the limitations of standard “flat-view” matrix models and the necessity to move toward more versatile data analysis tools. One such algebraic framework which is naturally suited to data of large volume, variety, and veracity is a multi-way array or a tensor. The associated tensor decompositions have been recognised as a viable system to break the “Curse of Dimensionality”, an exponential increase in data volume with the tensor order. Owing to a scalable way in which they deal with multi-way data and their ability to exploit inherent deep data structures when performing feature extraction, tensor decompositions have found application in a wide range of disciplines, from very theoretical ones, such as scientific computing and physics, to the more practical aspects of signal processing and machine learning.

Despite the obvious advantages, multi-dimensional analysis through tensors is not as intuitive as classical linear algebra, its counterpart for matrices and vectors. Also, tensor expressions are not so easily manipulated while the notation can be overwhelming; these may become obstacles when it comes to understanding the physical meaning of underlying operations. To this end, the material in this thesis is supported by numerous self-explanatory diagrams and examples, to equip interested readers with a natural way to grasp more sophisticated concepts inherent to multi-linear algebra.

An overreaching theme of this thesis is joint analysis of multi-block information that can be naturally represented in a form of an N -way array. We demonstrate that the flexibility of tensor decompositions provides a unique opportunity to explore tensor-value data, for example through a projection onto a lower dimensional subspace where latent components can be discovered more easily. Upon providing an intrinsic link between the properties of the outer vector product, we develop a novel method for common and individual feature extraction. The proposed framework is shown to be suitable for data structures obtained from multiple observations of the same phenomenon under various interconnected conditions, such as a collection of images of the same objects taken under a range of lighting conditions and from different angles.

Another aim of this thesis is to improve performance efficiency of tensor operations, the computation of which can be prohibitively expensive or completely infeasible on the raw input data in big data applications. At first, we devise a new algorithm for boosting computation of the contraction product by virtue of the Tensor Train decomposition, before introducing a cutting edge approach for evaluation of the Tucker decomposition based on the recursive update strategy that takes place only upon arrival of a new tensor-variate data sample. The efficacy of the proposed algorithms opens new avenues for the next generation of real-time streaming technologies and applications.

Next, we proceed to fill the void in the open literature by introducing a novel framework that naturally incorporates physically meaningful tensor decompositions and extends classical ensemble learning in order to accommodate the intrinsically multi-way and multi-modal data. This contribution is further supported by a blood flow analysis of patients that underwent renal replacement therapy and were under constant surveillance for identifying failure in cardiovascular procedure.

Finally, after revising publically available software for multi-linear analysis, we aspire to bridge the gap between sophisticated algorithms and their practical implementations. To this end, an open source toolbox is developed which offers a unifying interface that establishes a self-sufficient ecosystem for various forms of efficient representation of multi-way data and associated tensor decompositions and promotes rapid development and quick iteration from an idea to an initial functioning prototype for anyone familiar with existing machine learning libraries.

Contents

Statement of Originality	1
Copyright Declaration	3
Abstract	5
Contents	7
List of Figures	9
List of Tables	13
Nomenclature	17
Acknowledgements	21
Publications	23
1 Introduction	25
1.1 Perspective	25
1.2 Development and Adoption of Tensor Decompositions	27
1.3 Problem Statement	29
1.4 Thesis Contribution and Organisation	30
2 Prerequisites: Multilinear Algebra	35
2.1 Notation and Multilinear Operations	35
2.1.1 Basic definitions and conventions	35
2.1.2 Fundamental operations	37
2.1.3 Graphical representation of the main components	42
2.2 Tensor Generation and Data Rearrangement	43
2.2.1 Rearrangement of lower-dimensional data	43
2.2.2 Mathematical construction and experiment design	46
2.3 Tensor Decompositions	47
2.3.1 Canonical polyadic decomposition and Kruskal representation	47
2.3.2 Tucker decomposition and representation	50
2.3.3 Tensor train decomposition and representation	53
2.4 Chapter Summary	55
3 Feature Extraction and Data Fusion	57
3.1 Common and Individual Feature Extraction	57
3.1.1 Introduction	57
3.1.2 Conventional approach to multiblock data	58
3.1.3 Common and individual feature extraction	61

Contents

3.1.4	The LL1 decomposition with non-negativity constraint	62
3.1.5	Experimental results and discussion	65
3.2	Feature Fusion via Tensor Network Summation	68
3.2.1	Introduction	68
3.2.2	Summation framework	68
3.2.3	General properties	73
3.2.4	Feature fusion via TN summation	74
3.3	Chapter Summary	76
4	Tensor Decompositions and Wisdom of the Crowd	79
4.1	Introduction	79
4.2	Ensemble Learning: The Bagging Approach	80
4.3	Mixture of Experts: Multidimensional Crowd	81
4.4	Tensor Ensemble Learning for Classification	82
4.5	Multi-linear Learning for Cardiovascular Diagnosis	85
4.6	Chapter Summary	88
5	Computational Efficiency and Practical Implementations	89
5.1	Tensor Contraction Product via Tensor Train Decomposition	90
5.1.1	Introduction	90
5.1.2	Common multi-linear operations through nodes and edges	91
5.1.3	Graphical approach to contraction of tensors in the TT-format	94
5.1.4	Complexity analysis and numerical simulations	97
5.2	Recursive Tensor Decompositions for Streaming Big Data	100
5.2.1	Introduction	100
5.2.2	Preliminaries	101
5.2.3	Recursive HOSVD	102
5.2.4	Implementation considerations	104
5.2.5	Simulations	106
5.3	Higher Order Tensors ToolBOX	106
5.3.1	Introduction	106
5.3.2	Ecosystem of HOTTBOX	108
5.3.3	Core Structural Components	109
5.3.4	Supported Algorithms	111
5.3.5	Interactive Visualisations	114
5.3.6	Statistical Analysis Tools	114
5.4	Chapter Summary	115
6	Conclusions and Future Work	117
6.1	Conclusions	117
6.2	Future Work	119
	References	123

List of Figures

1.1	The 4V's of Big Data	26
1.2	Application of multi-linear analysis across data science fields.	27
1.3	Thesis structure	33
2.1	Graphical representation of the mode-1, $\underline{\mathbf{X}}_{(1)}$, mode-2, $\underline{\mathbf{X}}_{(2)}$, and mode-3, $\underline{\mathbf{X}}_{(3)}$, unfoldings on a 3-rd order tensor, $\underline{\mathbf{X}}$	37
2.2	Graphical representation of multi-dimensional array, $\underline{\mathbf{A}} \in \mathbb{R}^{3 \times 3 \times 3}$, of rank-1 which was obtained through a sequence of outer products between elementary units, i.e. vectors $\mathbf{a} \in \mathbb{R}^3$, $\mathbf{b} \in \mathbb{R}^3$ and $\mathbf{c} \in \mathbb{R}^3$	40
2.3	From a scalar to a tensor: a graphical approach to data representation. From left to right, top to bottom: 0-th order (scalar), 1-st order (vector), 2-nd order (matrix) and 3-rd order tensors.	42
2.4	Graphical representation of fundamental operations in form of a TN. From top to bottom: matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ and its transpose; contraction between matrices $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{Y} \in \mathbb{R}^{J_1 \times J_2}$ along the common mode $I_2 = J_1$; contraction between tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ along the common mode $I_3 = J_1$; contraction between tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ along all modes; unfolding of tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ along its first mode I_1	44
2.5	Tensor construction for data obtained during EEG recording of a subject. Signals of brain activity captured from different channels are first converted into a matrix form through STFT and then stacked together along the 3-rd dimension to form an order-3 tensor with <i>Channels</i> $\times$ <i>Time</i> $\times$ <i>Frequency</i> modes.	46
2.6	Illustration of the CPD for a 3-rd order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$. Left: Conventional illustration of the CPD. Right: Representation through the TN framework. . . .	48
2.7	Illustration of the TKD for a 3-rd order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$. Left: Conventional illustration of the TKD. Right: Representation through TNs framework.	50
2.8	Illustration of the TT decomposition for a 5-th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$. Top: Conventional illustration of the CPD. Bottom: Representation through TNs framework.	53
3.1	Conventional approaches to analysis of coupled datasets composed of “flat-view” samples.	60
3.2	Link between the outer product and the common features and rank for multi-dimensional data. The ensemble consists of the grey, yellow, magenta, bright blue and brown-orange colors, each of which is a combination of three base colors: red, green, blue. Their intensity is specified by the value at the corresponding position of mode-3 vectors $\mathbf{c}_R, \mathbf{c}_G, \mathbf{c}_B$	62
3.3	Interchangeable definitions of the Canonical Polyadic Decomposition. The original tensor $\underline{\mathbf{X}}$ is represented as a sum of rank-1 tensors $\underline{\mathbf{X}}_r = \lambda_r \cdot \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ and as mode-n product of the core tensor $\underline{\mathbf{A}}$ with the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$	63

3.4	The LL1 decomposition – specific case of the Block Term Decomposition. Each summation term is the Tucker model with $\text{mlrank}(\underline{\mathbf{X}}_r) = \text{mlrank}(\text{rank}(\mathbf{A}_r), \text{rank}(\mathbf{B}_r), 1)$ where $\text{rank}(\mathbf{A}_r) = \text{rank}(\mathbf{B}_r)$	63
3.5	Examples of the tensor representations, $\underline{\mathbf{X}}^m \in \mathbb{R}^{112 \times 92 \times 40}$ for tensor-based common and individual feature extraction, whereby the images of 40 different subjects were stacked along the third dimension during the training stage.	65
3.6	Common and individual feature extraction. Left: Examples of images in the ORL dataset. Center: Examples of common features computed through the CPD and LL1 decompositions. Right: Examples of extracted individual features obtained by subtraction of the common information from the corresponding original images.	66
3.7	Cosine similarity between samples in the dataset before (left) and after (center, right) individual information has been extracted. Each group of 40 randomly selected samples was processed separately. The yellow and blue colors stand respectively for the highest and lowest similarity level.	66
3.8	Classification accuracy for the CPD with different values of tensor rank.	67
3.9	The duality of a Tensor Network and Tensor Decomposition. Equivalent representation an 8-th order tensor (top left) as a standard TKD form with all physical nodes (top right) and as a TN of small interconnected contraction nodes in red color and the same physical nodes of purple color (bottom).	69
3.10	Topology preservation for the TN summation operator, illustrated for a sum of matrices $\mathbf{X}$ and $\mathbf{Y}$, expressed as a matrix chain. The matrices (nodes) $\mathbf{C}_n$ are composed of the matrices $\mathbf{A}_n$ and $\mathbf{B}_n$, which are arranged according to the sum of TNs framework, either through: (i) concatenation; or (ii) block-diagonal arrangement.	70
3.11	Topology of a sum of tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format.	71
3.12	Graphical representation of the sum of tensors $\underline{\mathbf{X}}, \underline{\mathbf{Y}}$ expressed in the Tensor Train format.	73
3.13	An example of TN summation for the TKD representations of the images in the training set. Left: A TKD arrangement of matrices and core tensors. Right: Sum of the Tucker factors represented in the TN format. Notice that upon TN summation the dimensions of the physical modes remain unchanged.	75
3.14	Classification accuracy of the SVM algorithm illustrated as a function of the proportion of training size. Three different training sets were formed, each of which contained either original samples, features extracted through tensor stacking and TN summation frameworks.	77
4.1	Tensor ensemble learning through the HOSVD with multi-linear rank (R_a, R_b, R_c) . All factor vectors of training samples, $\underline{\mathbf{X}}^m$, are reorganised into separate datasets, thus only requiring to train $N = R_a + R_b + R_c$ classifiers. At the final stage, a majority vote is used to classify a new unlabelled sample, $\underline{\mathbf{X}}^{new}$, achieved by aggregation of the individually learned knowledge about training data extrapolated to extracted latent components $\mathbf{a}_1^{new}, \dots, \mathbf{c}_{R_c}^{new}$	82
4.2	A conceptual diagram of the <i>Tensor Ensemble Learning</i> (TEL) framework. Each N-dimensional data sample (top left) from the original dataset is decomposed through tensor factorisation into a Kruskal or Tucker representation (top right). The so-extracted components are reorganised and form a series of new datasets, each containing an incomplete information about the original sample (middle). An ensemble of machine learning algorithms is then employed, whereby each base learner generates an independent hypothesis which is then used for “voting” in order to form a collective decision (bottom).	83

4.3	Comparison of the overall test performance for Bagging and TELVI approaches on the ETH-80 dataset, illustrated as a function of the proportion of training size proportion and across a variety of classification algorithms used as the base estimators.	84
4.4	Performance of the employed base classifiers which were individually trained and used for prediction of a particular latent component from factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, computed through HOSVD.	85
4.5	Tensorisation of B-mode ultrasound cine loop recording by breaking it down into a set of consecutive frames which are stacked into a tensor along the “frame” dimension.	87
4.6	Spectrograms of “frame” features obtained via the CPD of a recording. Left: A subject with stenosis exhibits higher frequencies across the entire time which correspond to high velocities and turbulent flow. Right: Laminar blood flow through vessels is characterised by the lower frequencies within the whole recording of a subject that does not have stenosis.	88
5.1	Example of two relatively involved tensor operations, but easily comprehensible through a graphical representation. Left: A TN representing a 16-th order tensor, with contraction nodes shown in blue, and physical nodes shown in green. Right: A TN representing a 32-th order tensor, with contraction nodes shown in blue (3-rd order) and orange (4-th order), and physical nodes in green. Image adapted from [1].	90
5.2	Graphical representation of $\mathbf{G} = \mathbf{X} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \mathbf{A}^{(3)T}$. By starting from the TKD of $\mathbf{X}$ and exploiting the fact that $\mathbf{A}^{(n)T} \mathbf{A} = \mathbf{I}_{R_n}$, the equality becomes evident.	91
5.3	A tensor inner product, from a graphical perspective. Treating the matrices as individual blocks and changing the order of contraction, it is possible to manipulate the original inner product into one expressed differently, but ultimately equivalent.	93
5.4	Graphical justification of (2.56). By performing the 1-st mode unfolding graphically, it becomes clear that $\mathbf{X}_{(1)} = \mathbf{A}^{(1)} \mathbf{G}_{(1)} \mathbf{K}$, where $\mathbf{K} = (\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T$	94
5.5	A simple example to show the possibility of different expressions for the same topology. In this case, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ can be written as in (5.7), as shown on the left-hand side in the figure, or as in (5.8), on the right-hand side.	95
5.6	Graphical illustration of TTCP. The contraction is performed on a 4th-order tensor $\mathbf{X}$ and a 3rd-order tensor $\mathbf{Y}$ along common modes $I_3 = J_2$. This is achieved by exploiting the topology of TTs. The TT decompositions of the permuted versions of $\mathbf{X}$ and $\mathbf{Y}$ are obtained, allowing for the reduction of the contraction to a single matrix multiplication.	96
5.7	Theoretical time complexity (in log-scale) required to perform the standard contraction product (left) versus the proposed TTCP (right) for tensors in TT-format with varying dimensions, I , tensor orders, N , and TT ranks, R	98
5.8	Left: Comparison of computational cost (in seconds, log-scale) between (2.24) and the TTCP for the $\mathbf{X} \times_1^1 \mathbf{Y}$ contraction. The 3-rd, 4-th and 5-th order cases are considered. Stemming from a graphical intuition, the TTCP is more computationally efficient than the direct definition of contraction. Right: The TTCP approximation error measured in explained variance (y-axis) as a function of TT rank (x-axis) across three popular machine learning datasets. Note that a TTCP with a low rank of $R_n = 4, n = 1, \dots, N - 1$ is often enough to approximate the original contraction product by more than 90%.	99
5.9	Graphical illustration of an “all-but- n ” contraction. $\langle \mathbf{X}, \mathbf{Y} \rangle_{-3} = \mathbf{X}_{(3)} \mathbf{Y}_{(3)}^T$ on tensors $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathbf{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $I_1 = J_1, I_2 = J_2$	99
5.10	Evolution of the principal angle over time. Convergence of R-HOSVD is maintained despite the regime-change.	107

List of Figures

5.11	Biomedical data (left), a video sequence (middle), and 3-D motion capturing system (right) organised as “raw” multi-dimensional data.	108
5.12	Left: The ecosystem of representations of an N-dimensional array; Right: The synergy of meta-information associated with an N-dimensional array descriptions.	110
5.13	An example of a dashboard for visualising the TensorTKD . It exploits the nature of relationships between the components and utilises the associated meta-information.	115
5.14	The ratio of distinct parameters between the structured tensor model, η_{tensor} , and the classical unstructured model, η_{multi} , given by $\frac{\eta_{\text{tensor}}}{\eta_{\text{multi}}}$, for a varying mode dimensionality, I , and tensor order, N	115

List of Tables

2.1	Basic Tensor and Matrix Notation	38
2.2	Definition of Main Products	39
3.1	Classification Performance in %	68
3.2	Algebraic properties of TN summation	74
3.3	Classification rates for the ETH-80 dataset	77
5.1	Computational complexity breakdown of R-HOSVD	106

List of Algorithms

1	ALS for the CP decomposition of an N -th order tensor	50
2	HOSVD for an N -th order tensor	52
3	HOOI for an N -th order tensor	52
4	TT-SVD for an N -th order tensor	54
5	LL1 decomposition of a 3rd-order tensor with non-negative constraint	64
6	Sum of TNs for feature extraction	76
7	Tensor-Train Contraction Product (TTCP)	97
8	Recursive HOSVD for streaming multi-dimensional data	105

Nomenclature

Acronyms

ALS	Alternating Least Squares
BTD	Block Term Decomposition
CANDECOMP	Canonical Decomposition
CCA	Canonical Correlation Analysis
CIFE	Common and Individual Feature Extraction
CMTF	Coupled Matrix and Tensor factorisation
CPD	Canonical Polyadic Decomposition
DFT	Discrete Fourier Transform
ECG	Electrocardiogram
EEG	Electroencephalogram
ERP	Event Related Potentials
EVD	Eigenvalue Decomposition
FFT	Fast Fourier Transform
HOOI	Higher-Order Orthogonal Iteration
HOSVD	Higher-Order Singular Value Decomposition
KNN	K-Nearest Neighbors
MLP	Multi-Layer Perceptron
MMSE	Minimum Mean Square Error
MPS	Matrix Product State
MSE	Mean Square Error
NMF	Non-negative Matrix Factorisation
NN	Neural Network
PARAFAC	Parallel Factor
PCA	Principal Component Analysis
RF	Random Forest

Nomenclature

RHOSVD	Recursive Higher-Order Singular Value Decomposition
SNR	Signal-to-Noise Ratio
STFT	Short Time Fourier Transform
STM	Support Tensor Machine
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TD	Tensor Decomposition
TEL	Tensor Ensemble Learning
TELVI	Tensor Ensemble Learning Vectors Independently
TKD	Tucker Decomposition
TN	Tensor Network
TT	Tensor Train
TTCP	Tensor Train Contraction Product
Products and Operators	
$\circ$	Outer product
$\times_n$	Mode- n product
$\times_n^m$	Contraction product
$\circledast$	Hadamard product
$\odot$	Khatri-Rao product
$\otimes$	Kronecker product
$\bigcirc_{n=1}^N$	Sequential outer product
$\bigoplus_{n=1}^N$	Sequential Hadamard product
$\bigodot_{n=1}^N$	Sequential Khatri-Rao product
$\bigotimes_{n=1}^N$	Sequential Kronecker product
$\bigtimes_{n=1}^N$	Sequential mode- n product
$\text{Diag}(\cdot)$	Operator to extract elements that occupy main diagonal
$\text{diag}(\cdot)$	Operator for placing elements of a vector onto main diagonal
$\text{stack}_n(\cdot)$	Stacking operator that combines samples along a new dimension
$\text{tSVD}(\cdot)$	Operator that perform truncated SVD
$\text{vec}(\cdot)$	Vectorisation

$(\cdot)^{-1}$	Inverse
$(\cdot)^\dagger$	Moore-Penrose pseudo-inverse
$(\cdot)^T$	Transposition
$\langle \cdot, \cdot \rangle$	Inner product
$\llbracket \cdot ; \cdot \rrbracket$	Representation through tensor factorisation
$\ \cdot \ _F$	Frobenius norm
$\langle\!\langle \cdot ; \cdot \rangle\!\rangle$	Representation through Tensor Train factorisation

Symbols

x	Scalar (lowercase letter)
$\mathbf{x}$	Vector (bold lowercase letter)
$\mathbf{X}$	Matrix (bold uppercase letter)
$\underline{\mathbf{X}}$	Tensor (underlined bold uppercase letter)
$\underline{\mathcal{X}}$	Tensor Network (underlined Euler script letter)
$\mathcal{X}(\cdot)$	Ensemble of samples (calligraphic uppercase letter)
$\mathcal{O}(\cdot)$	Big “O” operator
$\mathbb{R}$	Field of real numbers

Acknowledgements

First of all, I would like to thank Prof. Danilo Mandic who gave me the opportunity to undertake this PhD and for being my supervisor over the last few years. It was a pleasure to be part of his research team and to work alongside other PhD students and postdocs with a great sense for productive research and enjoyable breaks. He provided me with the unique opportunity to be involved in a wide range of very different projects. There is still a lot left to learn from him but overall, he has given me ample opportunity to grow as a researcher, for which I am truly grateful.

Next, I am thankful to Dr. Valentin Goverdovsky, Dr. Sithan Kanna, and Dr. Wilhelm von Rosenberg for their enlightening technical advice, comments, and patience with answering my countless questions which I greatly benefited from at the beginning of my research. This made the start in a new environment a lot easier.

I would like to acknowledge the financial support that I have received from the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1).

Moreover, I wish to express gratitude to my colleagues and collaborators Bruno Scalzo Dees and Giuseppe Calvi for the lively and heated arguments, and never-ending discussions on every imaginable topic which led to numerous publications.

I extend my thanks to my peers and friends, including Ahmad Moniri, David Joseph, Metin Yarici, Takashi Nakamura, Alex Stott who were always open to a cheeky pint and would transform an exhaustive and stressful day, week, month into enjoyable and memorable times. There can be no overstating the contributions at the personal and professional level of so many people who I was surrounded by throughout my time at Imperial.

Finally and most importantly, I deeply thank my incredible family, for their unconditional trust, timely encouragement, and endless patience. It was their love that raised me up again when I got weary. I would not be here without them.

Ilya Kisil
March, 2021

Publications

The methods and results developed over the course of this research work have led to the following publications (published, submitted for publication, or in preparation for submission):

Published Peer-Reviewed Conference Proceedings

- [C1] **I. Kisil**, G. G. Calvi and D. P. Mandic, “Common and individual feature extraction using tensor decompositions: A remedy for the curse of dimensionality?”, *In Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6299–6303, 2018.
- [C2] **I. Kisil**, A. Moniri and D. P. Mandic, “Tensor ensemble learning for multi-dimensional data”, *In Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1358–1362, 2018.
- [C3] G. G. Calvi, **I. Kisil** and D. P. Mandic, “Feature fusion via tensor network summation”, *In Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, pp. 2623–2627, 2018.
- [C4] A. Moniri, **I. Kisil**, A. G. Constantinides and D. P. Mandic, “Refreshing DSP courses through biopresence in the curriculum: A successful paradigm”, *In Proceedings of the 23rd IEEE International Conference on Digital Signal Processing (DSP)*, pp. 1–4, 2018.

Published Peer-Reviewed Journals

- [J1] A. E. Stott, B. Scalzo Dees, **I. Kisil** and D. P. Mandic, “A class of multi-dimensional NIPALS algorithms for quaternion and tensor partial least squares regression”, *Signal Processing*, pp. 316–327, 2019.
- [J2] L. Stankovic, D. P. Mandic, M. Dakovic, **I. Kisil**, E. Sejdic and A. G. Constantinides, “Understanding the basis of graph signal processing via an intuitive example-driven approach”, *IEEE Signal Processing Magazine*, pp. 133–145, 2019.
- [J3] L. Stankovic, D. P. Mandic, M. Dakovic and **I. Kisil**, “Demystifying the coherence index in compressive sensing”, *IEEE Signal Processing Magazine*, pp. 152–162, 2020.

Accepted Journals

- [AJ1] **I. Kisil**, G. G. Calvi, B. Scalzo Dees and D. P. Mandic, “HOTTBOX: Higher Order Tensors ToolBOX”, *Accepted to Journal of Machine Learning Research*, pp. TBA, 2021.

Submitted Journals

- [SJ1] **I. Kisil**, G. G. Calvi, K. Konstantinidis, Y. L. Xu and D. P. Mandic, “Accelerating tensor contraction products via tensor train decomposition”, *Submitted to IEEE Signal Processing Magazine*, 2021.

Nomenclature

- [SJ2] S. Poushpas, **I. Kisil**, P. Normahani, B. Szubert, D. P. Mandic and U. Jaffer, “Tensor decompositions and machine learning for the detection of arteriovenous fistula stenosis: An initial evaluation”, *Submitted to Journal of Ultrasound in Medicine*, 2021.

Journals in Preparation

- [PJ1] **I. Kisil**, K. Konstantinidis, Y. L. Xu and D. P. Mandic, “Recursive Tucker decomposition for streaming Big Data”, *In Preparation for IEEE Signal Processing Magazine*, 2021.

Published Book Chapters

- [B1] **I. Kisil**, G. G. Calvi, B. Scalzo Dees and D. P. Mandic, “Tensor decompositions and practical applications: A hands-on tutorial”, *Recent Trends in Learning From Data*, pp. 69–97, 2020.

Invited Lectures

- [L1] **I. Kisil**, G. G. Calvi and D. P. Mandic, “Tensor decompositions for big data analytics: Trends and applications”, *International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019.
- [L2] **I. Kisil**, G. G. Calvi and D. P. Mandic, “Tensor decompositions and applications: Blessing of dimensionality”, *INNS Big Data and Deep Learning Workshop*, Sestri Levante, Italy, 2019.

Chapter 1

Introduction

Information is the oil of the 21st century,
and analytics is the combustion engine

Peter Sondergaard

1.1 Perspective

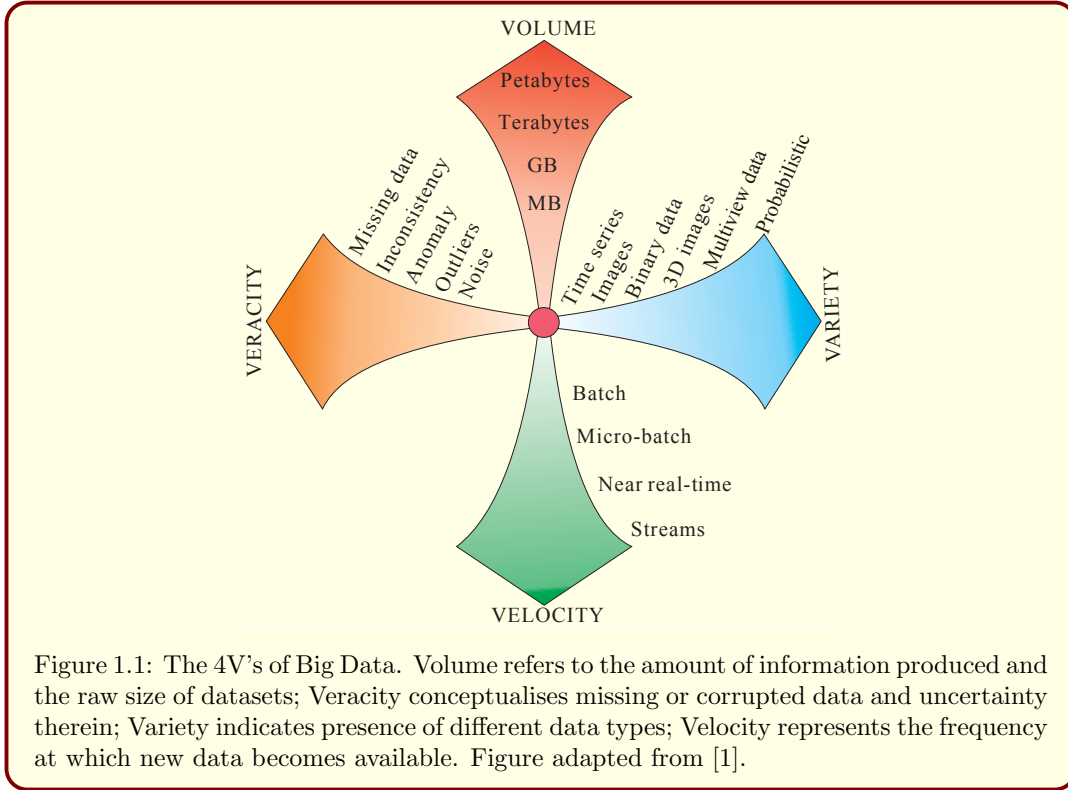
The ever-increasing amount of accessible data has posed unprecedented Data Analytics challenges, while at the same time it has provided novel opportunities to explore events and natural happenings yet requiring full explanation. Such data falls into the category of what is referred to within the Signal Processing and Machine Learning community as *Big Data*. Such datasets are characterised by the so-called “4V’s of Big Data”: Volume, Veracity, Variety, and Velocity, outlined in Fig. 1.1. Thus, Big Data is not necessarily just a synonym for exceedingly large raw size; indeed, contemporary datasets are far from “simple”, and analysing them is a non-trivial task which highlights the limitations of standard flat-view matrix models and underlines the necessity for sophisticated techniques able to assess the information within.

Tensors are higher order generalisations of matrices and vectors, which can represent data in a form of multi-dimensional array, i.e. through indexing by an arbitrary number of indices. From the Latin word *tendere*, i.e. “to stretch”, a tensor is a geometrical object initially proposed in mathematics and physics as an extension of vectors and matrices to higher dimensions [2]. Such representation of data brings up a whole host of new opportunities but also calls for new methods to seek for the otherwise undiscoverable inherent latent dependencies and intrinsic structures. Several modern-day data analysis tools started to employ N-dimensional arrays in their original form. These have been shown to offer greater flexibility in the choice of constraints matching data properties, hence providing more powerful schemes for performing extraction of latent components present in data [3].

Tensors were originally conceived in the theoretical fields of physics and mathematics since the early 20th century, when American mathematician Frank Hitchcock introduced the first principles of tensor decompositions (TDs) [4], which are used to overcome the early instances of the “*Curse of Dimensionality*” [5], a term coined by Bellman in 1961, to indicate that the number of parameters needed to estimate an arbitrary function grows exponentially with the number of parameters of the function itself. As in standard linear algebra matrix decompositions can be effectively used to compress matrices by obtaining low-rank representations, the concept of compression of multi-dimensional data through TDs can be intuitively explained as follows [6]. Consider an N -variate function, $f : \mathbb{R}^N \mapsto \mathbb{R}$, defined as,

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_N) \tag{1.1}$$

1. Introduction



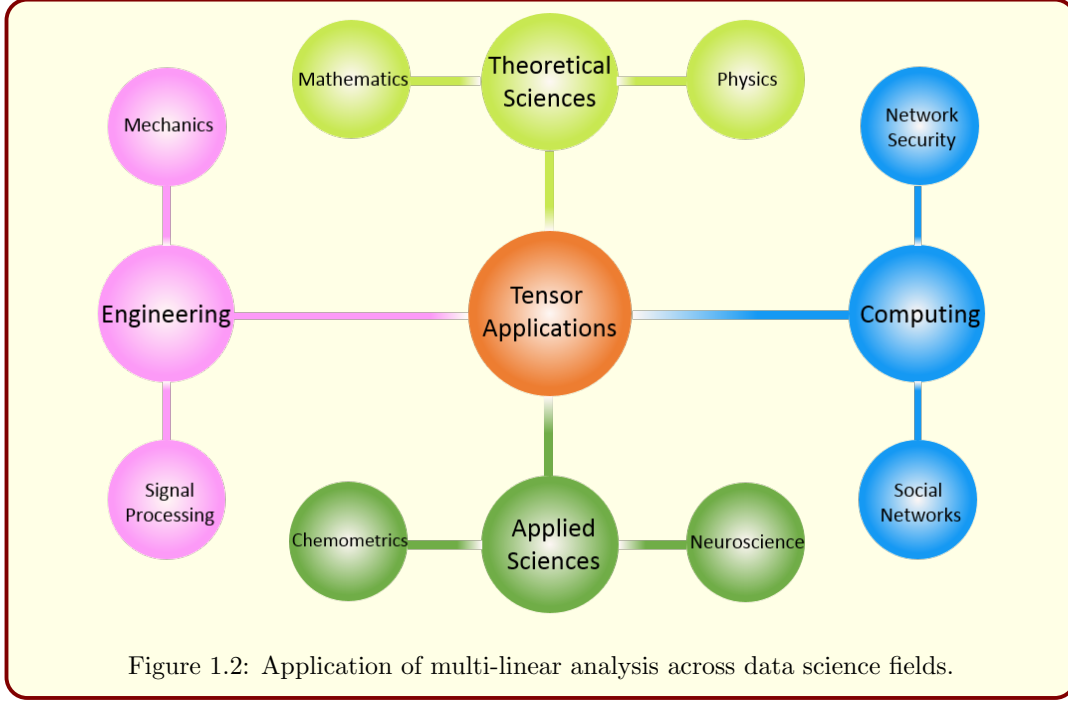
Suppose f is to be represented by a product of individual functions; in the simplest case it can be approximated by

$$f(x_1, x_2, \dots, x_N) \approx f^{(1)}(x_1)f^{(2)}(x_2) \dots f^{(N)}(x_N) \quad (1.2)$$

Unfortunately, the case where all variables are separable as in (1.2) is rare in practice. Instead, we can employ the concept of TDs which is considerably more flexible, as it relies on only partial separability of the variables of a higher-order tensor [1].

In multi-way analysis, tensor formation is a key step that defines interpretability of the obtained results. There are different approaches to *tensorisation* that can be taken prior to further actions. These include:

- Rearrangement of lower-dimensional data, commonly referred to as *folding*, where large scale matrices or vectors can be readily reshaped into a tensor. Such a procedure is extremely beneficial when underlying information exhibits, for instance, Hankel, Toeplitz or Löwner structure [7, 8, 9].
- Mathematical construction of a tensor, which implies that its slices are generated through transform-domain methods and are then stacked together across additional dimension. This can be achieved, for instance, by a time-frequency representation of a given signal via Short Time Fourier Transform (STFT), Gabor countourlet or wavelet transform [10, 11].
- Experimental design, which assumes a full control over the data collection process. Hence, this paradigm is highly suitable for any application that deals with high variety of data. In particular, such an approach can be employed when analysing information obtained during clinical trials where biomedical data, such as recordings of brain activity, are often collected over multiple patients which participated in multiple activities. This allows us



to organise data into a 4-th order tensor with $Time \times Channel \times Patient \times Trial$ modes [12].

- Natural formation, a classical example is an RGB color images that is represented separately in three base colors and stacked together as a set of slices along the third dimension.

1.2 Development and Adoption of Tensor Decompositions

Over the past decade, we have seen a rapid growth of the amount of data being collected on a day-to-day basis, storage capabilities and processing power, which is readily available through either local or cloud computing to users even outside of high-tech industry and academia. The multi-way analysis community has responded with numerous materials that help tackle and mitigate the “Curse of Dimensionality”, ranging from fundamental principles behind TDs [4] to more advanced and sophisticated techniques such as Tensor Networks (TNs) [1, 13]. Tensor decompositions are naturally suited to model and analyze large-scale measurements whose properties simultaneously evolve in time and space. Thus, they are perfectly equipped to maintain joint spatial, temporal, and spectral information among the data dimensions and have the ability to incorporate diverse constrained optimisation criteria (often having physical meaning), specifically tailored for every particular facet of the data tensor. This provides with much more freedom for designing a set of suitable constrained models, while at the same time maintaining natural links between the various extracted factors or latent (hidden) variables.

The first to propose the idea of a polyadic form of a tensor was Hitchcock in 1927 [14], by expressing a tensor as the sum of a finite number of rank-1 tensors. In 1944, Cattell proposed having multiple axes for analysis [15], but it was not until the 1970s that it became popular thanks to Carroll and Chang [16], after they applied it within the psychometrics community under the name of CANonical DECOMPosition (CANDECOMP), and to Harshman who named it the PARAllel FACtor model (PARAFAC) [17]. The model is now known as the Canonical Polyadic Decomposition (CPD). Variants of this factorisation model have subsequently been

1. Introduction

employed in Chemometrics, food industries, and social sciences [18, 19]. Another prominent member of the multi-linear analysis is the Tucker Decomposition (TKD) which was first introduced by Ledyard Tucker in 1963 [20, 21, 22]. A distinctive feature of this approach is that it is capable of modelling all possible combinations between the so-extracted components. Notably, this is achieved without imposing any assumptions on the underlying data which provides us with endless freedom in design at the cost of the uniqueness uncertainty. Around 2000, the scientific community came to the realisation that the TKD can be represented as the Higher-Order Singular Value Decomposition (HOSVD), a generalisation of the well-known matrix SVD [23, 24], which opened new avenues for its utilisation in applied mathematics and scientific computing in high dimensions [25]. This, combined with the Tensor-Train (TT) decomposition introduced by Oseledets in 2011 [26], made it possible for tensors to find applications in a variety of disciplines, such as wireless communications [27], audio and image processing [28, 29], graph analysis [30], and machine learning to name a few [31, 32].

Following the original success of TDs in applications to Chemometrics and Psychometrics, the benefits of multi-way analysis became apparent. With the technical progress of storage, together with the advances in computational power with the theoretical achievements, a whole new range of opportunities emerged for dealing with the generated and consumed data at an unprecedented high scale. In turn, this prompted a wider adoption of TDs in several other less mathematically minded fields where previously it was not feasible to employ them due to the sheer amount of available information. Some of those are as follows:

- The brain is by far one of the most complex systems while mysteries behind its information processing mechanisms are yet to be explored and fully understood. Electroencephalography (EEG) and Functional Magnetic Resonance Imaging (fMRI) are amongst those tools researchers use for the exploration of brain activity during certain mental or unconscious tasks. While it is particularly challenging to extract latent components or identify abnormal activities and patterns, there is no doubt that the gathered information exhibits a rich source of multi-way structure. Such tensor models quickly arrive at even four or five dimensions, for example, $Frequency \times Channel \times Time \times Subject \times Condition$, when attempting to analyse episodes of epileptic seizures or trying to detect brain regions of a patient who suffers from various neurodegenerative diseases such as Alzheimer's and Parkinsonism [33, 34].
- With the recent advances in sensor technologies, it became possible to monitor environmental and ecological conditions across vast regions of either underground spaces or overground coastlines. This is achieved through placement of hundreds of thousands of sensors at a considerable distance away from each other. The collected measurements usually have complex multivariate structure, often difficult to analyse and correlate with underlying spatiotemporal variations in such data. As a result, it became increasingly difficult to summarise an event or detect abnormal fluctuations of biological parameters solely based on conventional two-dimensional algorithms. Thus, the formation of multi-dimensional arrays of $Variables \times SamplingSite \times SamplingTime \times Season$ became a popular way to tackle problems associated with monitoring indoor pollution levels or exploring ground-water hydrochemistry [35, 36].
- Ever since the widespread growth of Internet technologies, computer based systems were constantly under intrusion threat with severe implications on network security. Conventional two-dimensional approaches to this problem are under serious scrutiny, imposed by extremely dynamic parameters of Internet traffic, where it is required to keep track of and jointly analyse information related to source and destination of thousands of requests per second, together with identifying and filtering out any malicious activity. Therefore, TDs have become a physically meaningful tool for analysis of tensor data in $SourceIP \times TargetIP \times Port \times Time$ format [37, 38].

1.3 Problem Statement

Technology across multiple fields now demands sophisticated models capable of analysing information whereby multiple high-dimensional characteristics are buried deep down like a needle in a haystack. However, the data organised in a matrix form is governed by only one degree of freedom, and thus, it is becoming harder and harder for conventional two-dimensional frameworks to meet expectations. Tensors are higher order generalisations of matrices and vectors, which can represent data in a form of a multi-way array. Due to their inherently flexible structure, tensors are perfectly equipped to maintain and describe not only spatial, temporal and spectral information among the data dimensions but also virtually any other imaginable characteristic. In turn, this opens a wide avenue of opportunities to integrate diverse and physically meaningful constrained optimisation principles, to be individually tailored to every aspect of the original data as well as to establish implicit links or correlation between latent features of different dimensions.

Owing to the inherent flexibility and horizontal scalability of multi-linear factorisations, over the last couple of decades the tensor decompositions have been adopted and successfully utilised in diverse range of disciplines, from very theoretical ones, such as physics and numerical analysis [9, 23, 39], to the more applied areas, such as face recognition, separation of unknown sources of brain activity, cross-language information retrieval [3, 40, 41, 42] to mention but a few. The main aim of the developed approaches has been to project high dimensional data onto a lower subspace in which latent components can be discovered more easily in order to provide new understanding of intrinsic data properties. Nevertheless, existing solutions often overlook the importance of physical meaning associated with the underlying multilinear operations, especially while attempting joint factorisation of multi-block datasets.

It comes as no surprise that preservation of the steady adoption rate of tensor decompositions within new disciplines is not feasible without reliable software libraries that can be shared amongst researchers. This call for robust programming methods has spurred the multi-way community to respond with the development of open source packages [43, 44, 45, 46] in order to mitigate, for a generally knowledgeable user, most of the technicalities behind the implementation of standard tensor decompositions and multi-linear operations, alongside the corresponding statistical methods and data analysis techniques. However, existing publicly available libraries are largely designed as general purpose toolboxes primarily comprising fundamental procedures, namely the Canonical Polyadic, Tucker, and Tensor Train decompositions while focusing on the efficient performance of such methods through the use of hardware acceleration. As a consequence, these tools treat tensors simply as N-dimensional arrays accompanied with multiple indices used to locate and define entries of this data structure. Moreover, existing software rarely extends its scope to emerging algorithms for classification, regression, and data fusion, that have been specifically tailored to operate with multi-dimensional data.

In light of these directions of development in the field of multi-dimensional decompositions and applications, together with the associated challenges, the focus of this thesis is to strengthen understanding of the fundamental tensor operations through a strong underpinning notion of interpretability within the intrinsic connections. In particular, we here consider the problem of a joint analysis of multi-block data with the aim of extracting physically meaningful information during data preprocessing steps, thus, mitigating the black-box nature of machine learning problems. A typical example of multi-block data is a set of recordings of the same phenomenon but under distinct experimental setups. For instance, such a structure can be observed in multiple images of objects taken under different lighting conditions and from different angles, or in a collection of medical recordings, such as brain activity, whereby patients perform various mental tasks. Intuitively, the so obtained ensemble of information contains some common and some individual patterns, and machine learning tasks would benefit from exploiting only either common features (for clustering) or individual features (for identification), both of much lower cardinality than the original data.

In the hope of demystifying tensor decompositions, and in this way, expediting their adoption in even more areas of data science and signal processing, we also address the issues of (i) diversity

within computational methods readily available to researchers; (ii) popularisation of multi-linear analysis and applications while lowering the entry barrier into the field for curious enthusiasts. To this end, we lay the foundation for a self-contained and easy to use piece of software that takes existing solutions and frameworks a step further in the demanding realm of data modelling by:

- Utilising tensors not only as N-dimensional arrays, but also exploiting complementary meta-information about the data a tensor represents;
- Offering scalable visualisation tools, at every stage of the experiments, to facilitate ease of analysis, multi-disciplinary efforts or educational purposes;
- Being focused on unifying the machine learning approaches which are specific to inherently multi-dimensional data;
- Allowing for seamless integration with popular data science packages for data wrangling and feature extraction.

This thesis aims to achieve the milestones from different perspectives, which include formulation of new algorithms, investigating their learning behaviour and computational characteristics, giving new perspectives and insights to some well-established multi-dimensional operations and decompositions, testing the proposed algorithms through real-world case studies, and finally by providing illustrative and practical examples in the form of self-explanatory diagrams and short code snippets.

1.4 Thesis Contribution and Organisation

Fig. 1.3 shows an overview of structure of this thesis. Concise summaries of the main contributions of each chapter, along with the relevant publications, are presented below.

- **Chapter 2.** Prerequisites: Multilinear Algebra.

The aim of this chapter is to provide the required background on multilinear tensor algebra for the analysis presented in this thesis. The reader will become familiarised with the fundamental operations attributed to the multiway arrays as well as becoming accustomed to various tensorisation techniques that allow shifting the analysis from “flat-view” matrix data representation to multi-dimensional perspectives. This motivates us to introduce standard tensor decomposition models, namely the Canonical Polyadic, Tucker, and Tensor Train decompositions. We also establish the basis for Tensor Network representation that abstracts the cumbersome notation but at the same time allows us to describe relations between tensors in a declarative and intuitive manner.

- **Chapter 3.** Feature Extraction and Data Fusion.

In pursuit of exploiting the full potential of the underlying structure of tensors and the associated feature localisation, this chapter proposes new algorithms for joint analysis of multi-dimensional data. First, we address a problem of separation individual from common information inherent to the multiblock data. This is achieved through a natural link between the intrinsic properties of the outer product and the components obtained through tensor decompositions. By exploring the fundamental meaning behind the so-obtained factor matrices, we reach to a conclusion that a non-negativity constraint is a necessary criteria for successful feature extraction in order to preserve physical meaning amongst the hindered relations. Next, we demonstrate how even seemingly elementary operations can be utilised as building blocks for a new framework of tensor network summation whereby interconnections among the cores in a tensor network representation are used to describe the coupling data structures of the original tensors. In turn, such an approach to data analysis opens up new avenues for feature fusion suitable for machine learning applications.

Relevant publications:

1. Introduction

- **I. Kisil**, G. G. Calvi and D. P. Mandic, “Common and individual feature extraction using tensor decompositions: A remedy for the curse of dimensionality?”, *In Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6299–6303, 2018.
- G. G. Calvi, **I. Kisil** and D. P. Mandic, “Feature fusion via tensor network summation”, *In Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, pp. 2623–2627, 2018.

- **Chapter 4.** Tensor Decompositions and Wisdom of the Crowd.

In this chapter, we take a fresh look at the Ensemble Learning, a well established technique exploited for increasing the performance of the machine learning algorithms. In particular, we address a common problem of correlated errors of the base classifiers, that significantly affects the accuracy of the ensemble classifier at the majority vote step. This is achieved by virtue of tensor decompositions, whereby the proposed method, referred to as tensor ensemble learning (TEL), decomposes every input data sample into multiple factors which allows for flexibility in the choice of multiple learning algorithms in order to improve test performance. Thus, TEL generalises classic flat-view ensemble learning to multi-dimensional tensor-valued data while taking advantage of natural compression associated with multi-way data structures, thus allowing for parallel processing to alleviate computation burden in big data applications. The proposed framework is first verified through the application of Higher Order Singular Value Decomposition to the classification of images from the ETH-80 dataset. Next, we assess the feasibility of multi-linear learning analysis for cardiovascular diagnosis, where a blood flow of 22 clinical participants is inspected through a combination of Canonical Polyadic Decomposition and TEL approach and compared to results obtained by a qualified vascular scientist during manual examination of B-mode ultrasound recording.

Relevant publications:

- **I. Kisil**, A. Moniri and D. P. Mandic, “Tensor ensemble learning for multi-dimensional data”, *In Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1358–1362, 2018.
- S. Poushpas, **I. Kisil**, P. Normahani, B. Szubert, D. P. Mandic and U. Jaffer, “Tensor decompositions and machine learning for the detection of arteriovenous fistula stenosis: An initial evaluation”, *Submitted to Journal of Ultrasound in Medicine*, 2021.

- **Chapter 5.** Computational Efficiency and Practical Implementations.

The aim of this chapter is to develop tools capable of coping with the sheer complexity of modern datasets as the natural high dimensionality of tensors implies that the applications of common algorithms on data in the raw tensor format may be intractable, owing to the Curse of Dimensionality. We start with a deep dive into the graphical representation of Tensor Networks (TNs). Such an approach has been proven to be effective in describing multi-linear operations in an intuitive way capable of elucidating even on the most basic procedures and has been widely adopted within the research community. By gaining deeper insight and a better understanding of the field, we devise a new algorithm, the Tensor Train Contraction Product (TTCP), that efficiently performs contraction of two multi-dimensional arrays while being impartial to their sizes or the number of dimensions. Next, we tackle a shortcoming of computation of the Tucker decomposition that requires the entire input data to be known beforehand which is not the case for on-line streaming scenarios and, in addition, can quickly become prohibitively expensive for large volume tensor data. We propose a novel approach based on recursive updates of the factor matrices through estimation of mode-wise eigenvalues and eigenvectors and the perturbation of the associated mode-wise covariance matrices. Thus, a distinctive feature

of the introduced framework is that computation only occurs upon the arrival of a new tensor-variate data sample. The efficacy of the proposed algorithms opens new avenues for the next generation of real-time streaming technology and applications. Finally, this chapter presents an open source toolbox for multi-linear analysis, namely the Higher Order Tensors ToolBOX (HOTTBOX), which combines practical implementations of tensor methods for feature extraction, regression, non-linear classification, and efficient representations of multi-dimensional data. The HOTTBOX comprises algorithms ranging from standard multi-way operations, for example, Higher Order Orthogonal Iteration (HOOI) and Tensor Train (TT) decomposition, through to advanced multi-linear algorithms, Randomised Canonical Polyadic Decomposition (RCPD), and Coupled Matrix and Tensor factorisation (CMTF) to name a few, as well as sophisticated algorithms for generalised multi-linear classification and data fusion such as Support Tensor Machine (STM) and Tensor Ensemble Learning (TEL).

Relevant publications and outreach activity:

- **I. Kisil**, G. G. Calvi, K. Konstantinidis, Y. L. Xu and D. P. Mandic, “Accelerating tensor contraction products via tensor train decomposition”, *Submitted to IEEE Signal Processing Magazine*, 2021.
 - **I. Kisil**, K. Konstantinidis, Y. L. Xu and D. P. Mandic, “Recursive Tucker decomposition for streaming Big Data”, *In Preparation for IEEE Signal Processing Magazine*, 2021.
 - **I. Kisil**, G. G. Calvi, B. Scalzo Dees and D. P. Mandic, “HOTTBOX: Higher Order Tensors ToolBOX”, *Accepted to Journal of Machine Learning Research*, pp. TBA, 2021.
 - **I. Kisil**, G. G. Calvi, B. Scalzo Dees and D. P. Mandic, “Tensor decompositions and practical applications: A hands-on tutorial”, *Recent Trends in Learning From Data*, pp. 69–97, 2020.
 - **I. Kisil**, G. G. Calvi and D. P. Mandic, “Tensor decompositions for big data analytics: Trends and applications”, *International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019.
 - **I. Kisil**, G. G. Calvi and D. P. Mandic, “Tensor decompositions and applications: Blessing of dimensionality”, *INNS Big Data and Deep Learning Workshop*, Sestri Levante, Italy, 2019.
- **Chapter 6.** Conclusions and Future Work.
- Concluding remarks drawn from the thesis and ideas for future works are outlined in this chapter.

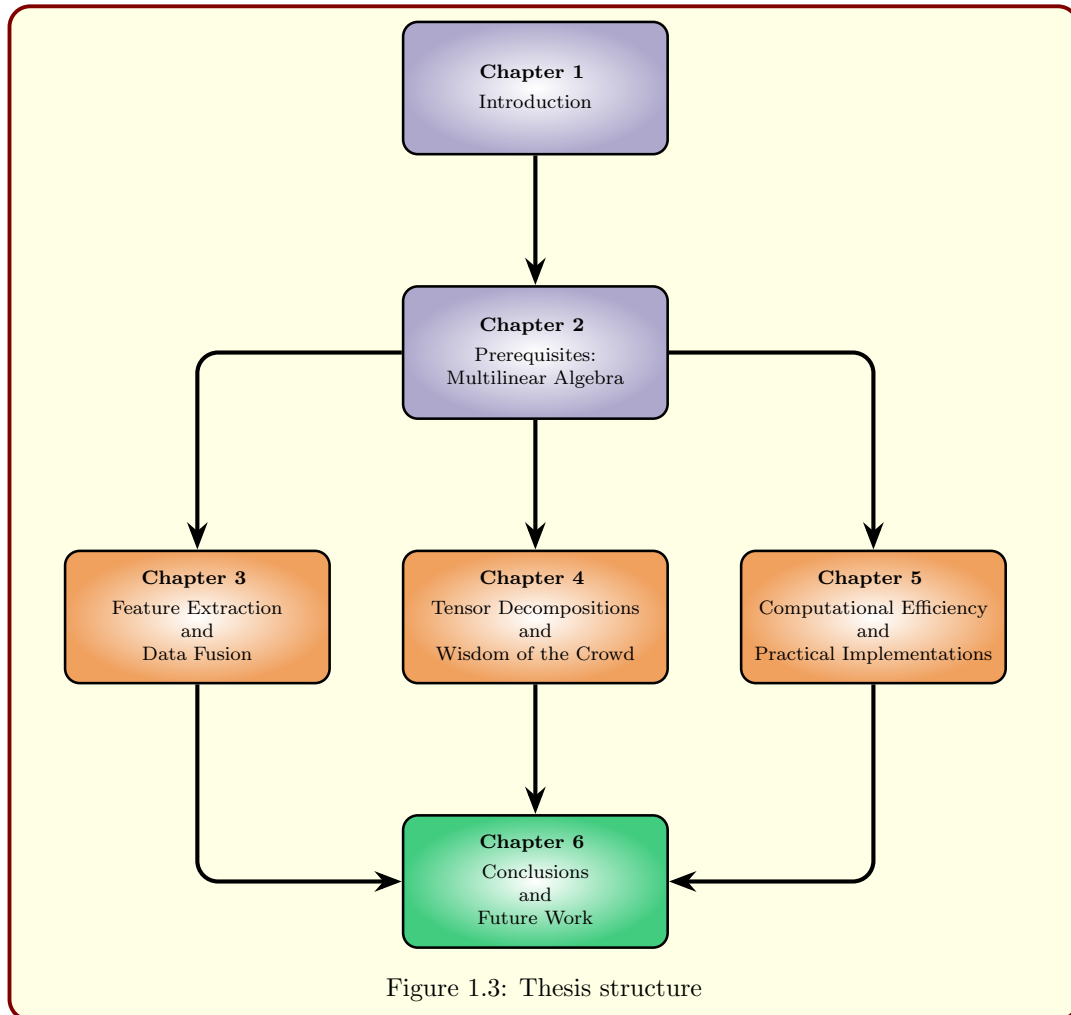


Figure 1.3: Thesis structure

Chapter 2

Prerequisites: Multilinear Algebra

Innovation involves no new technologies. Rather they consist of components built around proven technologies that are put together in novel architectures, offering the consumer a set of attributes never before available.

Clayton M. Christensen

Chapter Overview

This chapter presents the foundation of multi-linear analysis, a basis for the thesis developments. We begin by introducing the main definitions of multi-dimensional arrays (tensors), followed by core operations associated with them. Next, we establish a basis for Tensor Network representation that abstracts cumbersome notation but at the same time allows us to describe relations between tensors in a declarative and intuitive manner. Immediately after, we take a deep dive into various tensorisation techniques essential for the transition from the standard “flat-view” to the multiway representation of data. Finally, a comprehensive overview of the fundamental tensor decompositions is provided, namely the Canonical Polyadic, Tucker and Tensor Train decompositions, the associated efficient representations of data and numerical methods to obtain them.

2.1 Notation and Multilinear Operations

2.1.1 Basic definitions and conventions

First, we shall introduce fundamental definitions and relevant terminology that will be extensively used throughout the remainder of this thesis.

Definition 2.1. A **tensor** is a multi-dimensional array of data where by the position of each entry is defined by N indices. A tensor is denoted as underlined boldface capital letter, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$.

Definition 2.2. A **mode** of a tensor is a particular dimension within a given array of data. For instance, the n -th **mode** corresponds to the n -th index of the tensor. Oftentimes, it helps to think about modes of a tensor as certain properties or characteristics of data at hand.

2. Prerequisites: Multilinear Algebra

Definition 2.3. An **order** of a tensor is the total number of its dimensions or modes. Thus, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ is a tensor of order N .

Definition 2.4. A **superdiagonal** of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ is the collection of entries $\underline{\mathbf{X}}[i_1, i_2, \dots, i_n]$, where $i_1 = i_2 = \dots = i_n$.

Definition 2.5. A **sub-tensor** is a collection of original entries obtained by fixing at least one of the indices. Note that for a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, we do not differentiate between $\underline{\mathbf{Y}} = \underline{\mathbf{X}}[:, :, \dots, I_N]$ and $\underline{\mathbf{Z}} = \underline{\mathbf{X}}[I_1, \dots, :, :]$ until stated otherwise.

Definition 2.6. A **block tensor** is a tensor that is composed of sub-tensors of the same order but not necessarily of the same dimensionality; these are called **blocks** and located on the superdiagonal.

Definition 2.7. A **slice** is a sub-structure of a tensor obtained by fixing all but two of its indices. For example, to retrieve a frontal slice of a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, all but first two indices have to be fixed, $\underline{\mathbf{Y}} = \underline{\mathbf{X}}[:, :, i_3, \dots, i_N]$. Hence, a slice is equivalent to a matrix.

Definition 2.8. A **fiber** is a sub-structure of a tensor obtained by fixing all but one index, i.e. $\underline{\mathbf{y}} = \underline{\mathbf{X}}[:, i_2, \dots, i_N]$. Thus, the notation for a fiber is tightly coupled with a mode of tensor and property associated with it.

Definition 2.9. A **mode- n unfolding** of a tensor is a procedure of mapping the elements from a multi-dimensional array to two-dimensional array. Conventionally, such procedure is associated with stacking mode- n fibers as column vectors of the resulting matrix. For instance, mode-1 unfolding of $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is represented as $\underline{\mathbf{X}}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3 \dots I_N}$

$$\underline{\mathbf{X}}_{(1)} \left[\overline{i_1, i_2 i_3 \dots i_N} \right] = \underline{\mathbf{X}}[i_1, i_2, \dots, i_N] \quad (2.1)$$

Note that the overlined subscripts refer to linear indexing (or Little-Endian) [6]:

$$\begin{aligned} \overline{i_1 i_2 \dots i_N} &= 1 + \sum_{n=1}^N \left[(i_n - 1) \prod_{n'=1}^{n-1} I_{n'} \right] \\ &= 1 + i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \dots + (i_N - 1)I_1 \dots I_{N-1} \end{aligned} \quad (2.2)$$

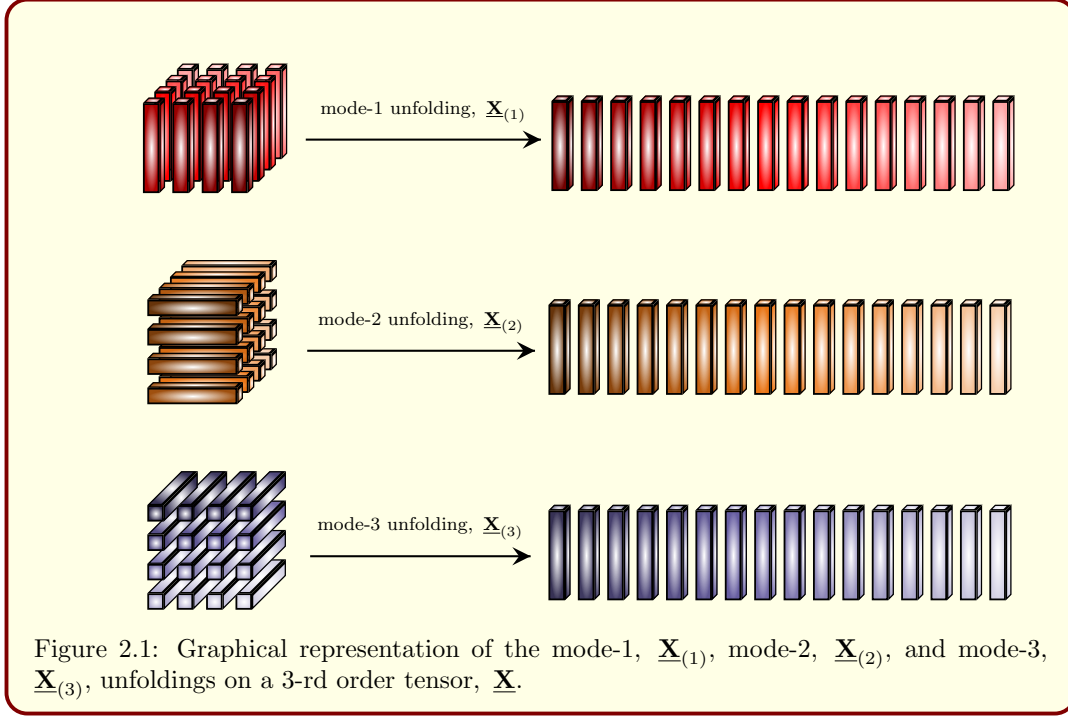
In order to facilitate the understanding of the concept of unfolding, consider the following example for a 3-order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{4 \times 2 \times 3}$, which is composed of frontal slices:

$$\underline{\mathbf{X}}[:, :, 1] = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix} \quad \underline{\mathbf{X}}[:, :, 2] = \begin{bmatrix} 9 & 13 \\ 10 & 14 \\ 11 & 15 \\ 12 & 16 \end{bmatrix} \quad \underline{\mathbf{X}}[:, :, 3] = \begin{bmatrix} 17 & 21 \\ 18 & 22 \\ 19 & 23 \\ 20 & 24 \end{bmatrix} \quad (2.3)$$

Then, all possible unfoldings of $\underline{\mathbf{X}}$ take form:

$$\begin{aligned} \underline{\mathbf{X}}_{(1)} &= \begin{bmatrix} 1 & 5 & 9 & 13 & 17 & 21 \\ 2 & 6 & 10 & 14 & 18 & 22 \\ 3 & 7 & 11 & 15 & 19 & 23 \\ 4 & 8 & 12 & 16 & 20 & 24 \end{bmatrix} \\ \underline{\mathbf{X}}_{(2)} &= \begin{bmatrix} 1 & 2 & 3 & 4 & 9 & 10 & 11 & 12 & 17 & 18 & 19 & 20 \\ 5 & 6 & 7 & 8 & 13 & 14 & 15 & 16 & 21 & 22 & 23 & 24 \end{bmatrix} \\ \underline{\mathbf{X}}_{(3)} &= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix} \end{aligned} \quad (2.4)$$

A graphical illustration of this procedure is provided on Fig. 2.1.



Remark 1. Unfolding a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, along different modes is not equivalent to permutation of data, even for the most simplistic case of $N = 3$:

$$\underline{\mathbf{X}}_{(n)} \neq \mathbf{P} \underline{\mathbf{X}}_{(m)} \quad (2.5)$$

2.1.2 Fundamental operations

Tensors have an incredible potential both from a computational and theoretical point of view. The possibility of analysing data structure from different angles is owing to their multi-linear structure. There are several main operations and techniques that facilitate the effective manipulation of incorporated data. A list of such operators and their notation is summarised in Table 2.2.

Kronecker product

The *Kronecker product* of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B}$ and the result is defined by the following relation:

$$\begin{aligned} \mathbf{C} = \mathbf{A} \otimes \mathbf{B} &= \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \\ &= [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_1 \otimes \mathbf{b}_2 \quad \mathbf{a}_1 \otimes \mathbf{b}_3 \quad \cdots \quad \mathbf{a}_J \otimes \mathbf{b}_{L-1} \quad \mathbf{a}_J \otimes \mathbf{b}_L] \end{aligned} \quad (2.6)$$

Here, $\mathbf{C} \in \mathbb{R}^{IK \times JL}$ while general relation between entries of $\mathbf{C}$ and $\mathbf{A}$, $\mathbf{B}$ is represented by:

$$c_{(i-1)K+k;(j-1)L+l} = a_{ij}b_{kl} \quad (2.7)$$

Thus, the Kronecker product considers all possible pair-wise combinations of initial matrix entries.

2. Prerequisites: Multilinear Algebra

Table 2.1: Basic Tensor and Matrix Notation

$\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Tensor of order N of size $I_1 \times I_2 \times \cdots \times I_N$
$\underline{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Tensor Network representation of order N tensor of size $I_1 \times I_2 \times \cdots \times I_N$
$x_{i_1 i_2 \cdots i_N} = \underline{\mathbf{X}}[i_1, i_2, \dots, i_N]$	Element of $\underline{\mathbf{X}}$ at position $(i_1, i_2, \dots, i_N)$
$\underline{\mathbf{X}}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$	Mode- n unfolding of a tensor $\underline{\mathbf{X}}$
$\mathbf{A}^{(n)} \in \mathbb{R}^{J_n \times I_n}$	Factor matrices in tensor decompositions
$\underline{\mathbf{D}} \in \mathbb{R}^{I \times \cdots \times I}$	Super diagonal tensor within tensor decomposition
$\underline{\mathbf{G}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Core tensor within tensor decompositions
$\underline{\mathbf{G}}^{(n)}$	Core tensor within Tensor Train decomposition
$\mathbf{I}_N \in \mathbb{R}^{N \times N}$	Identity matrix of size $N \times N$
$\mathbf{1}_N \in \mathbb{R}^N$	Vector of ones of size N
$\text{vec}(\underline{\mathbf{X}}) = \mathbf{x} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$	Vectorisation of tensor $\underline{\mathbf{X}}$

Khatri-Rao product

The *Khatri-Rao product* operates with matrices that have the same number of columns. Often, it is referred to as the column-wise Kronecker product. For matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, it is denoted by $\mathbf{A} \odot \mathbf{B}$ and yields $\mathbf{C} \in \mathbb{R}^{IJ \times K}$ which is computed as:

$$\begin{aligned} \mathbf{C} = \mathbf{A} \odot \mathbf{B} &= \begin{bmatrix} a_{11}\mathbf{b}_1 & a_{12}\mathbf{b}_2 & \cdots & a_{1K}\mathbf{b}_K \\ a_{21}\mathbf{b}_1 & a_{22}\mathbf{b}_2 & \cdots & a_{2K}\mathbf{b}_K \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{b}_1 & a_{I2}\mathbf{b}_2 & \cdots & a_{IK}\mathbf{b}_K \end{bmatrix} \\ &= [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_2 \otimes \mathbf{b}_2 \quad \cdots \quad \mathbf{a}_K \otimes \mathbf{b}_K] \end{aligned} \quad (2.8)$$

Observe that when we perform such operation over inputs represented as vectors $\mathbf{a}$ and $\mathbf{b}$, then the Khatri-Rao and Kronecker products are equal:

$$\mathbf{a} \odot \mathbf{b} = \mathbf{a} \otimes \mathbf{b}. \quad (2.9)$$

Hadamard product

The *Hadamard product* is the elementwise matrix product, therefore the involved matrices have to be of the same size. It is employed in order to describe one-to-one relations between elements for two matrices, e.g. $\mathbf{A}$ and $\mathbf{B}$ both $\in \mathbb{R}^{I \times J}$. The Hadamard product is denoted as $\mathbf{A} \circledast \mathbf{B}$ and takes form:

$$\mathbf{C} = \mathbf{A} \circledast \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix} \quad (2.10)$$

2. Prerequisites: Multilinear Algebra

Table 2.2: Definition of Main Products	
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n \mathbf{B}$	Mode- n product of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and matrix $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$ yields $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times J_n \times I_{n+1} \times \dots \times I_N}$
$\underline{\mathbf{C}} = \llbracket \underline{\mathbf{A}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)} \rrbracket$	Representation of an original tensor through a full multilinear product $\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_1 \mathbf{B}^{(1)} \times_2 \dots \times_N \mathbf{B}^{(N)}$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \bar{\times}_n \mathbf{b}$	Mode- n product of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and vector $\mathbf{b} \in \mathbb{R}^{I_n}$ yields a tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N}$ of order $(N-1)$
$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_n^m \underline{\mathbf{B}}$	Contraction product of some tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \dots \times I_n \times \dots \times I_N}$ and tensor $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \dots \times J_m \times \dots \times J_M}$ along modes m and n respectively, yields a new tensor of higher order, i.e. $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N \times J_1 \times \dots \times J_{m-1} \times J_{m+1} \times \dots \times J_M}$
$\underline{\mathbf{A}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \dots \circ \mathbf{a}^{(N)}$	Outer product of vectors $\mathbf{a}^{(n)} \in \mathbb{R}^{I_n}, n = 1, \dots, N$, yields a rank-1 tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$
$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$	Kronecker product of $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ yields a new matrix $\mathbf{C} \in \mathbb{R}^{IK \times JL}$
$\mathbf{C} = \mathbf{A} \odot \mathbf{B}$	Khatri-Rao product of $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ yields a new matrix $\mathbf{C} \in \mathbb{R}^{IJ \times K}$
$\mathbf{C} = \mathbf{A} \circledast \mathbf{B}$	Hadamard product of $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{I \times J}$ yields a new matrix $\mathbf{C} \in \mathbb{R}^{I \times J}$

Outer product

The *outer product* is depicted on Fig. 2.2 while its element-wise definition is given below:

$$\underline{\mathbf{X}}[i_1, i_2, \dots, i_N] = \prod_{n=1}^N \mathbf{x}_{i_n}^{(n)} \quad (2.11)$$

Definition 2.10. If a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ can be expressed as an outer product of N vectors $\mathbf{x}^{(n)} \in \mathbb{R}^{I_n} \quad \forall n$, that is:

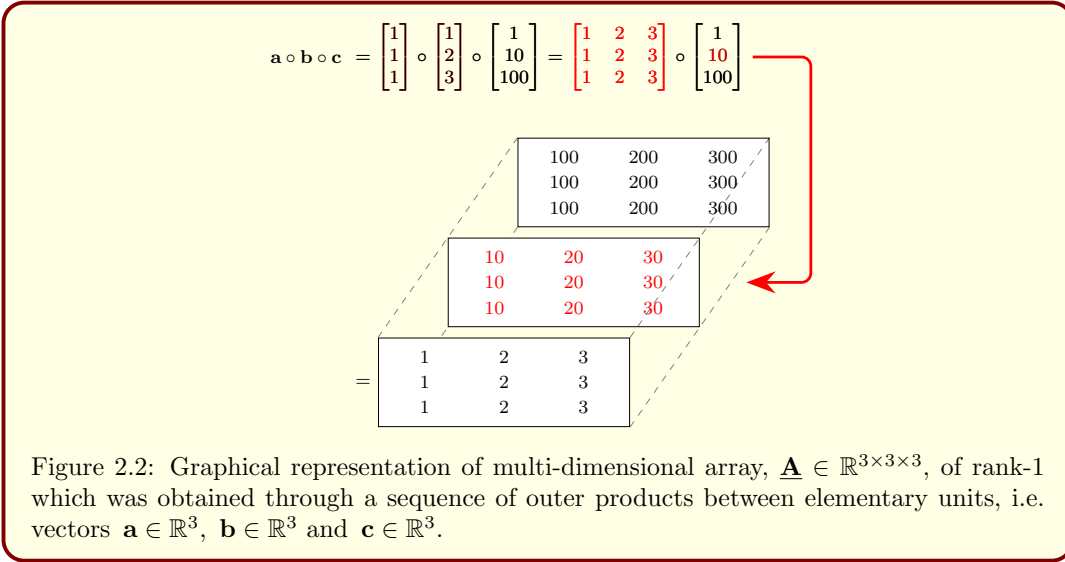
$$\underline{\mathbf{X}} = \mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \dots \circ \mathbf{x}^{(N)} \quad (2.12)$$

then such multi-dimensional array is said to be of rank-1.

Consequently, by analogy with linear algebra, a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, is said to be of rank R if it can be represented as a linear combination of R rank-1 tensors:

$$\begin{aligned} \underline{\mathbf{X}} &= \sum_{r=1}^R \underline{\mathbf{X}}_r \\ &= \sum_{r=1}^R \mathbf{x}_r^{(1)} \circ \mathbf{x}_r^{(2)} \circ \dots \circ \mathbf{x}_r^{(N)} \\ &= \sum_{r=1}^R \bigcirc_{n=1}^N \mathbf{x}_r^{(n)}. \end{aligned} \quad (2.13)$$

2. Prerequisites: Multilinear Algebra



Mode- n product

So far, we have introduced operations that form the basis of multi-linear relations, however, they take matrices as an input and produce another two-dimensional array. Now it is time to introduce the first and most prominent operation of multi-linear analysis, namely the *mode- n product*, which takes as inputs an order- N tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, and a matrix, $\mathbf{A} \in \mathbb{R}^{J \times I_n}$, to produce another tensor, $\underline{\mathbf{Y}}$, of the same order as the original tensor $\underline{\mathbf{X}}$. The operation is denoted by :

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_n \mathbf{A} \quad (2.14)$$

where $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N}$. The mode- n product is comprised of 3 consecutive steps:

$$\begin{aligned} \underline{\mathbf{X}} &\rightarrow \underline{\mathbf{X}}_{(n)} \\ \underline{\mathbf{Y}}_{(n)} &= \mathbf{A} \underline{\mathbf{X}}_{(n)} \\ \underline{\mathbf{Y}}_{(n)} &\rightarrow \underline{\mathbf{Y}} \end{aligned} \quad (2.15)$$

In other words, each mode- n fiber of $\underline{\mathbf{X}}$ is multiplied by the matrix $\mathbf{A}$ and in the element-wise form this is expressed as:

$$\underline{\mathbf{Y}}[i_1, \dots, i_{n-1}, j, i_{n+1}, \dots, i_N] = \sum_{i_n=1}^{I_n} a_{ji_n} \underline{\mathbf{X}}[i_1, \dots, i_n, \dots, i_N] \quad (2.16)$$

The mode- n product exhibits a couple of interesting properties one of which is the distributive law of sequential operations:

$$\underline{\mathbf{X}} \times_n \mathbf{A} \times_n \mathbf{B} = \underline{\mathbf{X}} \times_n (\mathbf{B}\mathbf{A}) \quad (2.17)$$

which can be readily verified through:

2. Prerequisites: Multilinear Algebra

$$\begin{aligned}
\underline{\mathbf{Y}} &= \underline{\mathbf{X}} \times_n \mathbf{A} \Leftrightarrow \underline{\mathbf{Y}}_{(n)} = \mathbf{A} \underline{\mathbf{X}}_{(n)} \\
\underline{\mathbf{Z}} &= \left(\underline{\mathbf{X}} \times_n \mathbf{A} \right) \times_n \mathbf{B} \Leftrightarrow \underline{\mathbf{Z}}_{(n)} = \mathbf{A} \underline{\mathbf{Y}}_{(n)} \\
\underline{\mathbf{Z}} &= \underline{\mathbf{X}} \times_n \mathbf{A} \times_n \mathbf{B} \Leftrightarrow \underline{\mathbf{Z}}_{(n)} = (\mathbf{B} \mathbf{A}) \underline{\mathbf{X}}_{(n)} \\
\underline{\mathbf{X}} \times_n \mathbf{A} \times_n \mathbf{B} &\Leftrightarrow \underline{\mathbf{X}} \times_n (\mathbf{B} \mathbf{A})
\end{aligned} \tag{2.18}$$

Consequently, this property implies that

$$\underline{\mathbf{X}} \times_n \mathbf{A} \times_n \mathbf{B} \neq \underline{\mathbf{X}} \times_n \mathbf{B} \times_n \mathbf{A}. \tag{2.19}$$

However, the order in which mode- n products are applied to a tensor $\underline{\mathbf{X}}$ does not matter, provided that they target different dimensions of $\underline{\mathbf{X}}$, that is:

$$\underline{\mathbf{X}} \times_m \mathbf{A} \times_n \mathbf{B} = \underline{\mathbf{X}} \times_n \mathbf{B} \times_m \mathbf{A}, \text{ for } m \neq n \tag{2.20}$$

Definition 2.11. *Tensor factorisation is a representation of a given order- N tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, through a sequence of some core tensor, $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times \dots \times J_N}$, and a collection of N factor matrices, $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, for instance:*

$$\begin{aligned}
\underline{\mathbf{X}} &= \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \dots \times_N \mathbf{A}^{(N)} \\
&= \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)} \rrbracket \\
&= \underline{\mathbf{G}} \times_{n=1}^N \mathbf{A}^{(n)}.
\end{aligned} \tag{2.21}$$

The operation in Eq. (2.21) can also be expressed using only two dimensional arrays in conjunction with the Kronecker product, which then takes the form:

$$\begin{aligned}
\underline{\mathbf{X}}_{(n)} &= \mathbf{A}^{(n)} \underline{\mathbf{G}}_{(n)} \left(\mathbf{A}^{(N)} \otimes \dots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \dots \otimes \mathbf{A}^{(1)} \right)^T \\
&= \mathbf{A}^{(n)} \underline{\mathbf{G}}_{(n)} \left(\bigotimes_{\substack{i=1 \\ i \neq n}}^N \mathbf{A}^{(i)} \right)^T \\
\text{vec}(\underline{\mathbf{X}}) &= \left(\bigotimes_{i=1}^N \mathbf{A}^{(i)} \right) \text{vec}(\underline{\mathbf{G}})
\end{aligned} \tag{2.22}$$

Definition 2.12. *The multi-linear rank of a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ is the N -tuple $(R_1, R_2, \dots, R_N)$ where R_n is the rank of a subspace spanned by mode- n fibers of $\underline{\mathbf{X}}$.*

$$mlrank(\underline{\mathbf{X}}) = mlrank \left(rank(\underline{\mathbf{X}}_{(1)}), rank(\underline{\mathbf{X}}_{(2)}), \dots, rank(\underline{\mathbf{X}}_{(N)}) \right) \tag{2.23}$$

Remark 2. *Note, that matrices, i.e. tensors of order 2, also have a notion of multi-linear rank, however, it utilises slightly different terminology, namely the column and row rank of a matrix. A remarkable difference between higher order tensors and matrices is that the values of $(R_1, R_2, \dots, R_N)$ can be different for $N \geq 3$.*

Contraction product

The *contraction product* is an operation similar to mode- n product, but it handles input consisting of multi-dimensional arrays that can be of different order. For instance, given an N -th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and another M -th order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \dots \times J_M}$, with common modes $I_n = J_m$, then their (n, m) -contraction denoted as, $\times_n^m$, yields a third tensor $\underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N \times J_1 \times \dots \times J_{m-1} \times J_{m+1} \times \dots \times J_M}$ of order $(N + M - 2)$, with entries

2. Prerequisites: Multilinear Algebra

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_n^m \underline{\mathbf{Y}}$$

$$z_{i_1, \dots, i_{n-1}, i_{n+1}, \dots, i_N, j_1, \dots, j_{m-1}, j_{m+1}, \dots, j_M} = \sum_{i_n}^{I_n} x_{i_1, \dots, i_{n-1}, i_n, i_{n+1}, \dots, i_N} y_{j_1, \dots, j_{m-1}, i_n, j_{m+1}, \dots, j_M} \quad (2.24)$$

In fact, mode- n product is a particular case of contraction product:

$$\underline{\mathbf{X}} \times_n \mathbf{A}^{(n)} = \underline{\mathbf{X}} \times_n^1 \mathbf{A}^{(n)} \quad (2.25)$$

which directly leads to the following equality:

$$\underline{\mathbf{X}} \times_n^1 \mathbf{A}^{(n)} = \underline{\mathbf{X}} \times_n^2 \mathbf{A}^{(n)^T} \quad (2.26)$$

It is worth mentioning that for a sequence of contraction products, there are cases when the order in which contraction is performed matters. More specifically:

$$\begin{aligned} \underline{\mathbf{X}} \times_a^b \underline{\mathbf{Y}} \times_c^d \underline{\mathbf{Z}} &= \underline{\mathbf{Y}} \times_c^d \underline{\mathbf{Z}} \times_a^b \underline{\mathbf{X}}, & \text{for } b \geq c \\ \underline{\mathbf{X}} \times_a^b \underline{\mathbf{Y}} \times_c^d \underline{\mathbf{Z}} &= \underline{\mathbf{X}} \times_a^b \left(\underline{\mathbf{Y}} \times_c^d \underline{\mathbf{Z}} \right), & \text{for } b < c \end{aligned} \quad (2.27)$$

Furthermore, the contraction of two tensors can be performed across several or even all modes at the same time. For simplicity we will consider tensors, $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ of fixed order, but multimode contraction product can be generalised to any arbitrary high order as expressed below:

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_{4,3}^{1,2} \underline{\mathbf{Y}}$$

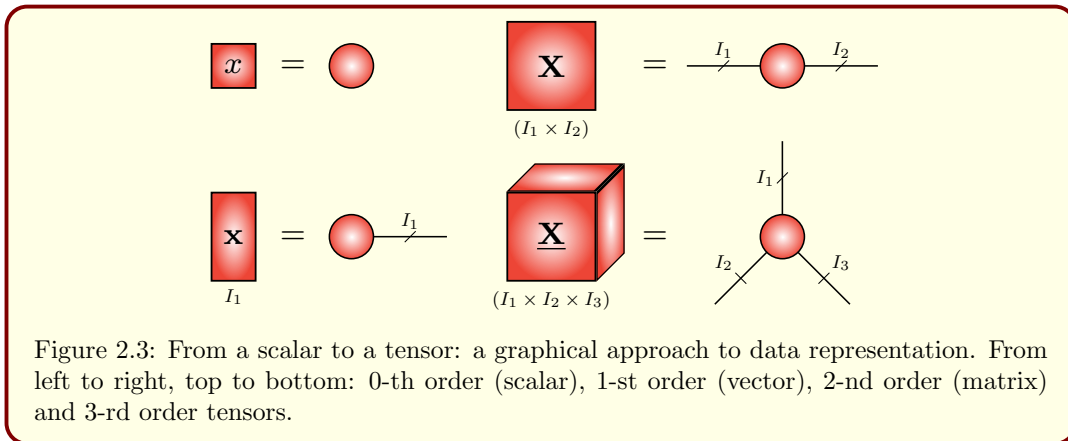
$$z_{i_1, i_2, j_3, j_4} = \sum_{i_3=1}^{I_3} \sum_{i_4=1}^{I_4} x_{i_1, i_2, i_3, i_4} y_{i_4, i_3, j_3, j_4} \quad (2.28)$$

where $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$, $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3 \times J_4}$, $\underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times I_2 \times J_3 \times J_4}$, $J_1 = I_4$, and $J_2 = I_3$.

Finally, a special case of contraction product between two tensors, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, across all modes is commonly referred to as the *inner product*:

$$\underline{\mathbf{X}} \times_{1, \dots, N}^{1, \dots, N} \underline{\mathbf{Y}} = \sum_{i_1=1}^{I_1} \dots \sum_{i_N=1}^{I_N} x_{i_1, \dots, i_N} y_{i_1, \dots, i_N} \quad (2.29)$$

2.1.3 Graphical representation of the main components



2. Prerequisites: Multilinear Algebra

Dealing with mathematical expressions that describe relations between multi-dimensional arrays of generic, arbitrary high order quickly becomes cumbersome, prone to errors and lacks intuitive explanations. Consequently, this contributes to a high entry barrier for new researchers who want to dive deep into the uncharted territories of tensor analysis. To address this issue, researchers in the signal processing field have drawn inspiration from quantum physics, chemometrics and biomedical engineering wherein mathematical operations are represented diagrammatically [1]. Such abstract nevertheless declarative approach to tensors and multi-linear operations is conventionally referred to as Tensor Networks (TNs) [9, 47, 48]. In the essence of *nodes and edges* lays two fundamental principals:

- Every data structure, i.e. tensor of any order, becomes illustrated as a node with leafs steaming out it that denote particular dimensions, as shown on Fig. 2.3;
- All mathematical operations are displayed in the form of simple connections between nodes, in conjunction with special signs for the Kronecker product or unfolding procedures where suitable. Some of the common multi-linear operations are depicted on Fig. 2.4.

Yielded structures closely resemble mathematical bidirectional graphs [1, 49] from which one can draw parallels on topology properties and definitions. This methodology has already been shown to open new horizons towards understanding advanced tensor interconnections and operations that often time involve several tensors at once, thus, lower entry barrier for new researchers in a field. Moreover, it offers avenues for insights even in the most simple cases of tensor representations and decompositions allowing familiar researchers to attain a deeper understanding of this captivating field.

Remark 3. *The graphical way to represent tensors adds the concept of “topology” to tensors and TNs. It allows for tensors to be represented as graphs, and makes it possible to grasp a visualization of their contractions. The association of each mode with their corresponding tensor is now clear, and working with graphical representations gives the possibility of “reshaping” problems in more convenient forms, facilitating a deeper understanding of the underlying mathematical operations.*

2.2 Tensor Generation and Data Rearrangement

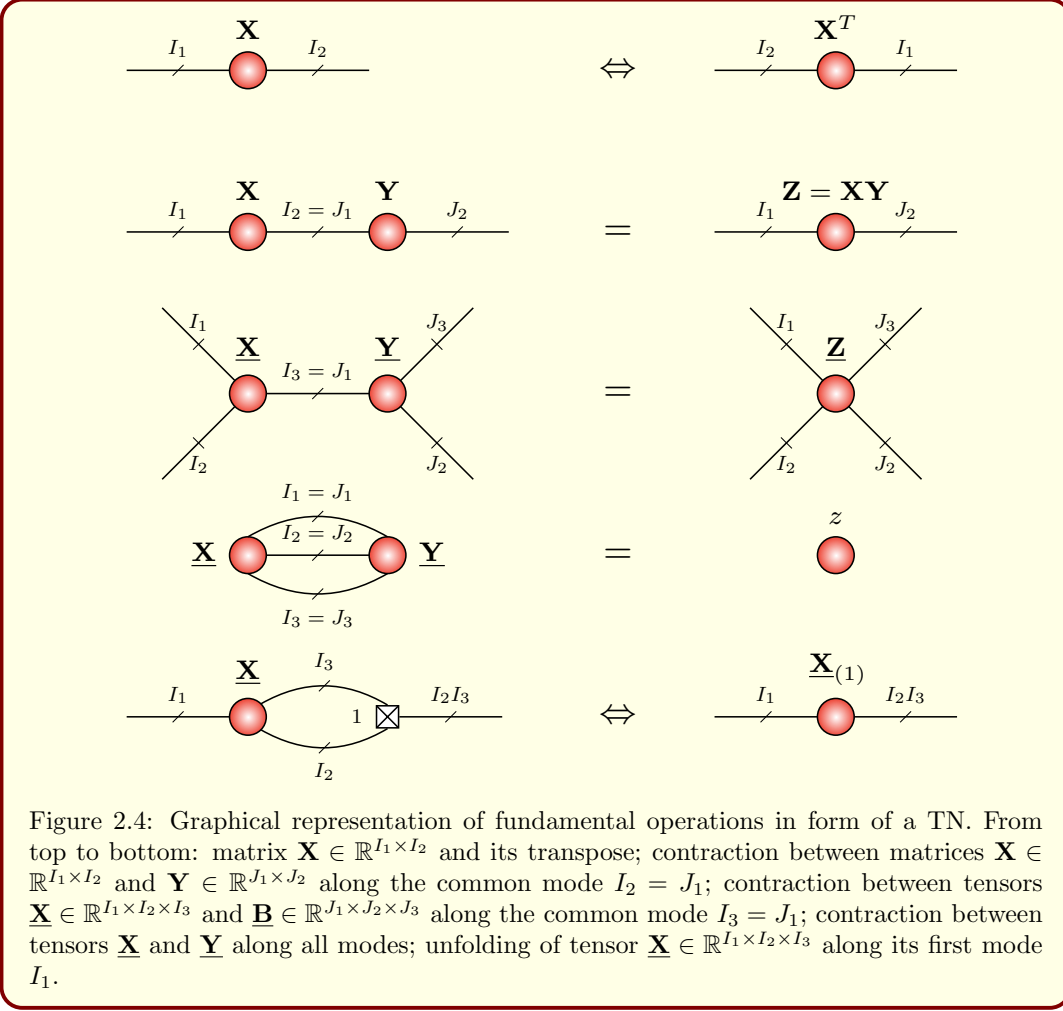
Tensor techniques require the availability of tensor data. Most commonly, information is captured as a of sequence or collection of sequences of discrete data points and then stored and processed in the form of a vector or a matrix respectively. However, various types of data can be represented in the form of a tensor. In such cases, it is essential to make a shift from a “flat-view” matrix way to the N-dimensional tensor way of representation for performing effective analysis. This procedure is known as *tensorisation* which implies generation of higher-order structured tensors from the lower-order data formats [9]. The most common tensorisation categories are:

- Rearrangement of lower-dimensional data structures;
- Mathematical construction;
- Experiment design.

2.2.1 Rearrangement of lower-dimensional data

Large scale matrices or vectors can be readily tensorised, i.e. reshaped into a tensor format, so that they can be compressed through tensor decompositions [50, 51, 52].

2. Prerequisites: Multilinear Algebra



Generic folding

One of the simplest ways to perform tensorisation is by reshaping or *generic folding* the original data [53]. This method can also be referred to as *segmentation*. Note, that it preserves the initial number of elements and some of the sequential ordering between them without introducing any new samples or explicit relations.

Definition 2.13. Any given vector $\mathbf{x} \in \mathbb{R}^{I_1 I_2 \dots I_N}$ can be **folded** into an N -th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, where relation between their entries is defined by:

$$\underline{\mathbf{X}}[i_1, i_2, \dots, i_N] = x_i \quad (2.30)$$

for all $1 \leq i_n \leq I_n$, where $i = 1 + \sum_{n=1}^N (i_n - 1) \prod_{k=1}^{n-1} I_k$.

For instance, tensorisation of a vector $\mathbf{x} \in \mathbb{R}^L$ (tensor of order 1) into a matrix $\mathbf{X} \in \mathbb{R}^{L \times L/I}$ (tensor of order 2) takes form:

$$\mathbf{X} = \begin{bmatrix} x(1) & x(I+1) & \cdots & x(L-I+1) \\ x(2) & x(I+2) & \cdots & x(L-1+2) \\ \vdots & \vdots & \ddots & \vdots \\ x(I) & x(2I) & \cdots & x(L) \end{bmatrix}. \quad (2.31)$$

2. Prerequisites: Multilinear Algebra

Extraction of small segments and stacking them sequentially can be beneficial for signals of exponential nature, e.g. $\mathbf{x} = [1 \quad z \quad z^2 \quad z^3 \quad z^4 \quad z^5]$:

$$\mathbf{X} = \begin{bmatrix} 1 & z & z^2 \\ z^3 & z^4 & z^5 \end{bmatrix} = \begin{bmatrix} 1 \\ z^3 \\ \vdots \end{bmatrix} [1 \quad z \quad z^2 \quad \cdots] \quad (2.32)$$

Hankel folding

The Hankel matrix is a structured matrix whereby each ascending skew diagonal from left to right is constant, and has found applications in control theory [8]. In particular, a *Hankel matrix* $\mathbf{X} \in \mathbb{R}^{I \times J}$ is defined as

$$\mathbf{X} = \begin{bmatrix} x(1) & x(2) & \cdots & x(J) \\ x(2) & x(3) & \cdots & x(J+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(I) & x(I+1) & \cdots & x(K) \end{bmatrix} \quad (2.33)$$

where $K = I + J - 1$.

Definition 2.14. An N -th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be **Hankel folding** of a vector $\mathbf{x} \in \mathbb{R}^K$ if the relation between their elements follows the rule:

$$\underline{\mathbf{X}}[i_1, i_2, \dots, i_N] = x_i \quad (2.34)$$

for all $1 \leq i_n \leq I_n$ where $i = \sum_{n=1}^N I_n - N + 1$.

It may be convenient to tensorise a vector or a matrix via a process that injects such ‘‘Hankel folding’’ structure [9]. For example, consider an exponential signal $\mathbf{x}$ with $x_i = az^i$ then its rearrangement into:

$$\mathbf{X} = a \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \end{bmatrix} [1 \quad z \quad z^2 \quad \cdots]. \quad (2.35)$$

Observe, that the obtained Hankel matrix has rank 1, thus, exhibiting the most simplistic structure which has a broad relevance in harmonic retrieval, for example.

Toeplitz folding

The Toeplitz matrix is a structured matrix with constant entries in each diagonal [9]. From an original vector, $\mathbf{x} \in \mathbb{R}^K$, it is constructed as:

$$\mathbf{X} = \begin{bmatrix} x(I) & x(I+1) & x(I+2) & \cdots & x(J) \\ x(I-1) & x(I) & x(I+1) & \cdots & x(J-1) \\ x(I-2) & x(I-1) & x(I) & \cdots & x(J-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(1) & x(2) & x(3) & \cdots & x(J-I+1) \end{bmatrix} \quad (2.36)$$

where $K = (I + J - 1)$

Definition 2.15. An N -th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be **Toeplitz folding** of a vector $\mathbf{x} \in \mathbb{R}^K$ if relation between their elements adheres:

$$\underline{\mathbf{X}}[i_1, i_2, \dots, i_N] = \mathbf{x}[\bar{i}_1 + \cdots + \bar{i}_{N-1} + i_N] \quad (2.37)$$

for all $1 \leq i_n \leq I_n$ where $\bar{i}_n = I_n - i_n$.

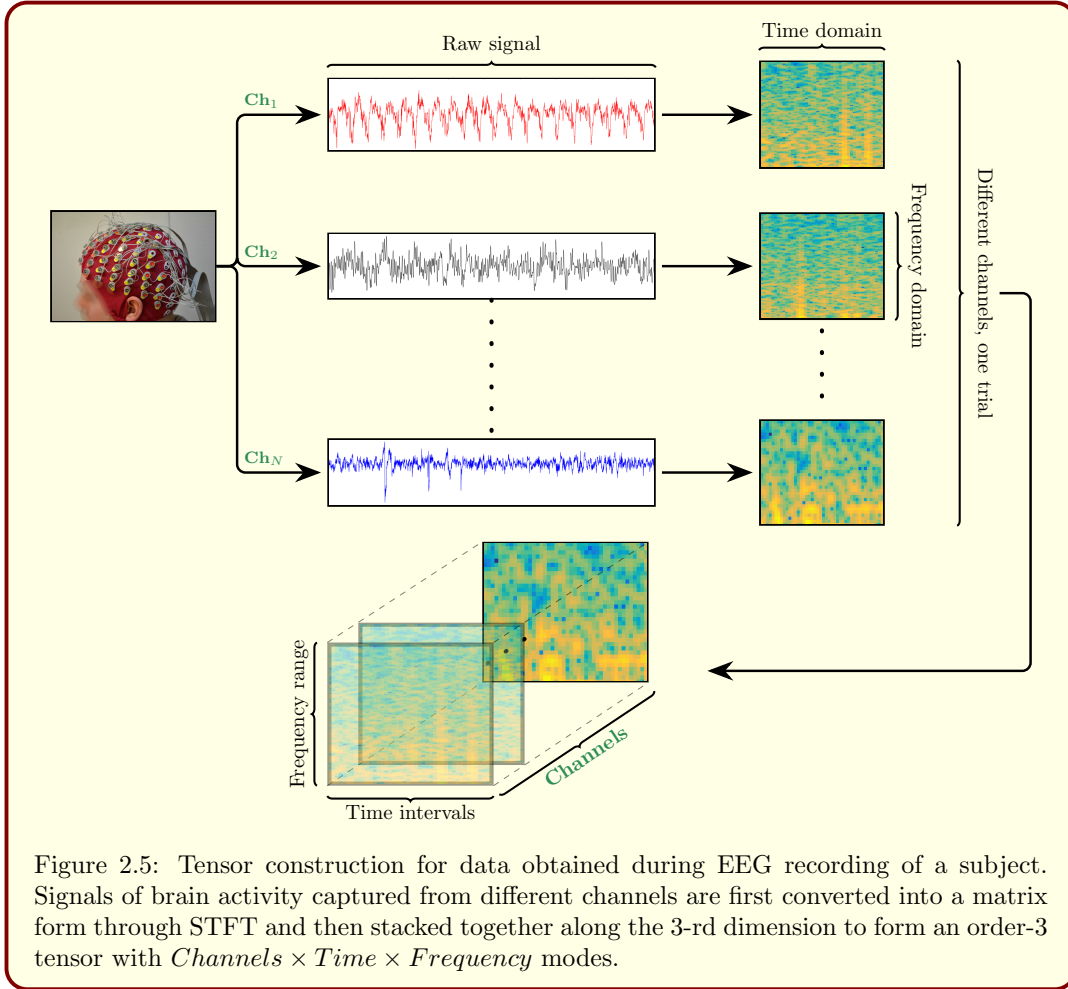


Figure 2.5: Tensor construction for data obtained during EEG recording of a subject. Signals of brain activity captured from different channels are first converted into a matrix form through STFT and then stacked together along the 3-rd dimension to form an order-3 tensor with $Channels \times Time \times Frequency$ modes.

Toeplitz structure of data have been adopted and had a great success in many signal processing applications through covariance matrices, such as prediction, estimation and harmonic analysis to name a few [54, 55].

Remark 4. *The examples above demonstrate tensorisation from a vector to matrix. However, the underlying concepts also apply for data rearrangement to a tensor of any arbitrary high order, and the generalisation of these relations is straightforward.*

Remark 5. *Any folding procedure should be based on a hypothesis that the outcome of the corresponding linear transformations (mappings) should exhibit low rank.*

2.2.2 Mathematical construction and experiment design

Other widely adopted tensorisation techniques are *experiment design* and *mathematical construction* that are tightly coupled to each other. The former naturally occurs while capturing multi-variate data, for example:

- Information collected for network analysis and security contains footprints of both sender and receiver. Hence, such data can be reasonably *stacked* in a form of a 5-th order tensor with dimensions $Source\ IP \times Target\ IP \times Port \times Resource \times Time$ [56];

2. Prerequisites: Multilinear Algebra

- Biomedical data, such as recordings of brain activity, often collected for multiple patients that participated in multiple activities. This allows to organise data into a 4-th order tensor with *Time* $\times$ *Channel* $\times$ *Patient* $\times$ *Trial* modes [12].

Definition 2.16. Consider grouping J order- N tensor samples, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, so as to form the order- $(N+1)$ data tensor, $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J}$. This is the **stacking** operation which is denoted by

$$\underline{\mathbf{Y}} = \text{stack}_{N+1}(\underline{\mathbf{X}}[1], \dots, \underline{\mathbf{X}}[J]) \quad (2.38)$$

In other words, the so combined tensor samples introduce another dimension, the $(N+1)$ th mode of $\underline{\mathbf{Y}}$, such that the mode- $(N+1)$ unfolding of $\underline{\mathbf{Y}}$ becomes

$$\underline{\mathbf{Y}}_{(N+1)} = [\text{vec}(\underline{\mathbf{X}}[1]), \dots, \text{vec}(\underline{\mathbf{X}}[J])] \quad (2.39)$$

The procedure of mathematical construction implies that slices of a tensor are generated through transform-domain methods which then are stacked together across additional dimension. For instance, this can be achieved by a time-frequency representation of a given signal via Short Time Fourier Transform (STFT), Gabor countourlet or wavelet transform. Thus, if data are stored as *Channel* $\times$ *Time* matrix then it can be converted to and analysed as a 3-rd order tensor with dimensions *Channel* $\times$ *Time* $\times$ *Frequency* [10, 11]. A self-explanatory example of tensorisation via mathematical construction is illustrated on Fig. 2.5.

2.3 Tensor Decompositions

2.3.1 Canonical polyadic decomposition and Kruskal representation

The CPD decomposes a tensor into a sum of rank-1 tensors. For an order-3 tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, the CPD is expressed as:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \quad (2.40)$$

where $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$, $\mathbf{c} \in \mathbb{R}^K$. If the approximation in Eq. (2.40) is exact, then R is said to be the rank of the tensor. Likewise, the CPD representation in element-wise form is:

$$\underline{\mathbf{X}}[i, j, k] = \sum_{r=1}^R \lambda_r a_{i_r} b_{j_r} c_{k_r} \quad (2.41)$$

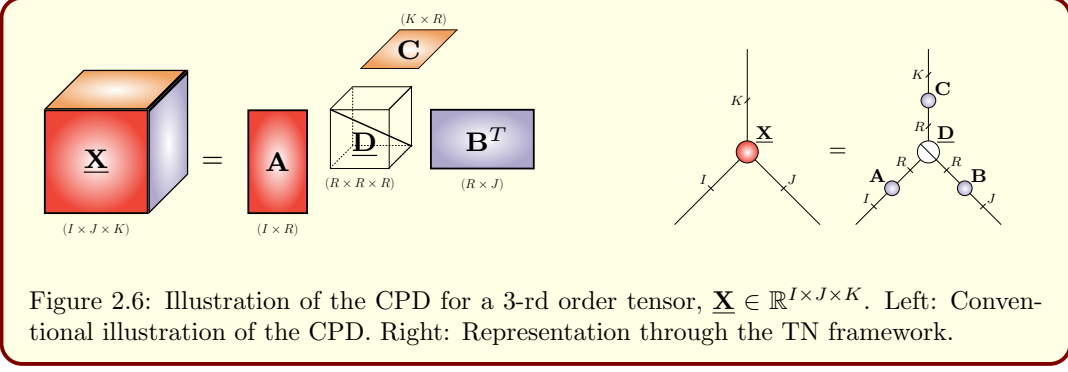
Remark 6. The CPD almost never yields an exact representation, $\hat{\underline{\mathbf{X}}}$, for a real world data tensor, $\underline{\mathbf{X}}$. However, if the approximation error $\|\underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}}\|_F$ is below a predefined threshold, then R is still considered to be the rank of the tensor $\underline{\mathbf{X}}$.

If the rank-1 component vectors that correspond to the same mode of $\underline{\mathbf{X}}$, e.g. $\mathbf{a}_r \ \forall r \in R$, are stacked together into matrices, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_R]$, and scale factors λ_r are placed on the super-diagonal of some core tensor, $\underline{\mathbf{D}}$, then the CPD can be expressed using tensor factorisation notation as follows:

$$\begin{aligned} \underline{\mathbf{X}} &= \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= \llbracket \underline{\mathbf{D}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket \end{aligned} \quad (2.42)$$

where the matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$ are referred to as *factor matrices* and $\underline{\mathbf{D}} \in \mathbb{R}^{R \times R \times R}$ as a *core tensor* with non-zero elements λ_r located on its super-diagonal, that is:

$$\underline{\mathbf{D}}[r_1, r_2, r_3] = \begin{cases} \lambda_r, & \text{if } r_1 = r_2 = r_3 = r \\ 0, & \text{otherwise} \end{cases} \quad (2.43)$$



Definition 2.17. The **Kruskal representation** of a given order- N tensor is a collection of N so called factor matrices and some core tensor that ensures a one-to-one relation between columns of associated factor matrices. Graphical representation of the CPD for the 3-rd order case is depicted on Fig. 2.6.

Since most of the computational implementations operate in the realm of two-dimensional data, we need to be able express the CPD in a matricised form, where the original tensors have to be replaced with their unfolded versions and the relation between factor matricised should be adjusted accordingly. For the 3-rd order case, Eq. (2.42) becomes

$$\begin{aligned}\underline{\mathbf{X}}_{(1)} &= \mathbf{A} \left(\mathbf{C} \odot \mathbf{B} \right)^T \\ \underline{\mathbf{X}}_{(2)} &= \mathbf{B} \left(\mathbf{C} \odot \mathbf{A} \right)^T \\ \underline{\mathbf{X}}_{(3)} &= \mathbf{C} \left(\mathbf{B} \odot \mathbf{A} \right)^T\end{aligned}\tag{2.44}$$

Generalisation of Eq. (2.42) and Eq. (2.44) to an order- N tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is straightforward:

$$\begin{aligned}\underline{\mathbf{X}} &= \sum_{r=1}^R \lambda_r \left(\underset{n=1}{\overset{N}{\circ}} \mathbf{a}_r^{(n)} \right) \\ &= \underline{\mathbf{D}} \underset{n=1}{\overset{N}{\times}} \mathbf{A}^{(n)} \\ &= \llbracket \underline{\mathbf{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)} \rrbracket\end{aligned}\tag{2.45}$$

According to Eq. (2.22), this can be expressed using mode- n unfolding of $\underline{\mathbf{X}}$ as:

$$\underline{\mathbf{X}}_{(n)} = \mathbf{A}^{(n)} \underline{\mathbf{D}}_{(n)} \left(\underset{i=n}{\overset{1}{\otimes}} \mathbf{A}^{(i)} \right)^T.\tag{2.46}$$

Notice, that the $\underline{\mathbf{D}}_{(n)}$ has a special structure due to the fact that $\underline{\mathbf{D}} = \text{diag}(\lambda_r)$, which allows us to re-write generalised form of the CPD from Eq. (2.46) in a more computationally efficient form by employing the Khatri-Rao product instead of the Kronecker product:

$$\underline{\mathbf{X}}_{(n)} = \mathbf{A}^{(n)} \mathbf{\Lambda} \left(\underset{i=n}{\overset{1}{\odot}} \mathbf{A}^{(i)} \right)^T.\tag{2.47}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_r) \in \mathbb{R}^{R \times R}$.

2. Prerequisites: Multilinear Algebra

Uniqueness

The *Kruskal representation* of any tensor of an arbitrary high order is often unique for a given R . In fact, it has been shown that the sufficient condition that guarantees the uniqueness of the CPD of a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, takes form:

$$\sum_{n=1}^N k_{\mathbf{A}^{(n)}} \geq 2R + (N - 1) \quad (2.48)$$

where R is the rank of a tensor $\underline{\mathbf{X}}$, while $k_{\mathbf{A}^{(n)}}$ denotes the rank of factor matrices $\mathbf{A}^{(n)}$ [57, 58].

Definition 2.18. The **Kruskal rank**, denoted by $k_{\mathbf{A}^{(n)}}$, is the maximum possible k such that any k columns of a factor matrix $\mathbf{A}^{(n)}$ from the Kruskal representation of the tensor at hand are linearly independent.

However, it is worth mentioning that the CPD is prone to scaling and permutation ambiguity due to its inherent multi-linear nature [4]. Consider a Kruskal representation of an order-3 tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$:

$$\begin{aligned} \underline{\mathbf{X}} &= \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \\ &= \llbracket \underline{\mathbf{D}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket \end{aligned} \quad (2.49)$$

Then, the permutation uncertainty implies that the columns of factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ can be reordered arbitrarily:

$$\llbracket \underline{\mathbf{D}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket = \llbracket \underline{\mathbf{D}}; \mathbf{A}\mathbf{P}, \mathbf{B}\mathbf{P}, \mathbf{C}\mathbf{P} \rrbracket \quad (2.50)$$

where $\mathbf{P} \in \mathbb{R}^{R \times R}$ is some permutation matrix. At the same time, the scalability indeterminacy refers to the fact that we can scale the individual column vectors:

$$\underline{\mathbf{X}} = \sum_{r=1}^R \lambda_r (\alpha_r \mathbf{a}_r) \circ (\beta_r \mathbf{b}_r) \circ (\gamma_r \mathbf{c}_r) \quad (2.51)$$

as long as $\alpha_r \beta_r \gamma_r = 1$ for all $r = 1, \dots, R$.

Computation

For a 3-rd order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, its CP representation is obtained through a solution to the following optimization problem:

$$\begin{aligned} &\arg \min_{\hat{\underline{\mathbf{X}}}} \left\| \underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}} \right\|_F \\ \text{s.t. } &\hat{\underline{\mathbf{X}}} = \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = \llbracket \underline{\mathbf{D}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket \end{aligned} \quad (2.52)$$

Since the CPD describes inherently multi-linear relations with a data tensor, the most common approach to solve the above is through an Alternating Least Squares (ALS) procedure. This implies that only one of the matrices $\mathbf{A}$, $\mathbf{B}$, or $\mathbf{C}$, is estimated at a time while others are kept fixed. Thus, based on the unfolded version of the CPD from Eq. (2.47), $\mathbf{A}$ is estimated via:

$$\hat{\mathbf{A}} = \mathbf{X}_{(1)} \left(\mathbf{C} \odot \mathbf{B} \right) \left(\mathbf{C}^T \mathbf{C} \otimes \mathbf{B}^T \mathbf{B} \right)^\dagger \quad (2.53)$$

where $\hat{\mathbf{A}} = \mathbf{A}\mathbf{D}$, $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_R)$, and $\mathbf{B}$, $\mathbf{C}$ are kept fixed. These steps are then repeated in an analogous manner for $\mathbf{B}$ and $\mathbf{C}$ until the model fits the underlying tensor

2. Prerequisites: Multilinear Algebra

Algorithm 1: ALS for the CP decomposition of an N -th order tensor

Input : Data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and rank R
Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$, for $n = 1, \dots, N$ and vector λ

```

1 Initialize  $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ 
2 while not converged or iteration limit is not reached do
3   for  $n = 1, \dots, N$  do
4      $\mathbf{U} \leftarrow \bigotimes_{\substack{i=1 \\ i \neq n}}^N \left( \mathbf{A}^{(i)T} \mathbf{A}^{(i)} \right)$ 
5      $\mathbf{A}^{(n)} \leftarrow \bigodot_{\substack{i=1 \\ i \neq n}}^N \left( \mathbf{A}^{(i)} \right) \mathbf{U}^\dagger$ 
6   end for
7   Normalize columns of  $\mathbf{A}^{(n)}$  and store values in vector  $\lambda$ 
8 end while
9 return  $\mathbf{A}^{(n)}$ , for  $n = 1, \dots, N$ , and  $\lambda$ 
```

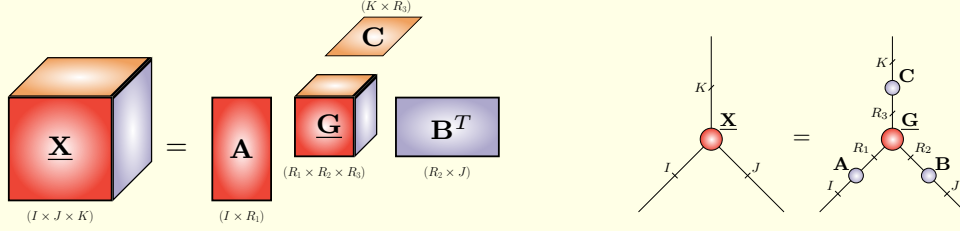


Figure 2.7: Illustration of the TKD for a 3-rd order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$. Left: Conventional illustration of the TKD. Right: Representation through TNs framework.

arbitrarily well or other stopping criteria are met [4]. Note, the order in which these factor matrices are obtained is irrelevant. The computation of the CPD for a generic N -th order tensor is summarised in Algorithm 1.

Remark 7. The Kruskal representation might change significantly for different values of tensor rank, R . This is a major downside of the CPD as it requires us to “randomly” choose R since it is rarely known beforehand in practice, and there is no algorithm for rank estimation [4, 59].

2.3.2 Tucker decomposition and representation

The Tucker decomposition (TKD) is a form of higher-order Principal Component Analysis (PCA). It decomposes a tensor into a *core tensor* which is transformed (multiplied) by a *factor matrix* along each of its modes [4]. Elementwise, for a 3-rd order tensor, TKD takes the form:

$$\underline{\mathbf{X}}[i_1, i_2, i_3] = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} a_{i_1 r_1} b_{i_2 r_2} c_{i_3 r_3}. \quad (2.54)$$

Here, the values R_1, R_2 and R_3 are most often smaller than the respective individual dimensions I, J and K . Additionally, a tuple (R_1, R_2, R_3) is conventionally referred to as the *multi-linear rank*, although it does not strictly follow its definition.

2. Prerequisites: Multilinear Algebra

In analogy with the Kruskal representation of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, the Tucker form can be expressed as:

$$\begin{aligned}\underline{\mathbf{X}} &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \mathbf{a}_{r_1} \circ \mathbf{b}_{r_2} \circ \mathbf{c}_{r_3} \\ &= \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \\ &= \llbracket \underline{\mathbf{G}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket\end{aligned}\tag{2.55}$$

where $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$, $\mathbf{C} \in \mathbb{R}^{K \times R_3}$ are called the factor matrices, and can be thought of as the principal components in each mode, whereas the tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is referred to as the core tensor.

Definition 2.19. The **Tucker representation** of a given order- N tensor is a collection of N so called factor matrices and some core tensor that determines the degree of interactions between all components of factor matrices without necessarily imposing a one-to-one relation. An illustration of the Tucker form for the 3-rd order case is depicted on Fig. 2.7.

The Tucker form of an original tensor $\underline{\mathbf{X}}$ can also be expressed using only linear algebra terms and operations as:

$$\begin{aligned}\underline{\mathbf{X}}_{(1)} &= \mathbf{A} \underline{\mathbf{G}}_{(1)} \left(\mathbf{C} \otimes \mathbf{B} \right)^T \\ \underline{\mathbf{X}}_{(2)} &= \mathbf{B} \underline{\mathbf{G}}_{(2)} \left(\mathbf{C} \otimes \mathbf{A} \right)^T \\ \underline{\mathbf{X}}_{(3)} &= \mathbf{C} \underline{\mathbf{G}}_{(3)} \left(\mathbf{B} \otimes \mathbf{A} \right)^T\end{aligned}\tag{2.56}$$

Observe that $\underline{\mathbf{G}}_{(n)}$ is not guaranteed to exhibit any special structure in TKD, therefore unlike in CPD it cannot be re-written in a more efficient way. Thus, for a general N -th order tensor, the Tucker representation takes the form:

$$\begin{aligned}\underline{\mathbf{X}} &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1, r_2, \dots, r_N} \left(\bigotimes_{i=1}^N \mathbf{a}_{r_i}^{(i)} \right) \\ &= \underline{\mathbf{G}} \times_{n=1}^N \mathbf{A}^{(n)} \\ &= \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)} \rrbracket\end{aligned}\tag{2.57}$$

whereby $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times \dots \times R_N}$ and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, with its mode- n unfolding expressed as:

$$\underline{\mathbf{X}}_{(n)} = \mathbf{A}^{(n)} \underline{\mathbf{G}}_{(n)} \left(\bigotimes_{\substack{i=1 \\ i \neq n}}^N \mathbf{A}^{(i)} \right)^T.\tag{2.58}$$

Remark 8. In general, the Tucker representation is not unique, however, the uniqueness can be achieved by imposing constraints on the factor matrices during computation.

Computation

Despite the obvious resemblance of notation, the CPD and TKD serve different purposes. The former is typically employed in problems that seek interpretability of extracted components, whereas the latter is utilised for efficient data representation which is satisfied by virtue of the multi-linear rank that provides great flexibility to approximation of the original data.

In practice, there exist several algorithms to represent a given tensor in the Tucker format. The most simplistic one is the Higher Order Singular Value Decomposition (HOSVD), introduced

2. Prerequisites: Multilinear Algebra

Algorithm 2: HOSVD for an N -th order tensor

Input : Data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \cdots \times I_N}$ and multi-linear rank $(R_1, \dots, R_N)$
Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for $n = 1, \dots, N$ and core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

```

1 for  $n = 1, \dots, N$  do
2   |  $\mathbf{A}^{(n)} \leftarrow R_n$  principal singular vectors of  $\underline{\mathbf{X}}_{(n)}$ 
3 end for
4  $\underline{\mathbf{G}} \leftarrow \underline{\mathbf{X}} \times_{n=1}^N \mathbf{A}^{(n)T}$ 
5 return  $\underline{\mathbf{G}}$  and  $\mathbf{A}^{(n)}$ , for  $n = 1, \dots, N$ 

```

Algorithm 3: HOOI for an N -th order tensor

Input : Data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \cdots \times I_N}$ and multi-linear rank $(R_1, \dots, R_N)$
Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for $n = 1, \dots, N$ and core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

```

1 Initialize  $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$  for  $n = 1, \dots, N$ , using HOSVD from Algorithm 2
2 while not converged or iteration limit is not reached do
3   for  $n = 1, \dots, N$  do
4     |  $\underline{\mathbf{Y}} \leftarrow \underline{\mathbf{X}} \times_{\substack{i=1 \\ i \neq n}}^N \mathbf{A}^{(i)T}$ 
5     |  $\mathbf{A}^{(n)} \leftarrow R_n$  principal singular vectors of  $\underline{\mathbf{Y}}_{(n)}$ 
6   end for
7 end while
8  $\underline{\mathbf{G}} \leftarrow \underline{\mathbf{X}} \times_{n=1}^N \mathbf{A}^{(n)T}$ 
9 return  $\underline{\mathbf{G}}$  and  $\mathbf{A}^{(n)}$ , for  $n = 1, \dots, N$ 

```

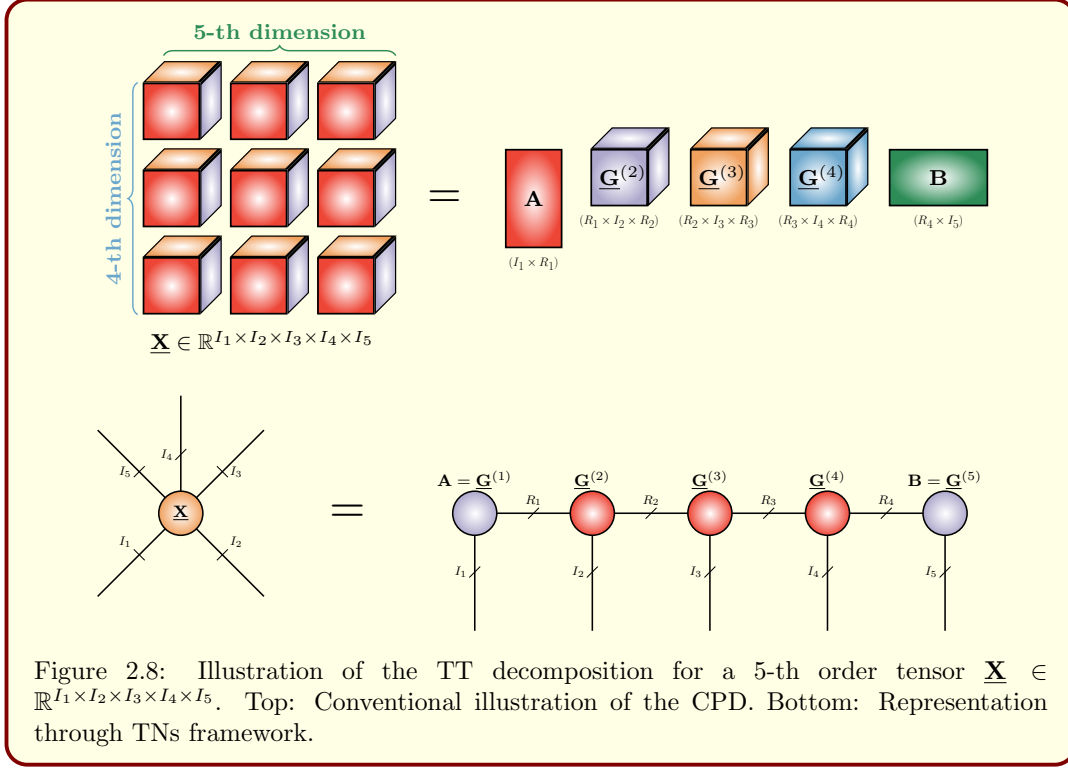
in [22] and further popularised in [23, 24], The HOSVD algorithm can be seen as a natural extension of Singular Value Decomposition (SVD) as it also constrains the factor matrices to be columnwise orthonormal which is achieved by computing a truncated SVD of the unfolded versions of an initial tensor. For instance, consider a tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, then the procedure to obtain its Tucker representation $\llbracket \underline{\mathbf{G}}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$ with multilinear rank (R_1, R_2, R_3) can be expressed as a sequence of two steps. First, we obtain the factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ through:

$$\begin{aligned}
\underline{\mathbf{X}}_{(1)} &= \mathbf{U}^{(1)} \boldsymbol{\Sigma}^{(1)} \mathbf{V}^{(1)T} & \rightarrow & \mathbf{A} = \mathbf{U}^{(1)}[1 : R_1] \\
\underline{\mathbf{X}}_{(2)} &= \mathbf{U}^{(2)} \boldsymbol{\Sigma}^{(2)} \mathbf{V}^{(2)T} & \rightarrow & \mathbf{B} = \mathbf{U}^{(2)}[1 : R_2] \\
\underline{\mathbf{X}}_{(3)} &= \mathbf{U}^{(3)} \boldsymbol{\Sigma}^{(3)} \mathbf{V}^{(3)T} & \rightarrow & \mathbf{C} = \mathbf{U}^{(3)}[1 : R_3].
\end{aligned} \tag{2.59}$$

Observe that it is not necessary to compute the right singular matrices, as one can take advantage of the common situation when, for instance, $I \ll JK$, and save a large number of operations. Then, the core tensor, $\underline{\mathbf{G}}$, is computed as a projection of the original tensor $\underline{\mathbf{X}}$ onto the subspaces spanned by the $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, that is

$$\underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T \tag{2.60}$$

Algorithm 2 summarises the HOSVD for a generic N -th order tensor. An important observation from truncated HOSVD is that it does not provide an optimal approximation of the raw multi-



dimensional array. This issue can be addressed by employing an iterative procedure for updating the factor matrices through the ALS-based algorithm, conventionally referred to as the Higher Order Orthogonal Iteration (HOOI). Assuming a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, the HOOI solves the following optimisation problem:

$$\begin{aligned} \arg \min_{\underline{\mathbf{G}}, \mathbf{A}, \mathbf{B}, \mathbf{C}} & \left\| \underline{\mathbf{X}} - \llbracket \underline{\mathbf{G}}; \mathbf{A} \mathbf{B} \mathbf{C} \rrbracket \right\|_F \\ \text{s.t. } & \mathbf{A} \mathbf{A}^T = \mathbf{I}, \mathbf{B} \mathbf{B}^T = \mathbf{I} \text{ and } \mathbf{C} \mathbf{C}^T = \mathbf{I} \end{aligned} \quad (2.61)$$

where $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R_1}$, $\mathbf{B} \in \mathbb{R}^{J \times R_2}$, $\mathbf{C} \in \mathbb{R}^{K \times R_3}$ are columnwise orthonormal. Similarly to Eq. (2.52), this minimisation problem needs to be addressed for one factor matrix at a time. The procedure computing HOOI is summarised in Algorithm 3.

2.3.3 Tensor train decomposition and representation

Despite of all the advantages in efficient representation and data handling through the Kruskal and Tucker forms, their performance deteriorates and suffers from the *Curse of Dimensionality* with the growth of the order of a multi-dimensional array, thus making them not feasible for real world applications. A numerically reliable way to tackle such cases was originally introduced in the field of quantum information theory through the concept of Tensor Networks (TN). A TN represents a given raw tensor as a set of sparsely interconnected lower-order tensors and factor matrices. This makes processing information in such form more suitable for distributed storage and computation purposes. A particular case of TNs, known as *Tensor Train (TT)*, allows for coupling of only adjacent TT-cores. The TT decomposition was first introduced by Oseledets in [60] and has been attracting more and more attention ever since [1, 61, 62]. Elementwise, it can be expressed as:

2. Prerequisites: Multilinear Algebra

$$\begin{aligned}\underline{\mathbf{X}}[i_1, i_2, \dots, i_N] &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{(N-1)}=1}^{R_{(N-1)}} g_{1i_1 r_1}^{(1)} g_{r_1 i_2 r_2}^{(2)} \cdots g_{r_{(N-1)} i_N 1}^{(N)} \\ &= \underline{\mathbf{G}}_{[:, i_1, :]}^{(1)} \underline{\mathbf{G}}_{[:, i_2, :]}^{(2)} \cdots \underline{\mathbf{G}}_{[:, i_N, :]}^{(N)}\end{aligned}\quad (2.62)$$

where $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $n = 1, \dots, N$ the tuple $(R_1, \dots, R_{N-1})$ is referred to as the *TT-rank*. Observe that the consequent TT-cores, i.e. $\underline{\mathbf{G}}^{(n)}$ and $\underline{\mathbf{G}}^{(n+1)}$, interact with one another only through mode-2 slices, $\underline{\mathbf{G}}_{[:, i_n, :]}^{(n)} \in \mathbb{R}^{R_{n-1} \times R_n}$. There are two special cases amongst TT-cores, specifically, the very first one and the very last one which are of order-2 and therefore represent matrices. Nevertheless, it is straightforward to artificially convert them into order-3 tensors by introducing an “artificial dimension” via a simple data manipulation:

$$\begin{aligned}\underline{\mathbf{G}}^{(1)} &\in \mathbb{R}^{I_1 \times R_1} \rightarrow \underline{\mathbf{G}}^{(1)} \in \mathbb{R}^{1 \times I_1 \times R_1} \\ \underline{\mathbf{G}}^{(N)} &\in \mathbb{R}^{R_{N-1} \times I_N} \rightarrow \underline{\mathbf{G}}^{(N)} \in \mathbb{R}^{R_{N-1} \times I_N \times 1}\end{aligned}\quad (2.63)$$

Remark 9. Because $\underline{\mathbf{G}}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$ and $\underline{\mathbf{G}}^{(N)} \in \mathbb{R}^{R_{N-1} \times I_N}$ are matrices, they are often denoted by $\mathbf{A}$ and $\mathbf{B}$, respectively.

Finally, by employing the notation of contraction product from Eq. (2.24), the TT representation of any tensor takes the form:

$$\begin{aligned}\underline{\mathbf{X}} &= \underline{\mathbf{G}}^{(1)} \times_3^1 \underline{\mathbf{G}}^{(2)} \times_3^1 \underline{\mathbf{G}}^{(3)} \times_3^1 \cdots \times_3^1 \underline{\mathbf{G}}^{(N)} \\ &= \langle \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \dots, \underline{\mathbf{G}}^{(N)} \rangle\end{aligned}\quad (2.64)$$

Definition 2.20. The *Tensor Train representation* of a given order- N tensor is a collection of 3-rd order core tensors sequentially connected through contraction products that prohibit any relations other than between their respective mode-2 slices. A graphical representation of the Tensor Train for the 5-th order case is depicted on Fig. 2.8.

Computation

Algorithm 4: TT-SVD for an N -th order tensor

Input : Data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and desired approximation accuracy ϵ
Output: Core tensors $\underline{\mathbf{G}}^{(1)}, \dots, \underline{\mathbf{G}}^{(N)}$ approximating $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$

- 1 Initialize cores $\underline{\mathbf{G}}^{(1)}, \dots, \underline{\mathbf{G}}^{(N)}$ and $R_0 = 1$
- 2 Compute truncation parameter $\delta = \frac{\epsilon}{\sqrt{N-1}} \|\underline{\mathbf{X}}\|_F$
- 3 $\underline{\mathbf{Z}} \leftarrow \underline{\mathbf{X}}$, and $\underline{\mathbf{Z}} \leftarrow \underline{\mathbf{Z}}_{(1)}$
- 4 **for** $n = 1$ **to** $N - 1$ **do**
- 5 Compute δ -truncated SVD: $\underline{\mathbf{Z}} = \underline{\mathbf{U}} \underline{\mathbf{S}} \underline{\mathbf{V}} + \underline{\mathbf{E}}$, s.t. $\|\underline{\mathbf{E}}\|_F \leq \delta$; $\underline{\mathbf{U}} \in \mathbb{R}^{R_{(n-1)} \times I_n \times R_n}$
- 6 $\underline{\mathbf{G}}^{(n)} \leftarrow \text{Reshape}(\underline{\mathbf{U}}, [R_{(n-1)}, I_n, R_n])$
- 7 $\underline{\mathbf{Z}} \leftarrow \text{Reshape}(\underline{\mathbf{S}} \underline{\mathbf{V}}^T, [R_n I_{(n+1)}, I_{(n+2)} I_{(n+3)} \cdots I_N])$
- 8 **end for**
- 9 $\underline{\mathbf{G}}^{(N)} \leftarrow \underline{\mathbf{Z}}$
- 10 **return** $\underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \dots, \underline{\mathbf{G}}^{(N)}$

The algorithm for a multi-dimensional array of any order, summarised in Algorithm 4, involves an iterative procedure composed of a series of foldings and unfoldings on an original tensor

$\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, in conjunction with employing truncated SVD, hence its name – TT-SVD. At every cycle a core $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ is computed based on the values of the TT-rank $(R_1, R_2, \dots, R_{N-1})$ which are estimated on the fly or have been specified beforehand.

2.4 Chapter Summary

Several important concepts of tensor analysis which are relevant to this thesis have been thoroughly reviewed and summarised in Table 2.1 and Table 2.2. This has allowed for deeper understanding of the implementation of computation algorithms for the CPD-ALS, HOSVD, HOOI and TT-SVD as well as to seamlessly manipulate tensors, with the associated fundamental tensor representations in either Kruskal, Tucker or Tensor Train forms. These are introduced in Section 2.3 and accompanied by intuitive and self explanatory examples. Moreover, we have also covered conditions of the efficient tensor representations that are sufficient to manifest uniqueness of the decomposition at hand. Finally, a notion of the Tensor Network framework has been introduced, before our journey into various tensorisation techniques has started in Section 2.2. This has allowed us to appreciate tensor generation procedures by virtue of a rearrangement of lower-dimensional structures, mathematical construction, or experimental design that are indispensable tools for the analysis and unravelling the mysteries of latent components in multi-dimensional data.

Chapter 3

Feature Extraction and Data Fusion

Some of the best theorising comes after collecting data because then you become aware of another reality.

Robert J. Shiller

Chapter Overview

Feature extraction is a key factor in model reduction, classification and pattern recognition. This is especially important for input data with large dimensions such as multiview images, biomedical data and network traffic, to name a few. Joint analysis of data has the potential to improve our understanding of the underlying structures in complex data sets that exhibit high variety and structural richness, thus, it is of great importance to be equipped with appropriate feature extraction techniques. This chapter begins with exploring a problem of separation of individual from common information intrinsic to the multiblock data. Building upon advances and established methodology within the two dimensional realm, we provide a natural link between the properties of the outer product and the components obtained through tensor decompositions. This additional insight leads to a novel method of identifying common and individual feature subspaces. The non-negativity constraint is further imposed on the so extracted factors, to preserve the physical properties of the original multi-dimensional data samples and relations amongst them. Next, we address the concept of feature fusion in pursuit of exploiting the full potential of the underlying structure and the associated feature localisation. This is achieved by virtue of a rigorous framework for tensor network summation, where we illuminate that interconnections among the cores in a tensor network representation describe the coupling of data structures of the original tensors. The effectiveness of the considered methods for feature extraction and data fusion has been demonstrated through practical applications to multi-class classification problems, whereby the performance has been compared across various supervised learning algorithms.

3.1 Common and Individual Feature Extraction

3.1.1 Introduction

Modern datasets in data science applications have immense volume, veracity, velocity and variety (the four V's of big data) [1, 47], and often exhibit a large degree of structural richness among

3. Feature Extraction and Data Fusion

their entries. This includes various long-term characteristics which makes it prohibitive to the application of classical matrix algebra. Indeed, its “flat-view” way of operation cannot cope with the sheer volume of data and the corresponding imbalanced matrix structures, such as as “tall and narrow” or “short and wide” ones. On the other hand, when arranged into multi-dimensional structures (tensors), the same data often admit much more convenient and mathematically tractable ways of analysis, by virtue of the associated multi-linear algebra. Given that implementation of the standard optimisation and minimisation models does not depend on the type of input data, this makes it easy to extrapolate well defined and extremely efficient algorithms of 2-way component analysis onto the more general case – tensor framework.

The TDs have been established as a promising tool for feature extraction, missing data completion and anomaly detection. However, the generalisation of a matrix to a tensor is intuitive but highly non-trivial, not least due to multi-linear algebra having different properties to linear algebra. In order to obtain meaningful representations, most common factorisation techniques require certain constraints to be imposed in accordance with the properties of the dataset in hand. However, the data organised in a matrix form is bounded by only two degrees of freedom, whereas a tensor can have infinitely many dimensions and consequently infinitely many modes, thus, a variety of different constraints can be imposed at the same time. A “workhorse” algorithm for feature extraction has been presented in [63]. It efficiently identifies statistically dominant characteristics of a given dataset through simultaneous orthogonal factorisations of factor matrices for every dimension of an original tensor. Along these lines, the authors in [10] consider physical meaning of factor matrices obtained through TDs to select the extracted component as a feature of interest. Missing data can also be handled through tensor dictionary learning [64, 65], whereby the tensor structure allows for a simultaneous retrieval of local patterns and establishing the global information. An alternative to supervised learning algorithms for classification has been proposed in [28]. In particular, the HOSVD was employed to reveal a set of basis matrices for describing feature space of each class, hence, a new unknown sample can be assigned label based on the projection and cosine distance. Another approach to classification of multi-dimensional data has been developed around the flexibility of tensor construction aggregated with the fact that TDs do not obscure latent interconnections [66]. By combining slices that represent ground truth and examples to be classified, TDs reveal a degree of correlation among all of them which in its turn could be sufficient to correctly perform class assignment for unlabelled samples.

Over the past decade, the storage capacity has increased dramatically without having a negative effect on affordability of high storage devices. This allows researchers to conduct recordings of the very same phenomenon multiple times but with external conditions changing ever so slightly, such as multiple images of objects recorded under different lighting and angle combination. As a result, this generates an immense amount of multiblock data. Intuitively, the so obtained ensemble of images contains some common and some individual features, and machine learning tasks would benefit from exploiting only either common features (for clustering) or individual features (for identification), both of much lower dimensionality than the original data.

3.1.2 Conventional approach to multiblock data

The analysis of coupled datasets states that multiple measurements of the same event under various conditions are composed of common and individual information and can be subsequently represented as their linear combination. In order to take full advantage of high-dimensional multiblock data and explore as many hidden connections as possible, all entries of a dataset have to be jointly factorised. The main focus of such optimisation problems is to find and extract unique informational subspace that is shared among all members, while at the same time separating it from solely individual characteristics. A number of algorithms has been developed for naturally linked data. However, most of them require us to impose either hard or soft constraints, which sometimes are too restrictive or counter-intuitive. In general, the proposed models for multiset data analysis can be grouped into three categories (illustrated on

3. Feature Extraction and Data Fusion

Fig. 3.1) that are based on:

1. Concatenation of the PCA components (PCAConc);
2. Correlation maximisation (CorrMax);
3. Simulations approximation (SimApprox).

Consider a set of matrices $\mathcal{X}$:

$$\mathcal{X} = \{\mathbf{X}_n \in \mathbb{R}^{I \times J_n} : n \in \mathcal{N}\} \quad (3.1)$$

where the so called block-matrix structure $\mathcal{X}$ could be a representation of medical images, EEG recordings, or financial stock characteristics. All members of such a set of matrices, $\mathcal{X}$, are naturally linked together and it would be beneficial to analyse them simultaneously at the same time, however, the representation in (3.1) yields imbalanced (tall and narrow) structure which is cumbersome for further processing. The main goal of the common and individual feature analysis is, therefore, to make use of the “blessing of dimensionality” associated with tensor structures, in order to find a much lower-dimensional unique subspace, $\bar{\mathbf{A}}$, that is common across all $n \in \mathcal{N}$. In this way, the common subspace, $\bar{\mathbf{A}}$, can be separated from the individual information, $\check{\mathbf{A}}_n$, for every n .

The approaches from the first category above, PCAConc, such as [67], stack all entries $\mathbf{X}_n$ into a tall matrix $\mathbf{X} = [\mathbf{X}_1^T, \mathbf{X}_2^T, \dots, \mathbf{X}_N^T]^T$, and subsequently perform matrix factorisations, such as the PCA, over all data blocks at the same time. Thus, common and individual information can be interpreted as

$$\mathbf{X} = \check{\mathbf{A}}\bar{\mathbf{A}}^T \quad (3.2)$$

Note, PCA like methods do not involve permutation, therefore the relative position of an original block matrix and a corresponding set of individual features is preserved $\check{\mathbf{A}} = [\check{\mathbf{A}}_1^T, \check{\mathbf{A}}_2^T, \dots, \check{\mathbf{A}}_N^T]^T$. In [68], this method was applied to neuroimaging data of patients with Alzheimer’s disease, whereby $\bar{\mathbf{A}}$ is interpreted as well established knowledge about the disease (common components), while $\check{\mathbf{A}}_n$ represents the individual state for a specific patient. However, as with all matrix models, this approach does not generalise well and is only appropriate when all components, $\mathbf{X}_n$, exhibit exactly the same common information.

The most popular techniques from the second category, CorrMax, for the analysis of multi-block data are the Partial Least Squares (PLS) [69] and Canonical Correlation Analysis (CCA) [70]. Despite the similarity of these approaches, the former is most frequently used for regression problems, whereas the latter has become a standard for feature extraction. One of the drawbacks is that they do not provide information which is directly shared between the original blocks. Instead, PLS seeks for a new common subspace that maximises the cross-covariance between dataset members, while CCA – seeks a subspace that is the most correlated with all of them, that is

$$\max_{\mathbf{a}_1, \mathbf{a}_2} \frac{\mathbf{a}_1^T \mathbf{X}_1^T \mathbf{X}_2 \mathbf{a}_2}{\|\mathbf{X}_1 \mathbf{a}_1\|_F \|\mathbf{X}_2 \mathbf{a}_2\|_F} \quad (3.3)$$

Here, the vectors $\mathbf{a}_1, \mathbf{a}_2$ are the basis vectors of the common subspace $\bar{\mathbf{A}}$. Both PLS and CCA have been proven to be very powerful tools for the analysis of data ensembles, as in Eq. (3.1) for $n = 2$. However, they cannot accommodate more than two matrices at the same time and, as a consequence, fail to generalise. As opposed to the methods from the first category, grouping all available information into two tall matrices is not effective, since the mutual subspace will depend on a particular realisation and concatenation sequence.

The last category, SimApprox, considers the problem of separating common and individual features from the optimisation point of view and extends beyond most frequently employed matrix factorisation problems [71]. Namely, for naturally linked datasets, it is reasonable to rewrite a PCA like solution for every entry from Eq. (3.1) into the following form:

3. Feature Extraction and Data Fusion

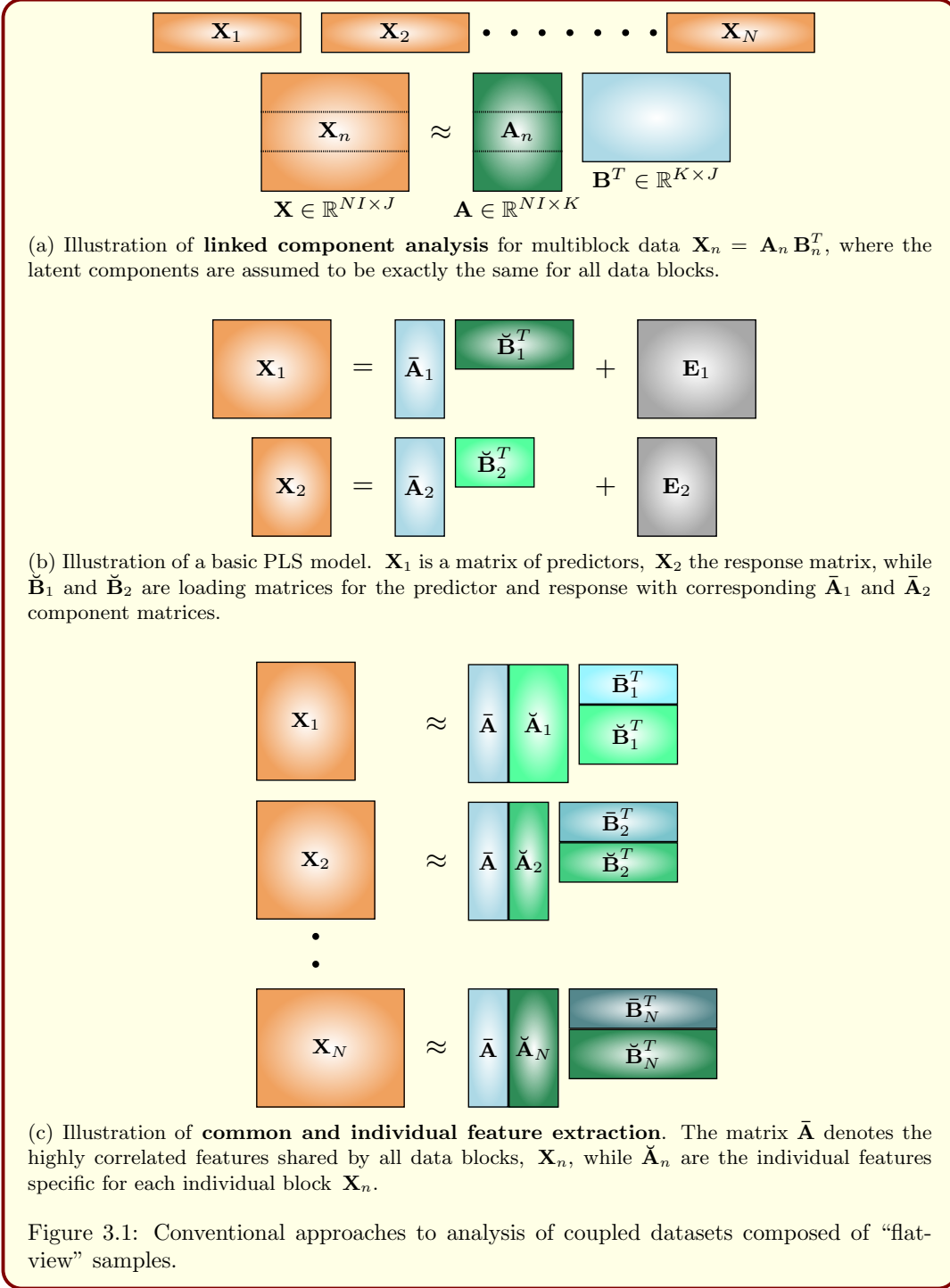


Figure 3.1: Conventional approaches to analysis of coupled datasets composed of “flat-view” samples.

$$\begin{aligned}
 \mathbf{X}_n &= \mathbf{A}_n \mathbf{B}_n^T = [\bar{\mathbf{A}} \quad \check{\mathbf{A}}_n] \begin{bmatrix} \bar{\mathbf{B}}_n^T \\ \check{\mathbf{B}}_n^T \end{bmatrix} \\
 &= \bar{\mathbf{A}} \bar{\mathbf{B}}_n^T + \check{\mathbf{A}}_n \check{\mathbf{B}}_n^T = \bar{\mathbf{X}}_n + \check{\mathbf{X}}_n
 \end{aligned} \tag{3.4}$$

3. Feature Extraction and Data Fusion

where the matrices, $\bar{\mathbf{X}}_n$, represent the common components across the dataset $\mathcal{X}$, while the matrices, $\check{\mathbf{X}}_n$, are the individual components for every $\mathbf{X}_n$ in $\mathcal{X}$. The matrices $\bar{\mathbf{A}}$ and $\check{\mathbf{A}}_n$ are the basis matrices respectively for the matrices $\bar{\mathbf{X}}_n$ and $\check{\mathbf{X}}_n$, while the matrices, $\bar{\mathbf{B}}_n$ and $\check{\mathbf{B}}_n$ represent mixing coefficients so that $\bar{\mathbf{X}}_n = \bar{\mathbf{A}}\bar{\mathbf{B}}_n^T$ and $\check{\mathbf{X}}_n = \check{\mathbf{A}}_n\check{\mathbf{B}}_n^T$.

Remark 10. Due to the linear separability of the matrices $\bar{\mathbf{X}}_n$ and $\check{\mathbf{X}}_n$, it is sufficient to establish the basis of the common information, $\bar{\mathbf{X}}$, as the individual counterpart, $\check{\mathbf{X}}$, can be retrieved by a simple subtraction.

Thus, the main challenge is to estimate basis for common information $\bar{\mathbf{X}}$. To tackle this problem, the algorithm proposed in [72] tries to minimise the cost function formulated as:

$$J(\mathbf{a}_m, \mathbf{z}_{(n,m)}) = \sum_{n=1}^N \|\mathbf{Q}_n \mathbf{z}_{(n,m)} - \mathbf{a}_m\|_F^2 \quad (3.5)$$

where the orthogonal matrix $\mathbf{Q}_n$ is obtained from $\mathbf{X}_n = \mathbf{Q}_n \mathbf{R}_n$ via the QR decomposition, $\mathbf{z}_{(n,m)}$ is the column vector of $\mathbf{Z}_n = \mathbf{R}_n(\mathbf{B}_n^T)^\dagger$ and $\mathbf{a}_m$ is the common component which defines the basis of $\bar{\mathbf{A}}$. Furthermore, for every column vector, $\mathbf{a}_m$, and N respective vectors of mixing coefficients, $\mathbf{z}_{(n,m)}$, the objective function in Eq. (3.5) is iteratively minimised until there is no or little change. If the obtained value is less than a certain threshold $J(\mathbf{a}_m, \mathbf{z}_{(n,m)}) \leq \epsilon$, then $\mathbf{a}_m$ can be included in the basis of common information of every matrix from the data ensemble Eq. (3.1). The huge advantage of this framework is its capability of establishing not only strong relations but also weak similarities that are inherent to the multiblock data [73].

The matrix approaches have their pros and cons and are powerful if exploited appropriately, however, they do not account directly for the intrinsic multi-dimensional form of data. To this end, we propose a novel method for common and individual feature extraction which exploits multimodal properties of tensor decompositions.

3.1.3 Common and individual feature extraction

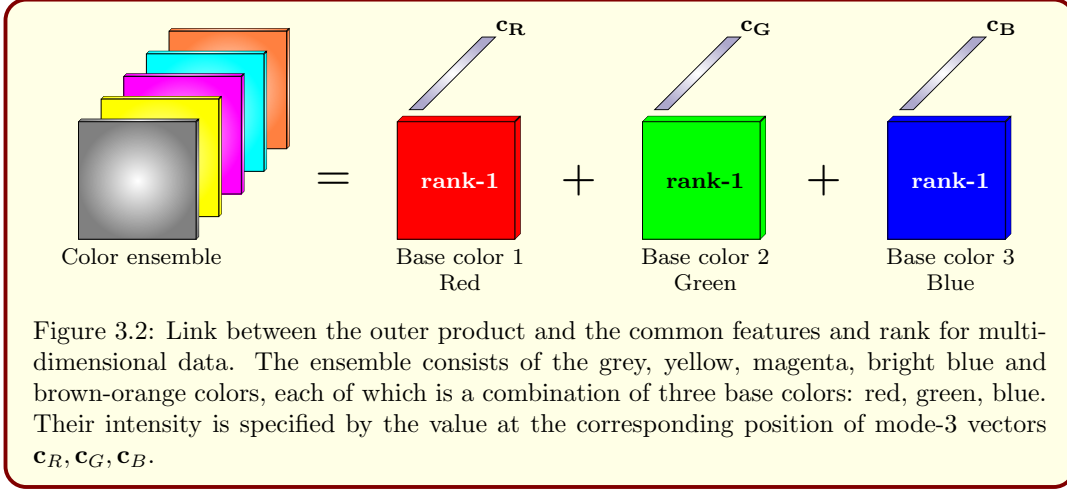
In multi-dimensional algebra, the outer product is the most fundamental operation that increases the order of the base structure, i.e. the outer product of a matrix $\mathbf{A}$ with a vector $\mathbf{c}$ is a three-dimensional array $\underline{\mathbf{X}}$. Another important characteristic of this transformation is that it connects different slices of a tensor $\underline{\mathbf{X}}$ from the same mode and describes how they are related to each other. The proposed framework for common and individual feature analysis is based on the physical meaning of the outer product and intuition behind it is given in the following examples.

Example 1. Observe a rank-1 tensor of order-3, expressed as

$$\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} = \mathbf{Y} \circ \mathbf{c} \quad \text{with } \mathbf{c} = [1 \quad 4 \quad 8]^T \quad (3.6)$$

According to the values of the vector $\mathbf{c}$ and the definition of the outer product, the values in the first frontal slice of $\underline{\mathbf{X}}[:, :, 1]$ are respectively four and eight times smaller than the values in the second $\underline{\mathbf{X}}[:, :, 2]$ and third $\underline{\mathbf{X}}[:, :, 3]$ frontal slices. Hence, each observation stored as the frontal slice of $\underline{\mathbf{X}}$ exhibits the same pattern (base matrix $\mathbf{Y} = \mathbf{a} \circ \mathbf{b}$) that can be considered as a **common feature**. At this point, no individual information can be extracted since there is only one base matrix.

Example 2. Consider a collection, $\underline{\mathbf{X}}$, of five different color matrices stacked along the third dimension, as illustrated in Fig. 3.2. As we discussed in Section 2.3.1, the CPD represents a data ensemble as a combination of the simplest rank-1 structures. Furthermore, the tensor rank of such a 3rd order tensor (color ensemble) is three, that is equivalent to the number of base colors (red, green, blue), which are the simplest structures from which all data can be generated through a mixing matrix $\mathbf{C} = [\mathbf{c}_R, \mathbf{c}_G, \mathbf{c}_B] \in \mathbb{R}^{5 \times 3}$. Thus, adopting the multi-linear notation and the RGB representation of colors, we can write



$$\begin{aligned}
 \underline{\mathbf{X}} &= \underline{\mathbf{A}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \underline{\mathbf{Y}} \times_3 \mathbf{C} \\
 &= \underline{\mathbf{Y}}[:, :, 1] \circ \mathbf{c}_R + \underline{\mathbf{Y}}[:, :, 2] \circ \mathbf{c}_G + \underline{\mathbf{Y}}[:, :, 3] \circ \mathbf{c}_B \\
 \mathbf{c}_R &= \begin{bmatrix} 128 \\ 256 \\ 256 \\ 0 \\ 256 \end{bmatrix} \quad \mathbf{c}_G = \begin{bmatrix} 128 \\ 256 \\ 0 \\ 256 \\ 128 \end{bmatrix} \quad \mathbf{c}_B = \begin{bmatrix} 128 \\ 0 \\ 256 \\ 256 \\ 32 \end{bmatrix}
 \end{aligned} \tag{3.7}$$

Here, $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times 5}$ is the original data, $\mathbf{A} \in \mathbb{R}^{I \times 3}$ and $\mathbf{B} \in \mathbb{R}^{J \times 3}$ are factor matrices that contain spatial information, such as vertical and horizontal position of a pixel respectively, and $\mathbf{C} = [\mathbf{c}_R, \mathbf{c}_G, \mathbf{c}_B] \in \mathbb{R}^{5 \times 3}$ contains the intensity values of the red, green and blue colors. These three base colors are stored in different rank-1 frontal slices of the tensor $\underline{\mathbf{Y}} = \underline{\mathbf{A}} \times_1 \mathbf{A} \times_2 \mathbf{B} \in \mathbb{R}^{I \times J \times 3}$ and represent **common information** among the frontal slices $\underline{\mathbf{X}}[:, :, n]$. The **individual features** can be obtained by subtracting the weighted common features, to give

$$\check{\mathbf{X}}_n = \mathbf{X}_n - \bar{\mathbf{X}} = \underline{\mathbf{X}}[:, :, n] - \sum_{k \in K_n} \alpha_k \underline{\mathbf{Y}}[:, :, k] \tag{3.8}$$

where K_n is a subset of common features for $\mathbf{X}_n$ with respect to the values in the n -th row of $\mathbf{C}$.

Remark 11. The common features from Eq. (3.4) are represented as column vectors of $\bar{\mathbf{X}}$, whereas the model Eq. (3.7) depicts the mutual information in the form of a set of matrices $\bar{\mathbf{X}}_k = \underline{\mathbf{Y}}[:, :, k]$ which is more descriptive.

Remark 12. For the illustration of the proposed approach, we used a tensor of order three, however, unlike matrices the proposed approach generalises well and allows for the common and individual features to be extracted from a tensor of any order, with only one requirement that observations must be concatenated along the same mode.

3.1.4 The LL1 decomposition with non-negativity constraint

In spite of the CPD in (2.40) and TKD in (2.55) being very powerful, unique, and compact representations of the data collected during any modern experiment, they are based on the assumption that an underlying structure exhibits rank-1 properties. The Block Term Decomposition (BTD) overcomes this restriction and allows us to reproduce components with more

3. Feature Extraction and Data Fusion

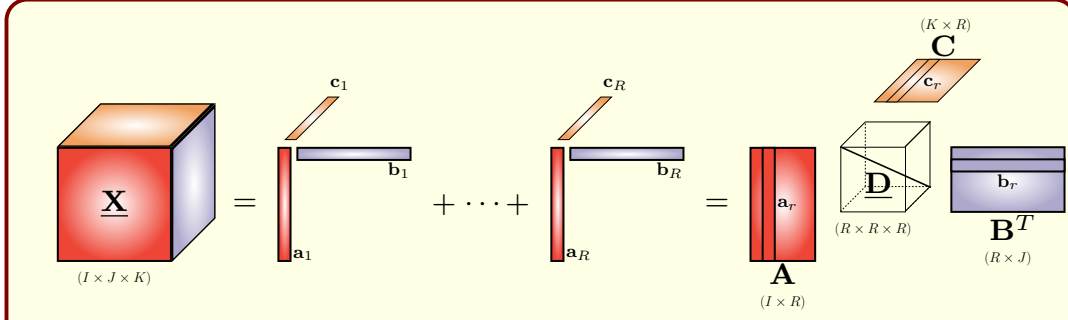


Figure 3.3: Interchangeable definitions of the Canonical Polyadic Decomposition. The original tensor $\underline{\mathbf{X}}$ is represented as a sum of rank-1 tensors $\underline{\mathbf{X}}_r = \lambda_r \cdot \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ and as mode-n product of the core tensor $\underline{\mathbf{A}}$ with the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

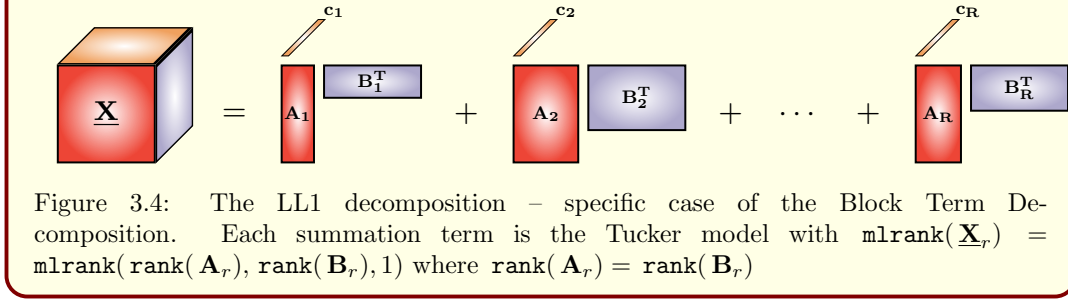


Figure 3.4: The LL1 decomposition – specific case of the Block Term Decomposition. Each summation term is the Tucker model with $\text{mlrank}(\underline{\mathbf{X}}_r) = \text{mlrank}(\text{rank}(\mathbf{A}_r), \text{rank}(\mathbf{B}_r), 1)$ where $\text{rank}(\mathbf{A}_r) = \text{rank}(\mathbf{B}_r)$

complex dependencies [74]. Ultimately the BTM comprises the TKD and CPD in the sense that it decomposes the multiway information as a sum of the terms from (2.57):

$$\begin{aligned} \underline{\mathbf{X}} &= \sum_{r=1}^R \underline{\mathbf{X}}_r = \sum_{r=1}^R \underline{\mathbf{G}}_r \times_1 \mathbf{A}_r \times_2 \mathbf{B}_r \times_3 \mathbf{C}_r \\ &= \sum_{r=1}^R \llbracket \underline{\mathbf{G}}_r; \mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r \rrbracket \end{aligned} \quad (3.9)$$

Here each set of $\underline{\mathbf{G}}_r, \mathbf{A}_r, \mathbf{B}_r$ and $\mathbf{C}_r$ assume the Tucker model with a unique range for the multi-linear rank:

$$\text{mlrank}(\underline{\mathbf{G}}_n) \neq \text{mlrank}(\underline{\mathbf{G}}_k) \text{ for } n \neq k \quad (3.10)$$

To make use of the desirable properties of both CPD and TKD and increase structural complexity of the common features, a specific case of the BTM is of particular interest, namely the LL1 decomposition [75, 76]. It efficiently combines both concepts, by decomposing the tensor $\underline{\mathbf{X}}$ into a linear combination of K tensors, whereby each term $\underline{\mathbf{X}}_r = \llbracket \underline{\mathbf{A}}_r; \mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r \rrbracket$ has a multilinear rank $(L_r, L_r, 1)$, that is

$$\underline{\mathbf{X}} = \sum_{r=1}^R \underline{\mathbf{X}}_r = \sum_{r=1}^R (\mathbf{A}_r \mathbf{B}_r^T) \circ \mathbf{c}_r = \sum_{r=1}^R \mathbf{G}_r \circ \mathbf{c}_r \quad (3.11)$$

The LL1 decomposition for an order-3 tensor is illustrated on Fig. 3.4, where $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, whereby the “one to one” relation between the factor matrices $\mathbf{A}_r \in \mathbb{R}^{I \times L_r}$, $\mathbf{B}_r \in \mathbb{R}^{J \times L_r}$, and factor vector $\mathbf{c}_r \in \mathbb{R}^K$ is preserved. Moreover, upon employing the matrix-tensor duality, we can represent the matrix $\mathbf{G}_r \in \mathbb{R}^{I \times J}$ as a tensor of order three, $\hat{\mathbf{G}}_r \in \mathbb{R}^{I \times J \times 1}$.

3. Feature Extraction and Data Fusion

Remark 13. The matrix $\mathbf{G}_r$ in (3.11) is no longer of rank-1 and is consequently more informative. However, the so-obtained tensor, $\underline{\mathbf{X}}_r = \mathbf{G}_r \circ \mathbf{c}_r$, is still considered to exhibit the simplest structure as far as the LL1 decomposition is concerned.

If a slice $\underline{\mathbf{Y}}[:, :, k]$ from Eq. (3.8) belongs to the set of common features for the data sample $\underline{\mathbf{X}}[:, :, n]$, then an intuitive implication is that the corresponding value of $\mathbf{C}_r[n, k]$ is positive. However, this cannot be guaranteed for a general implementation of the LL1 decomposition, and the non-negativity constraint should be imposed on the factor matrices $\mathbf{C}_r$, since they correspond to the mode along which members of the ensemble are stacked together. Thus, the extraction of the common features from the LL1 decomposition is given by:

$$\bar{\mathbf{X}}_r = \mathbf{A}_r \mathbf{B}_r^T \quad (3.12)$$

and requires the minimization of the cost function

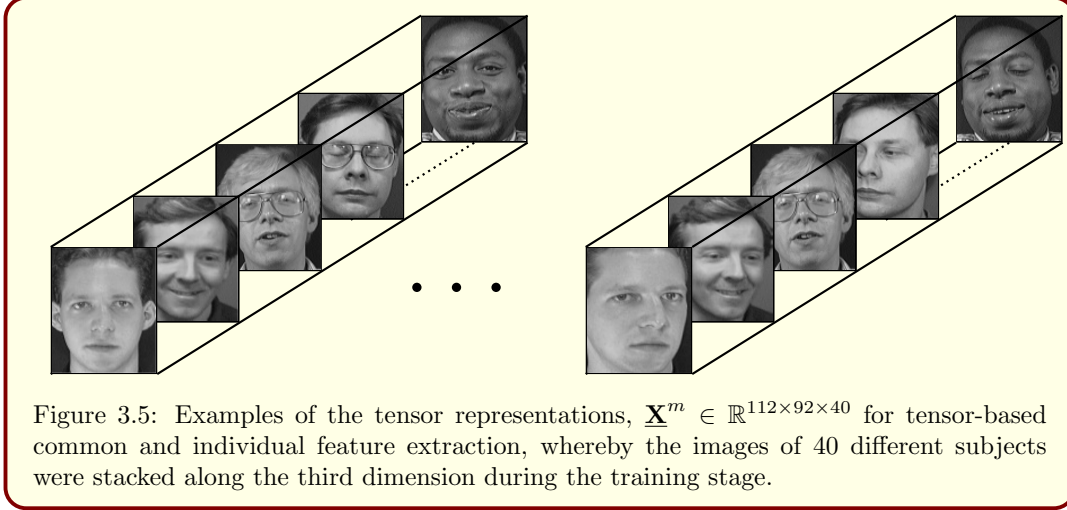
$$\begin{aligned} J(\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r) &= \left\| \underline{\mathbf{X}} - \sum_{r=1}^R \hat{\underline{\mathbf{X}}}_r \right\|_F^2 \\ &= \left\| \underline{\mathbf{X}} - \sum_{r=1}^R [\underline{\mathbf{A}}_r; \mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r] \right\|_F^2 \\ \text{s.t. } \mathbf{c}_r &> 0 \end{aligned} \quad (3.13)$$

Notice that this problem is similar to the computation of the CPD in (2.52). Therefore, our solution is based on the CPD-ALS algorithm and is summarised in Algorithm 5, where the routine `NonNegLeastSq`($\mathbf{X}, \mathbf{Y}$) performs least squares estimation an input $\mathbf{X}$ and an output $\mathbf{Y}$, whereby the least squares coefficients are constrained to be non-negative [77], while `Repeat`($\mathbf{x}, n$) duplicates an input $\mathbf{X}$ n times.

Algorithm 5: LL1 decomposition of a 3rd-order tensor with non-negative constraint

Input : $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ and R sets of multi-linear tensor rank $(L_r, L_r, 1)$
Output: Factor matrices $\mathbf{A}_r \in \mathbb{R}^{I \times L_r}, \mathbf{B}_r \in \mathbb{R}^{J \times L_r}, \mathbf{c}_r \in \mathbb{R}^K$, and scaling vectors $\lambda_r \in \mathbb{R}^{L_r}$

- 1 Initialize factor matrices $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r\}$
- 2 **while** not converged or iteration limit is not reached **do**
- 3 **for** $r = 1, \dots, R$ **do**
- 4 $\underline{\mathbf{X}} = \underline{\mathbf{X}} - \sum_{n=1, n \neq r}^N [\underline{\mathbf{A}}_n; \mathbf{A}_n, \mathbf{B}_n, \mathbf{c}_n]$
- 5 $\mathbf{C}_r = \text{Repeat}(\mathbf{c}_r, L_r)$
- 6 $\hat{\mathbf{A}}_r = \underline{\mathbf{X}}_{(1)}(\mathbf{C}_r \odot \mathbf{B}_r)(\mathbf{C}_r^T \mathbf{C}_r \otimes \mathbf{B}_r^T \mathbf{B}_r)^\dagger$
- 7 $\hat{\mathbf{B}}_r = \underline{\mathbf{X}}_{(2)}(\mathbf{C}_r \odot \mathbf{A}_r)(\mathbf{C}_r^T \mathbf{C}_r \otimes \mathbf{A}_r^T \mathbf{A}_r)^\dagger$
- 8 $\hat{\mathbf{c}}_r = \text{NonNegLeastSq}(\mathbf{A} \odot \mathbf{B}, \underline{\mathbf{X}}_{(3)})$
- 9 Normalise each column of $\hat{\mathbf{A}}_r, \hat{\mathbf{B}}_r$ and $\hat{\mathbf{c}}_r$ to unit length and store the norms in λ_r
- 10 Assign $\hat{\mathbf{A}}_r \rightarrow \mathbf{A}_r; \hat{\mathbf{B}}_r \rightarrow \mathbf{B}_r; \hat{\mathbf{c}}_r \rightarrow \mathbf{c}_r$
- 11 **end for**
- 12 **end while**
- 13 **return** $\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r$ and λ_r



3.1.5 Experimental results and discussion

General overview of the classification setup

The proposed approach was employed for the classification of face images from the benchmark ORL dataset [78]. This database includes a total of 400 grey scale images of 40 subjects in ten different illumination conditions and with different facial expressions. Ten sets of 40 images were created by randomly choosing one image of every subject. Six of these sets were arbitrarily selected for the training set with the remaining four forming the test set. Each group from the training set was represented as a tensor, $\underline{\mathbf{X}}^m \in \mathbb{R}^{112 \times 92 \times 40}$, where the images of 40 different subjects were stacked along the third dimension, as in Fig. 3.5. Their individual features were extracted by applying the proposed framework with the non-negativity constraint imposed on the mode-3 factor matrix, while the number of common features was found empirically. The classification models used were Support Vector Machine (SVM) with Gaussian kernel, feed-forward Neural Network (NN) with two hidden layers, cosine distance based k-Nearest Neighbours (cKNN) and Quadratic Discriminant (QD) and were trained on the so obtained individual information. The classification scores were calculated from 100 realisations. Note that during the test stage, we used the original images in order to make a fair and realistic evaluation.

Feature extraction and classification

Consider a training set that consists of $\mathbf{X}^{MN} \in \mathbb{R}^{112 \times 92}$ images from N different classes, but is split into M groups with one sample from every class $\mathcal{X}^m = \{\mathbf{X}_n^m \in \mathbb{R}^{112 \times 92}; m = 1, \dots, M; n = 1, \dots, N\}$. After stacking the images within each training group $\mathcal{X}$ into a tensor $\underline{\mathbf{X}}^m \in \mathbb{R}^{112 \times 92 \times N}$ such that $\underline{\mathbf{X}}^m[:, :, n] = \mathbf{X}_n^m$, we arrive at a point when common features of training groups and, consecutively, individual features of data samples can be identified by the virtue of tensor factorisation with the non-negativity constraint imposed along the third mode of $\underline{\mathbf{X}}^m$. Thus, for the CPD based algorithm this can be accomplished by:

$$\underline{\mathbf{X}}^m = \underline{\mathbf{A}}^m \times_1 \mathbf{A}^m \times_2 \mathbf{B}^m \times_3 \mathbf{C}^m \quad (3.14)$$

where the set of common features, $\underline{\mathbf{Y}}^m$, is expressed as:

$$\begin{aligned} \underline{\mathbf{Y}}^m &= \underline{\mathbf{A}}^m \times_1 \mathbf{A}^m \times_2 \mathbf{B}^m \\ \bar{\mathbf{X}}_r^m &= \underline{\mathbf{Y}}^m[:, :, r] \end{aligned} \quad (3.15)$$

while individual information intrinsic to a particular sample, $\mathbf{X}_n^m$, is obtained with:

3. Feature Extraction and Data Fusion

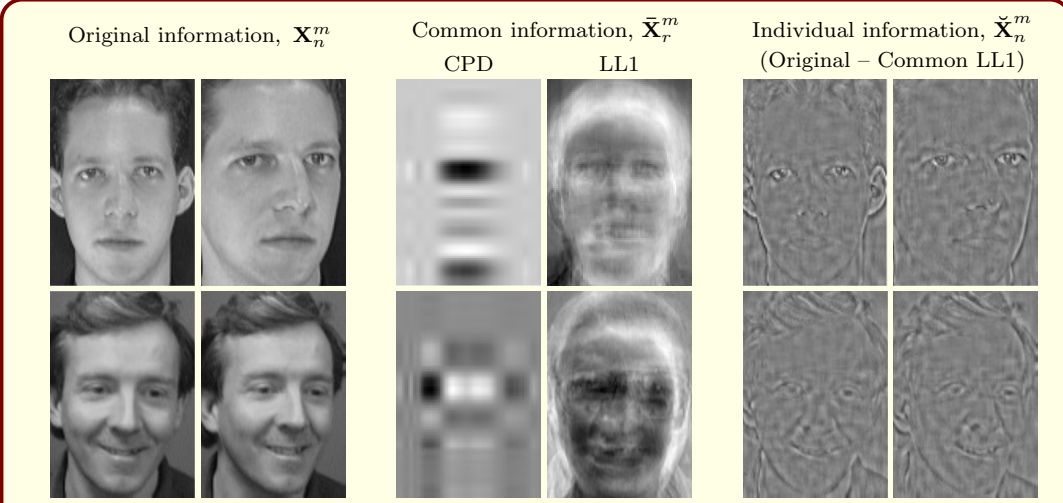


Figure 3.6: Common and individual feature extraction. Left: Examples of images in the ORL dataset. Center: Examples of common features computed through the CPD and LL1 decompositions. Right: Examples of extracted individual features obtained by subtraction of the common information from the corresponding original images.

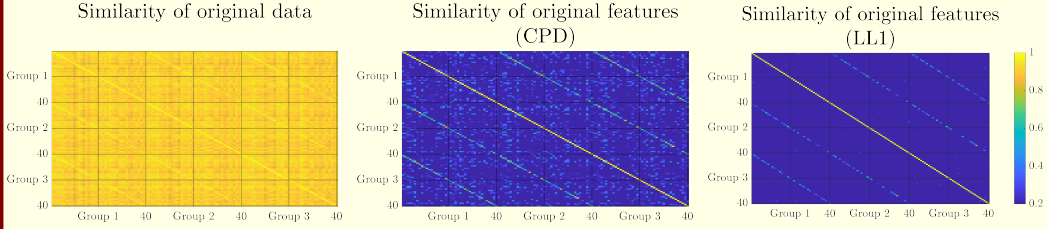


Figure 3.7: Cosine similarity between samples in the dataset before (left) and after (center, right) individual information has been extracted. Each group of 40 randomly selected samples was processed separately. The yellow and blue colors stand respectively for the highest and lowest similarity level.

$$\begin{aligned}\check{\mathbf{X}}_n^m &= \mathbf{X}_n^m - \mathbf{C}^m[n, r] \bar{\mathbf{X}}_r^m \\ &= \mathbf{X}_n^m - \mathbf{C}^m[n, r] \underline{\mathbf{Y}}^m[:, :, r]\end{aligned}\quad (3.16)$$

In contrast to the CPD, the procedure of feature extraction for an algorithm based on the LL1 decomposition is slightly different due to structural differences between (3.11) and (2.40). For instance, common and individual information is given by:

$$\begin{aligned}\bar{\mathbf{X}}_r^m &= \mathbf{A}_r^m (\mathbf{B}_r^m)^T \\ \check{\mathbf{X}}_n^m &= \mathbf{X}_n^m - \mathbf{c}_r^m[n] \bar{\mathbf{X}}_r^m \\ &= \mathbf{X}_n^m - \mathbf{c}_r^m[n] \mathbf{A}_r^m (\mathbf{B}_r^m)^T\end{aligned}\quad (3.17)$$

where Algorithm 5 is employed to obtain $\mathbf{A}_r^m$, $\mathbf{B}_r^m$ and $\mathbf{c}_r^m$. Fig. 3.6 illustrates examples of images used in the experiments, and the extracted individual information and common features for the CPD and LL1 decomposition.

Remark 14. Both the CPD and LL1 decomposition seek common features across all samples. However, similarly to example illustrated on Fig. 3.2, common information might be insignificant

3. Feature Extraction and Data Fusion

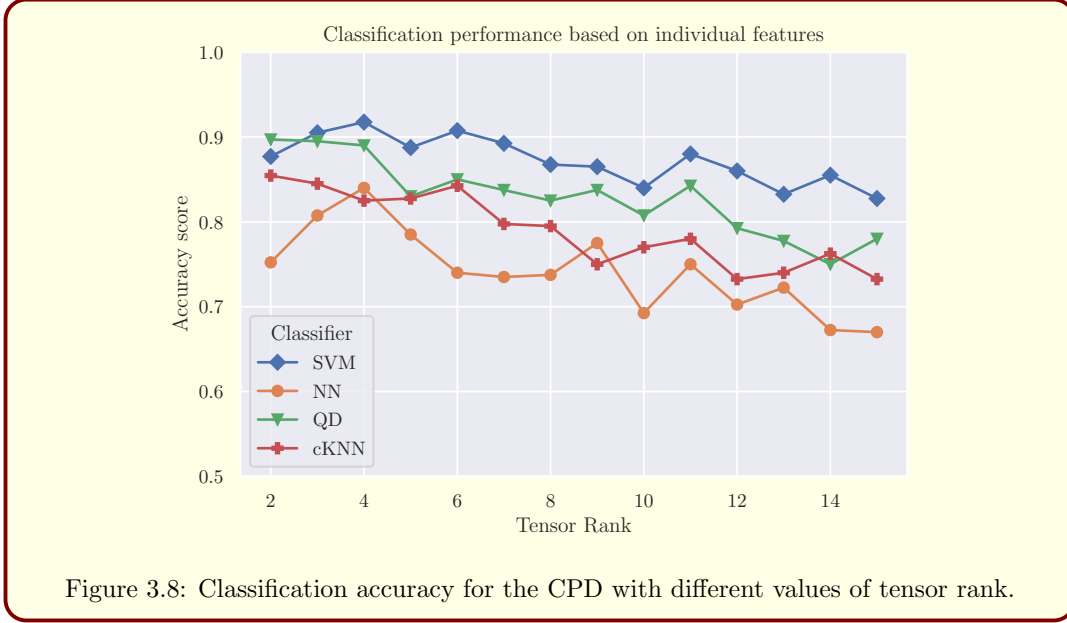


Figure 3.8: Classification accuracy for the CPD with different values of tensor rank.

for some of the data samples, $\mathbf{X}_n^m$, within considered group due to the flexibility of the TDs. Thus, a threshold should be applied to the values of $\mathbf{C}^m$ and $\mathbf{c}_r^m$ that determines influence level of a common feature on a given $\mathbf{X}_n^m$. Such value can be considered as a hyperparameter for the overall classification model.

It is vital for successful classification that the samples should exhibit similar patterns, if they belong to the same category. In other words, if we were to look at the individual information of images of different subjects from the first group, then it would be distinct for every member of this ensemble, while the individual features of the same subject but from several different groups would be correlated.

Fig. 3.7 depicts the degree of cosine similarity between some samples used at the training stage of the classification. Observe that all members of the original dataset are highly correlated with one another, whereby our algorithm reveals underlying patterns that are shared by the same subject across different groups, despite being estimated separately. This confirms that the concept of individual feature extraction is a promising tool that can be exploited as part of the training set preprocessing.

In order to further evaluate the proposed framework, four different classifiers were employed: SVM, NN, QD and cKNN. The learning coefficients for these models were estimated based on the 10-fold cross-validation. Their performance on multi-class classification of the original images and the corresponding individual information, extracted through the proposed method for the CPD and LL1 decompositions, is summarised in the Table 1. These scores were obtained by averaging 100 independent simulations with random split of the whole database into the training and test sets in each iteration.

As expected, the classification rates of the SVM, NN and cKNN have increased due to the common information being removed from the dataset. The most significant improvement can be observed for the NN classifier. Here, the poor accuracy on the original data is associated with the high variance of the original samples and the small size of the training set which resulted in overfitting. On the other hand, the extracted individual features were of lower variance, which allowed the NN classifier to find a decision boundary that is less prone to fluctuations in the training data, leading to much higher classification accuracy.

Next, we evaluated performance of the classification algorithms on individual information obtained through the CPD with respect to different values of the tensor rank, as shown in

3. Feature Extraction and Data Fusion

Table 3.1: Classification Performance in %				
	SVM	NN	QD	cKNN
Original	83.9	4.35	91.5	79.0
CPD	91.5	85.8	89.8	85.5
LL1	94.7	92.2	86.8	84.3

Fig. 3.8. As discussed previously, the maximum number of the common components is linked to the tensor rank. Bearing in mind that the CPD is merely a technique to approximate the multi-dimensional array, the tensor rank could be set to any arbitrarily big number. Thus, more common features are found as the tensor rank increases. Consequently, classification models would suffer from high bias of training data set and their accuracy would noticeably decrease.

3.2 Feature Fusion via Tensor Network Summation

3.2.1 Introduction

Any tensor decomposition can be considered as a special case of the more general concept of a tensor network (TN), which represents a high order tensor as a set of sparsely interconnected small scale core tensors and the associated factor matrices [1]. In other words, TNs can be viewed as multi-core interconnections of features of the original tensor. The advantages of representing a tensor as a TN are:

1. TNs are perfectly suited to deal with the Curse of Dimensionality, as a high order tensor, represented as a TN, can be stored on different machines which deal with only the individual cores;
2. Each core may be representative of a specific characteristic of the underlying tensor, thus implying inherent feature extraction from the original data.

Despite these advantages, open problems in practical design of TNs still include:

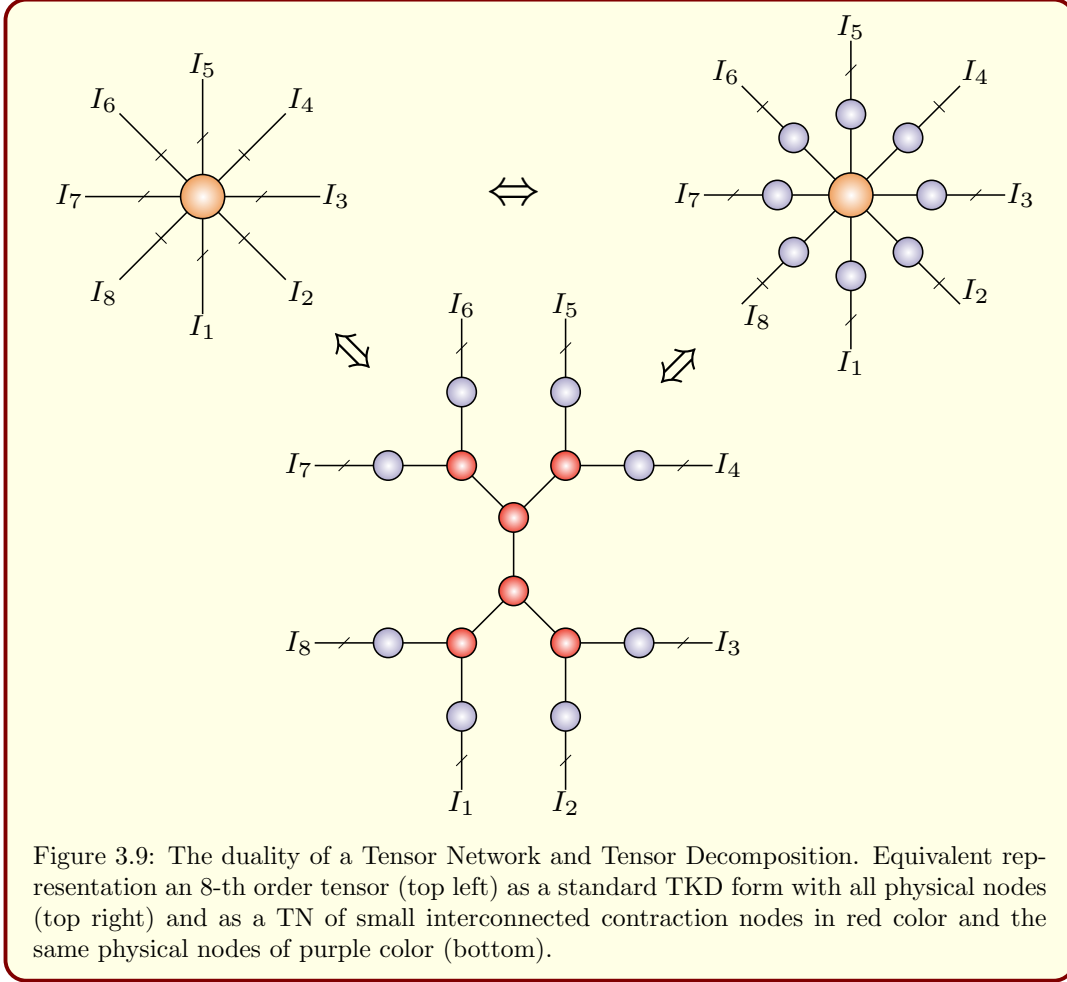
1. The choice of TD for a particular application;
2. Minimisation of the number of the parameters necessary for a TN representation;
3. A rigorous framework to combine TNs.

We address the last issue by introducing a TN summation operator for TNs of the same topology. Summation is the most natural way to reduce the number of entities, and we show that a sum of multiple tensors (summands), in their TN format, preserves their underlying structure. In this way, the sum of TNs yields another isomorphic TN, the cores of which are a combination of the corresponding cores of the summand TNs. It is shown that in the TN context, the physical interpretation of summation represents feature fusion of the original tensors in the raw format and provides a starting point for new theoretical avenues in TN applications. In particular, the proposed framework opens up new perspectives on algebraic manipulation of TNs, and represents a first step towards establishing their taxonomy, thus removing the preconception that they have to be treated as stand-alone entities.

3.2.2 Summation framework

To establish a framework for the summation of two or more TNs, the following assumptions are made about the summand TNs:

1. They exhibit the same topology;



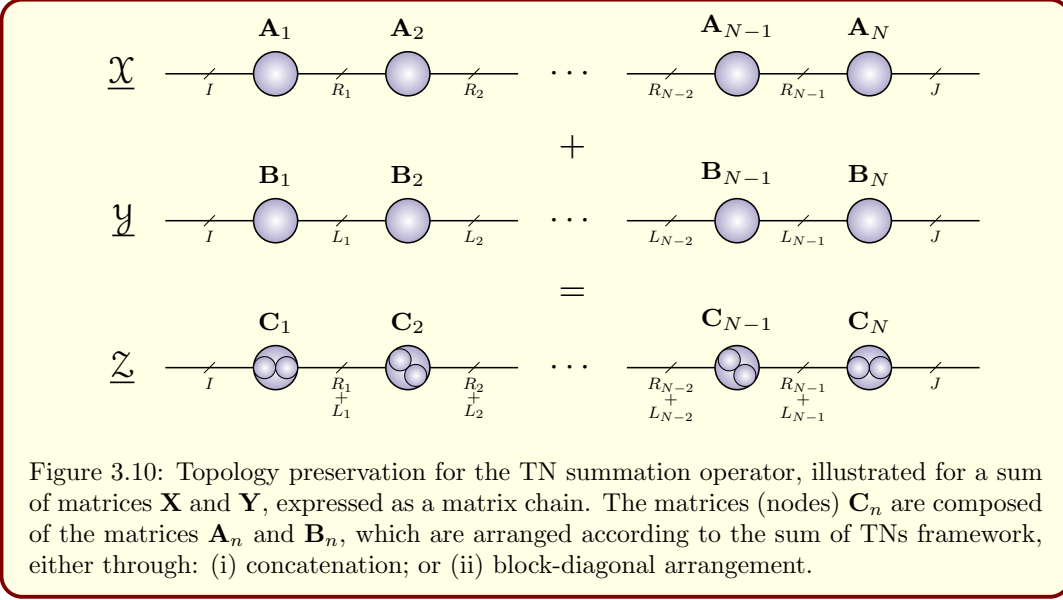
2. The physical modes of the summands are of equal dimensions.

Fig. 3.9 illustrates an example of transition from a multi-dimensional array, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_8}$, to tensor factorisation and consequently to TNs (denoted as underlined Euler script letter, e.g. $\underline{\mathcal{X}}$), whereby any mode which is not a *physical mode*, I_n , is a *contraction mode*, R_n . Similarly, nodes connected to one or more physical modes are referred to as physical nodes, while the rest are contraction nodes. Thus, each node in $\underline{\mathcal{X}}$ can represent either a particular feature of the original tensor $\underline{\mathbf{X}}$ (in case of physical nodes), or a model of how features are combined (in case of contraction nodes). The “shape” of a TN represents its topology, where the concept of topology is the same as that adopted in Graph Theory [79].

Remark 15. We refer to any node within a TN as “core” when treating it as a tensor, while in a topological context, we use the term “node”.

It is clear that a framework for combining TNs is a prerequisite to mixing features of individual cores, which would then allow the extraction of the common features across the individual tensors. Yet, the “feature locality” inherent to the nodes of a TN is of fundamental importance, but is still largely under-explored. Therefore, the main motivation for this work has been to establish the missing framework for TN summation via a combination of their corresponding cores, thus simultaneously performing a feature fusion.

3. Feature Extraction and Data Fusion



Definition 3.1. Consider two tensors $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ represented as TNs denoted by $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ which have equivalent topologies, but not necessarily the same dimensionality of contraction modes. The sum $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ can then be thought of as a new TN, $\underline{\mathbf{Z}}$, with an equivalent topology to $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$. Its contraction nodes are in the form of a block tensor which is obtained by stacking the corresponding contraction nodes of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ along its superdiagonal. The physical nodes of $\underline{\mathbf{Z}}$ are obtained by arranging the corresponding physical nodes of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ in such a way that the dimensionality of all contracting modes is increased but the dimensionality of the physical modes is kept fixed.

Proposition 1. The proposed framework for a sum of TNs is valid for chains of matrices and the associated graphical representation is depicted on Fig. 3.10.

Proof. Suppose $\mathbf{X} = \mathbf{A}_1 \mathbf{A}_2 \dots \mathbf{A}_N$, $\mathbf{Y} = \mathbf{B}_1 \mathbf{B}_2 \dots \mathbf{B}_N$, where $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{I \times J}$, and $\mathbf{A}_n, \mathbf{B}_n \in \mathbb{R}^{R_{n-1} \times R_n}$, with $R_0 = I, R_N = J$. Define a new chain of matrices $\mathbf{Z} = \mathbf{C}_1 \mathbf{C}_2 \dots \mathbf{C}_N$, where each $\mathbf{C}_n$, $n = 1, \dots, N$, is an arrangement of $\mathbf{A}_n, \mathbf{B}_n$ according to the proposed framework:

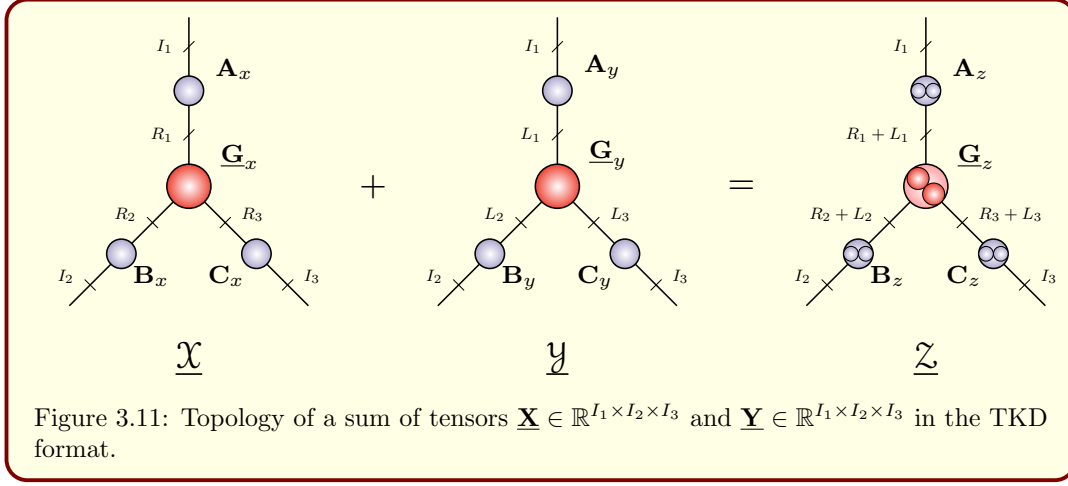
$$\begin{aligned} \mathbf{C}_1 &= \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \end{bmatrix} \\ \mathbf{C}_n &= \begin{bmatrix} \mathbf{A}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_n \end{bmatrix} \quad \text{for } n = 2, \dots, N-1 \\ \mathbf{C}_N &= \begin{bmatrix} \mathbf{A}_N \\ \mathbf{B}_N \end{bmatrix} \end{aligned} \quad (3.18)$$

Then by a direct inspection of $\mathbf{Z}$, we have:

$$\begin{aligned} \mathbf{Z} &= \mathbf{C}_1 \mathbf{C}_2 \dots \mathbf{C}_N \\ &= \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2 \end{bmatrix} \dots \begin{bmatrix} \mathbf{A}_{N-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}_N \\ \mathbf{B}_N \end{bmatrix} \\ &= \mathbf{A}_1 \mathbf{A}_2 \dots \mathbf{A}_N + \mathbf{B}_1 \mathbf{B}_2 \dots \mathbf{B}_N \\ &= \mathbf{X} + \mathbf{Y} \end{aligned} \quad (3.19)$$

□

3. Feature Extraction and Data Fusion



The above result for matrices is well-known, and serves here as a specific and straightforward intuition behind the general concept of TN summation.

Proposition 2. *The TN summation operator is valid for any multi-dimensional array expressed through a tensor factorisation.*

Proof. For simplicity, we here provide proof for 3-rd order tensors, but without loss of generality the result holds for sum of tensors of any order. Fig. 3.11 shows the tensors $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format, expressed as

$$\begin{aligned}\underline{\mathbf{X}} &= \underline{\mathbf{G}}_x \times_1 \mathbf{A}_x \times_2 \mathbf{B}_x \times_3 \mathbf{C}_x \\ \underline{\mathbf{Y}} &= \underline{\mathbf{G}}_y \times_1 \mathbf{A}_y \times_2 \mathbf{B}_y \times_3 \mathbf{C}_y\end{aligned}\tag{3.20}$$

with the respective TN representations, $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$. A new TN, $\underline{\mathbf{Z}}$, is then obtained by combining $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ according to the proposed framework, and the so generated tensor, $\underline{\mathbf{Z}}$, can be described by

$$\underline{\mathbf{Z}} = \underline{\mathbf{G}}_z \times_1 \mathbf{A}_z \times_2 \mathbf{B}_z \times_3 \mathbf{C}_z\tag{3.21}$$

where, new factor matrices $\mathbf{A}_z, \mathbf{B}_z$ and $\mathbf{C}_z$ are obtained by concatenating $\mathbf{A}_x, \mathbf{A}_y; \mathbf{B}_x, \mathbf{B}_y$ and $\mathbf{C}_x, \mathbf{C}_y$ respectively, while core tensors $\underline{\mathbf{G}}_x$ and $\underline{\mathbf{G}}_y$ are placed on the superdiagonal of $\underline{\mathbf{G}}_z$. Next, we show that $\underline{\mathbf{Z}} = \underline{\mathbf{X}} + \underline{\mathbf{Y}}$, based on the mode-1 unfolding, however, note that the same procedure can be applied to any mode. Define $\mathbf{A}_z = [\mathbf{A}_x \ \mathbf{A}_y]$, and $\underline{\mathbf{G}}_z$ as an arrangement of $\underline{\mathbf{G}}_x$ and $\underline{\mathbf{G}}_y$ along the superdiagonal of $\underline{\mathbf{G}}_z$, and consider

$$\begin{aligned}\mathbf{K}_x &= (\mathbf{C}_x \otimes \mathbf{B}_x)^T \\ \mathbf{K}_y &= (\mathbf{C}_y \otimes \mathbf{B}_y)^T\end{aligned}\tag{3.22}$$

Upon performing the mode-1 unfolding of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ according to (2.22), and combining the matrices $\underline{\mathbf{X}}_{(1)}, \underline{\mathbf{Y}}_{(1)}$ in their TN formats, then from Proposition 1 the resulting matrix, $\mathbf{Z}_*$, can be expressed as

$$\mathbf{Z}_* = [\mathbf{A}_x \ \mathbf{A}_y] \begin{bmatrix} \underline{\mathbf{G}}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{G}}_{y(1)} \end{bmatrix} [\mathbf{K}_x \ \mathbf{K}_y]^T\tag{3.23}$$

Therefore, in order to prove Proposition 2 it is sufficient to show that $\underline{\mathbf{Z}}_{(1)} = \mathbf{Z}_*$, where $\underline{\mathbf{Z}}_{(1)}$ is the mode-1 unfolding of $\underline{\mathbf{Z}}$, represented as in (3.21). To this end, consider

3. Feature Extraction and Data Fusion

$$\begin{aligned}\underline{\mathbf{Z}}_{(1)} &= \mathbf{A}_z \underline{\mathbf{G}}_{z(1)} (\mathbf{C}_z \otimes \mathbf{B}_z)^T \\ &= [\mathbf{A}_x \quad \mathbf{A}_y] \underline{\mathbf{G}}_{z(1)} \left([\mathbf{C}_x \quad \mathbf{C}_y] \otimes [\mathbf{B}_x \quad \mathbf{B}_y] \right)^T\end{aligned}\quad (3.24)$$

For convenience, assume $\underline{\mathbf{X}}, \underline{\mathbf{Y}}, \underline{\mathbf{G}}_x, \underline{\mathbf{G}}_y \in \mathbb{R}^{2 \times 2 \times 2}$, and denote slices $\underline{\mathbf{G}}_\alpha[:, :, j] = \hat{\mathbf{G}}_\alpha[j]$, where $\alpha \in \{x, y\}$. Hence,

$$\underline{\mathbf{G}}_{z(1)} = \begin{bmatrix} \hat{\mathbf{G}}_x[1] & \mathbf{0} & \hat{\mathbf{G}}_x[2] & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y[1] & \mathbf{0} & \hat{\mathbf{G}}_y[2] \end{bmatrix}\quad (3.25)$$

Without loss of generality, assume the concrete values

$$\mathbf{C}_x = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \quad \mathbf{C}_y = \begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix}\quad (3.26)$$

to give

$$\begin{aligned}[\mathbf{C}_x \quad \mathbf{C}_y] \otimes [\mathbf{B}_x \quad \mathbf{B}_y] &= \\ &= \begin{bmatrix} 1 [\mathbf{B}_x \quad \mathbf{B}_y] & 2 [\mathbf{B}_x \quad \mathbf{B}_y] & 3 [\mathbf{B}_x \quad \mathbf{B}_y] & 4 [\mathbf{B}_x \quad \mathbf{B}_y] \\ 5 [\mathbf{B}_x \quad \mathbf{B}_y] & 6 [\mathbf{B}_x \quad \mathbf{B}_y] & 7 [\mathbf{B}_x \quad \mathbf{B}_y] & 8 [\mathbf{B}_x \quad \mathbf{B}_y] \end{bmatrix} \\ &= \mathbf{U}\end{aligned}\quad (3.27)$$

Upon substituting $\underline{\mathbf{G}}_{z(1)}$ and $\mathbf{U}$ into (3.24) and making use of the sparse nature of (3.25), we arrive at

$$\begin{aligned}\underline{\mathbf{Z}}_{(1)} &= \mathbf{A}_z \underline{\mathbf{G}}_{z(1)} \mathbf{U}^T \\ &= [\mathbf{A}_x \quad \mathbf{A}_y] \begin{bmatrix} \hat{\mathbf{G}}_x[1] & \hat{\mathbf{G}}_x[2] & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y[1] & \hat{\mathbf{G}}_y[2] \end{bmatrix} \begin{bmatrix} 1\mathbf{B}_x^T & 5\mathbf{B}_x^T \\ 2\mathbf{B}_x^T & 6\mathbf{B}_x^T \\ 3\mathbf{B}_y^T & 7\mathbf{B}_y^T \\ 4\mathbf{B}_y^T & 8\mathbf{B}_y^T \end{bmatrix} \\ &= [\mathbf{A}_x \quad \mathbf{A}_y] \begin{bmatrix} \underline{\mathbf{G}}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{G}}_{y(1)} \end{bmatrix} \begin{bmatrix} \mathbf{K}_x^T \\ \mathbf{K}_y^T \end{bmatrix} \\ &= [\mathbf{A}_x \quad \mathbf{A}_y] \begin{bmatrix} \underline{\mathbf{G}}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{G}}_{y(1)} \end{bmatrix} [\mathbf{K}_x \quad \mathbf{K}_y]^T \\ &= \mathbf{Z}_*\end{aligned}\quad (3.28)$$

□

Proposition 3. *The TN summation framework is valid for tensor trains. An illustration of this concept for two 3-rd order tensors expressed in TT format is provided on Fig. 3.12.*

Proof. In order to make equations and derivations more tractable, without loss of generality we consider two 3-rd order tensors of low dimensionality $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{2 \times 2 \times 2}$ and represent them in TT format in accordance with (2.64):

$$\begin{aligned}\underline{\mathbf{X}} &= \underline{\mathbf{G}}_x^{(1)} \times_3^1 \underline{\mathbf{G}}_x^{(2)} \times_3^1 \underline{\mathbf{G}}_x^{(3)} \\ \underline{\mathbf{Y}} &= \underline{\mathbf{G}}_y^{(1)} \times_3^1 \underline{\mathbf{G}}_y^{(2)} \times_3^1 \underline{\mathbf{G}}_y^{(3)}.\end{aligned}\quad (3.29)$$

3. Feature Extraction and Data Fusion

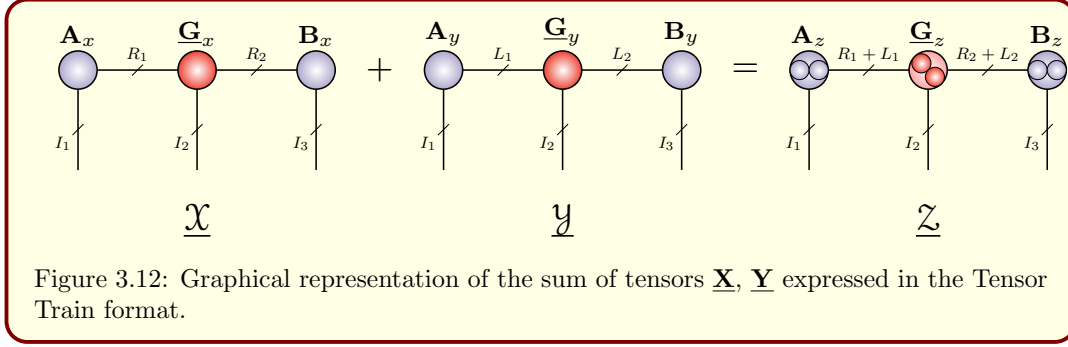


Figure 3.12: Graphical representation of the sum of tensors $\underline{\mathbf{X}}$, $\underline{\mathbf{Y}}$ expressed in the Tensor Train format.

Taking into account tensor matrix duality:

$$\begin{aligned} \underline{\mathbf{G}}_n^{(1)} \in \mathbb{R}^{1 \times I_n \times R_n} &\rightarrow \mathbf{A}_n \in \mathbb{R}^{I_n \times R_n} && \text{for } n \in [x, y] \\ \underline{\mathbf{G}}_m^{(3)} \in \mathbb{R}^{R_m \times I_m \times 1} &\rightarrow \mathbf{B}_m^T \in \mathbb{R}^{R_m \times I_m} && \text{for } m \in [x, y] \end{aligned} \quad (3.30)$$

we can simplify notation from (3.29) to:

$$\begin{aligned} \underline{\mathbf{X}} &= \mathbf{A}_x \times_2^1 \underline{\mathbf{G}}_x \times_3^1 \mathbf{B}_x^T \\ \underline{\mathbf{Y}} &= \mathbf{A}_y \times_2^1 \underline{\mathbf{G}}_y \times_3^1 \mathbf{B}_y^T. \end{aligned} \quad (3.31)$$

By employing the relation between the contraction product and the mode- n product from (2.25) and (2.26), we arrive at:

$$\begin{aligned} \underline{\mathbf{X}} &= \mathbf{A}_x \times_2^1 \underline{\mathbf{G}}_x \times_3^1 \mathbf{B}_x^T \\ &= \mathbf{A}_x \times_2^1 (\underline{\mathbf{G}}_x \times_3 \mathbf{B}_x) \times_1^1 \mathbf{I} \\ &= (\underline{\mathbf{G}}_x \times_3 \mathbf{B}_x) \times_1^1 \mathbf{I} \times_2^1 \mathbf{A}_x \\ &= (\underline{\mathbf{G}}_x \times_3 \mathbf{B}_x) \times_1^1 (\mathbf{I} \times_2^1 \mathbf{A}_x) \\ &= \underline{\mathbf{G}}_x \times_3 \mathbf{B}_x \times_1 \mathbf{A}_x \\ &= \underline{\mathbf{G}}_x \times_1 \mathbf{A}_x \times_2 \mathbf{I} \times_3 \mathbf{B}_x. \end{aligned} \quad (3.32)$$

Similarly, for $\underline{\mathbf{Y}}$:

$$\begin{aligned} \underline{\mathbf{X}} &= \mathbf{A}_y \times_2^1 \underline{\mathbf{G}}_y \times_3^1 \mathbf{B}_y^T \\ &= \underline{\mathbf{G}}_y \times_1 \mathbf{A}_y \times_2 \mathbf{I} \times_3 \mathbf{B}_y. \end{aligned} \quad (3.33)$$

where $\mathbf{I}$ is the identity matrix and $\underline{\mathbf{G}}_n \in \mathbb{R}^{2 \times 2 \times 2}$, $\mathbf{A}_n, \mathbf{B}_n, \mathbf{I} \in \mathbb{R}^{2 \times 2}$ for $n \in [x, y]$. Observe, that derived expressions are equivalent to (3.20), therefore, Proposition 3 is proven. $\square$

3.2.3 General properties

Provided that TN summands are compliant with the following assumptions listed in Section 3.2.2, they inherit associativity, distributivity and commutativity properties from its two dimensional counterpart of liner algebra. Furthermore, the identity and inverse operations hold true as well. General properties of TNs are summarised in Table 3.2, for tensor factorisation of a multi-dimensional array, $\underline{\mathbf{O}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, where all elements equal to zero is represented as tensor network, $\underline{\mathbf{O}}$.

Table 3.2: Algebraic properties of TN summation

Associativity	$(\underline{X} + \underline{Y}) + \underline{Z} = \underline{X} + (\underline{Y} + \underline{Z})$ $(a_1 a_2) \underline{X} = a_1 (a_2 \underline{X})$
Distributivity	$a_1 (\underline{X} + \underline{Y}) = a_1 \underline{X} + a_1 \underline{Y}$ $(a_1 + a_2) \underline{X} = a_1 \underline{X} + a_2 \underline{X}$
Commutativity	$\underline{X} + \underline{Y} = \underline{Y} + \underline{X}$
Inverse	$\underline{X} + (-\underline{X}) = (-\underline{X}) + \underline{X} = \underline{O}$ $\underline{X} - \underline{Y} = \underline{X} + (-\underline{Y})$
Identity	$\underline{X} \pm \underline{O} = \underline{X}$

3.2.4 Feature fusion via TN summation

General overview of classification setup

The TN summation operator is next validated through a practical example of feature fusion in image classification for the benchmark ETH-80 dataset [80]. It consists of 3280 color images (128×128 pixels), from 8 categories: apple, car, cow, cup, dog, horse, pear, tomato. Each category contains 10 different objects with 41 views per object, spaced equally over the viewing hemisphere. Observe that the samples of ETH-80 dataset are inherently multi-dimensional since any colour (RGB) image can be represented as a three-dimensional array $pixel_X \times pixel_Y \times 3$. For the following simulations, the images were rescaled to 32×32 pixels to form 3-rd order tensors $\underline{\mathbf{X}}_m \in \mathbb{R}^{32 \times 32 \times 3}$. The so-extracted features were then used as input to the classification models:

- Support Vector Machine which employed an RBF kernel and a “one-vs-one” approach for multi-class classification;
- Euclidean distance based K-Nearest-Neighbours with the number of neighbours set to three;
- Feed-forward multi-layer Perceptron neural network with two hidden layers and a ReLU activation function;
- Random Forest (RF) with 10 trees, the nodes of which were expanded until all leaves were pure.

All multi-linear operations for feature extraction were performed with the use of our own open source software for tensor analysis HOTTBOX[81], whereby the implementation of the classifiers was adopted from the highly optimised library of classical machine learning algorithms Scikit-Learn[82]. The classification rates were computed as an average of 100 independent realisations.

Feature extraction and classification

Consider a training set that consists of $\underline{\mathbf{X}}^m \in \mathbb{R}^{32 \times 32 \times 3}$ images from different categories. First, every sample was represented in the Tucker form (2.55) either through HOSVD or HOOI in order to ensure uniqueness:

$$\underline{\mathbf{X}}^m \approx \underline{\mathbf{G}}^m \times_1 \mathbf{A}^m \times_2 \mathbf{B}^m \times_3 \mathbf{C}^m \quad (3.34)$$

3. Feature Extraction and Data Fusion

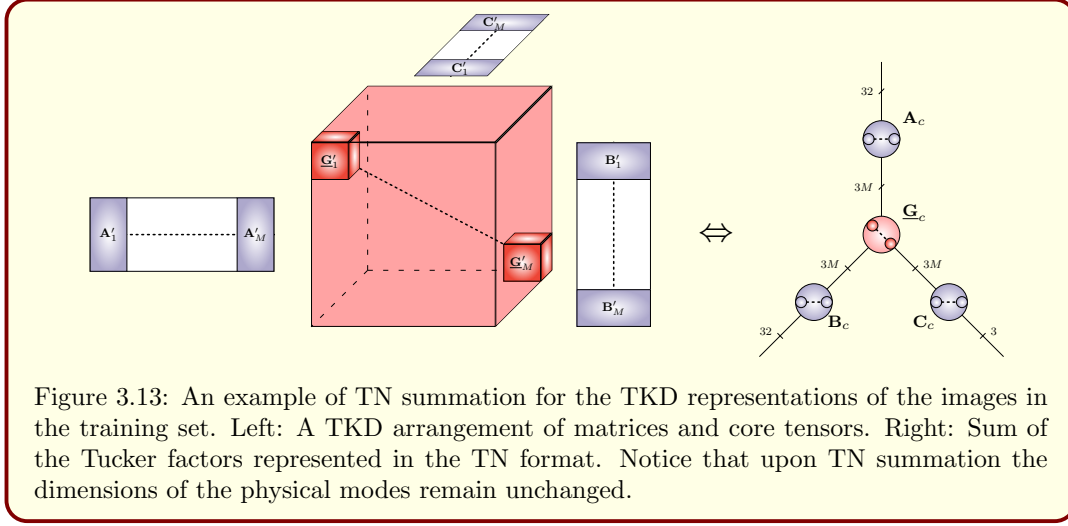


Figure 3.13: An example of TN summation for the TKD representations of the images in the training set. Left: A TKD arrangement of matrices and core tensors. Right: Sum of the Tucker factors represented in the TN format. Notice that upon TN summation the dimensions of the physical modes remain unchanged.

where the core tensor $\mathbf{G}^m \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ expresses relation between multi-linear features $\mathbf{A}^m$, $\mathbf{B}^m$ and $\mathbf{C}^m$. Two main factors impact choice of multi-linear rank, $\text{mlrank}(R_1, R_2, R_3)$, across every data sample: (i) a core tensor, $\mathbf{G}$, to be kept at relatively small size; (ii) while the TKD form provides sufficiently good approximation

$$\left\| \underline{\mathbf{X}}^m - \llbracket \mathbf{G}^m; \mathbf{A}^m, \mathbf{B}^m, \mathbf{C}^m \rrbracket \right\|_F < \delta. \quad (3.35)$$

For the rescaled images from the ETH-80 dataset, the values of multi-linear rank were found empirically and set to $R_1 = R_2 = R_3 = 3$. To emphasise the features of each image without affecting the TKD approximation, each core tensor $\mathbf{G}^m$ was normalised to unit norm and the factor matrices $\{\mathbf{A}^m, \mathbf{B}^m, \mathbf{C}^m\}$ were scaled by $\eta_m^{1/3}$, where $\eta_m = \|\mathbf{G}^m\|_F$. Through the TN summation approach, the scaled factor matrices $\{\mathbf{A}^m, \mathbf{B}^m, \mathbf{C}^m\}$ were concatenated into the matrices $\{\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c\}$ and the corresponding core tensors into $\mathbf{G}_c$, as illustrated on Fig. 3.13. Because the goal was to examine the mixture of the actual features, focus was placed on the factor matrices $\{\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c\}$, onto which an SVD was subsequently applied. For dimensionality reduction, the first $\{R_1, R_2, R_3\}$ singular vectors were retained, to yield matrices $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}\}$. Next, every image from the training set had to be projected onto the feature space inherent to all data samples which is spanned by $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}\}$:

$$\hat{\underline{\mathbf{X}}}^m = \underline{\mathbf{X}}^m \times_1 \hat{\mathbf{A}}^T \times_2 \hat{\mathbf{B}}^T \times_3 \hat{\mathbf{C}}^T \quad (3.36)$$

Finally, we vectorised the so-obtained features, $\hat{\mathbf{x}}^m = \text{vec}(\hat{\underline{\mathbf{X}}}^m)$, to make them compatible with the standard machine learning classifiers. Therefore, these could be fed into, for example, SVM or NN during the training stage. Furthermore, each new element from a test set, $\underline{\mathbf{X}}^n$, needed to be represented within a common feature space through (3.36) using pre-computed basis matrices $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}\}$ as $\hat{\underline{\mathbf{X}}}^n$, and consequently vectorised into $\hat{\mathbf{x}}^n = \text{vec}(\hat{\underline{\mathbf{X}}}^n)$ prior to making a prediction. This procedure for feature extraction is outlined in Algorithm 6.

The effectiveness of the proposed data fusion framework via TN summation was evaluated against the ETH-80 benchmark dataset, whereby the selected classifiers were trained either on features extracted in accordance with the Algorithm 6 or using the original samples. Moreover, for a fair comparison, we also employed an intrinsically multi-dimensional method to features acquisition, namely “tensor stacking” [83]. In contrast to the TN summation approach, it attempts to perform a simultaneous factorisation of all training samples which is achieved through stacking samples along a new dimension, $\text{stack}_4(\underline{\mathbf{X}}^m) \rightarrow \underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K \times M}$, followed by the estimation of common feature space via the HOSVD:

Algorithm 6: Sum of TNs for feature extraction

Input : Dataset $\mathcal{X} = \{\underline{\mathbf{X}}^n \in \mathbb{R}^{I \times J \times K}; \quad n = 1, \dots, N\}$
Multilinear rank (R_1, R_2, R_3) .

Output: Extracted features $\mathring{\mathcal{X}} = \{\mathring{\mathbf{x}}^n \in \mathbb{R}^L; \quad n = 1, \dots, N; \quad L = R_1 R_2 R_3\}$
suitable for classical ML algorithms.

- 1 Select portion of $\mathcal{X}$ that will form a training set $\mathcal{X}^m$
- 2 Initialise $\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c$ as empty matrices with I, J and K rows respectively
- 3 **foreach** element m from a training set $\mathcal{X}^m$ **do**
- 4 $\underline{\mathbf{X}}^m \rightarrow \underline{\mathbf{G}}^m \times_1 \times \mathbf{A}^m \times_2 \mathbf{B}^m \times_3 \mathbf{C}^m$; // HOOI or HOSVD algorithm
- 5 $\eta_m \leftarrow \|\underline{\mathbf{G}}^m\|_F$
- 6 $\mathbf{A}_c \leftarrow \begin{bmatrix} \mathbf{A}_c & \eta_m^{1/3} \mathbf{A}^m \end{bmatrix}$
- 7 $\mathbf{B}_c \leftarrow \begin{bmatrix} \mathbf{B}_c & \eta_m^{1/3} \mathbf{B}^m \end{bmatrix}$
- 8 $\mathbf{C}_c \leftarrow \begin{bmatrix} \mathbf{C}_c & \eta_m^{1/3} \mathbf{C}^m \end{bmatrix}$
- 9 **end foreach**
- 10 $\mathring{\mathbf{A}} \leftarrow \text{tSVD}(\mathbf{A}_c, R_1)$
- 11 $\mathring{\mathbf{B}} \leftarrow \text{tSVD}(\mathbf{B}_c, R_2)$
- 12 $\mathring{\mathbf{C}} \leftarrow \text{tSVD}(\mathbf{C}_c, R_3)$
- 13 **foreach** element n in dataset $\mathcal{X}$ **do**
- 14 $\mathring{\underline{\mathbf{X}}}^n \leftarrow \underline{\mathbf{X}}^n \times_1 \mathring{\mathbf{A}}^T \times_2 \mathring{\mathbf{B}}^T \times_3 \mathring{\mathbf{C}}^T$
- 15 $\mathring{\mathbf{x}}^n \leftarrow \text{vec}(\mathring{\underline{\mathbf{X}}}^n)$
- 16 **end foreach**
- 17 **return** $\mathring{\mathcal{X}} = \{\mathring{\mathbf{x}}^n; \quad n = 1, \dots, N\}$

$$\begin{aligned}
\underline{\mathbf{X}} &\approx \underline{\mathbf{G}} \times_1 \mathring{\mathbf{A}} \times_2 \mathring{\mathbf{B}} \times_3 \mathring{\mathbf{C}} \times_4 \mathbf{D} \\
\mathring{\underline{\mathbf{X}}} &= \underline{\mathbf{G}} \times_4 \mathbf{D} \\
\mathring{\mathbf{x}}^m &= \mathring{\underline{\mathbf{X}}}_{(4)}[m, :]
\end{aligned} \tag{3.37}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ denote the feature subspaces and $\mathring{\mathbf{x}}^m$ represents features to be fed into the classifiers at the training stage.

The classification rates for the SVM, kNN, RF and MLP were computed as an average of 100 realisations. The results based on 75% of the available images to serve as training data and are summarised in Table 3.3, whereas Fig. 3.14 depicts the accuracy as a function of the portion of dataset used for training the SVM algorithm. Observe an improvement of about 2% over tensor stacking, and almost 5% with respect to a direct usage on the original data.

3.3 Chapter Summary

In this chapter, we have presented a novel framework for common and individual feature extraction based on the CPD and LL1 decomposition with the non-negativity constraint (Sec-

3. Feature Extraction and Data Fusion

Table 3.3: Classification rates for the ETH-80 dataset				
	SVM	kNN	MLP	RF
Original images	89.84	88.31	85.43	83.33
Tensor stacking	92.94	88.55	88.28	89.75
Sum of TNs	94.81	90.51	89.23	86.20

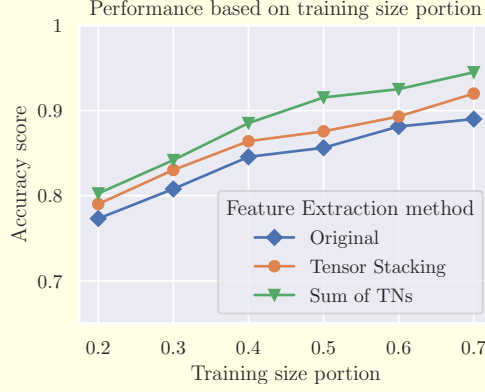


Figure 3.14: Classification accuracy of the SVM algorithm illustrated as a function of the proportion of training size. Three different training sets were formed, each of which contained either original samples, features extracted through tensor stacking and TN summation frameworks.

tion 3.1.4). The choice of these tensor decompositions for the proposed approach is justified by their intrinsic characteristic which allows us to establish a one-to-one connection between the components to facilitate intuitive interpretation. The multi-modal relations expressed through the outer product play a key role in the elimination of the shared information from the multi-block data. Unlike the matrix techniques, the proposed method is very flexible and is not restricted to input data of a specific shared structure. In this way, the performance of machine learning algorithms can be enhanced, as the classification models use only the much lower dimensional and significantly more discriminative individual information during the training stage. Simulations have employed the ORL database of images taken from various angles, under several illumination conditions, and with different facial expressions of the subjects, and have achieved excellent results, as shown in Section 3.1.5, despite the small size of the dataset and a high number of distinct classes.

Next, our focus shifted towards the parallel problem associated with an effective approach to feature fusion. Drawing inspiration from the fundamental methods and properties of linear algebra, we have introduced a mathematical formalism behind the summation of tensor networks in Section 3.2.2 with general properties of the equivalent matrix operations generalised to the case of multi-dimensional arrays of any arbitrarily high order. This has naturally led to a physically meaningful interpretation of the summation operation as a means for information blending that is possible only when tensor network representations of different information are considered as unified entities. Practical advantages have been demonstrated in Section 3.2.4 through an image classification application based on the ETH-80 dataset, whereby every sample is represented in accordance with the multi-linear decompositions, and extracted features are combined via the tensor summation framework. This makes it possible to extract relevant shared information in the original data, which is shown to yield significant advantages throughout the processing chain.

Chapter 4

Tensor Decompositions and Wisdom of the Crowd

Hiding within those mounds of data is the knowledge that could change the life of a patient, or change the world.

Atul Butte

Chapter Overview

Big Data applications are prohibitive to classical ensemble learning on the exceedingly large scale, raw input data and dimensionality reduction techniques are a prerequisite. To this end, this chapter considers a novel framework that generalises classic flat-view ensemble learning to multi-dimensional tensor-valued data. This is achieved by virtue of tensor decompositions, whereby the proposed method, referred to as tensor ensemble learning (TEL), decomposes every input data sample into multiple factors which allows for a flexibility in the choice of multiple learning algorithms in order to improve test performance. The TEL framework is shown to naturally compress multi-dimensional data in order to take advantage of the inherent multi-way data structure and exploit the benefit of ensemble learning. The proposed framework is first verified through the application of Higher Order Singular Value Decomposition for the classification of images from the ETH-80 dataset. This leads to a considerable improvement compared to the classical ensemble learning approach of bootstrap aggregating. Next, as a case study, we demonstrate the feasibility of the proposed multi-linear learning analysis for cardiovascular diagnosis. In particular, a blood flow of 22 participants is inspected through a combination of the Canonical Polyadic Decomposition and TEL approaches, and the outcomes are compared to results obtained by a qualified vascular scientist during a manual examination of B-mode ultrasound recording.

4.1 Introduction

The phenomenon of the *wisdom of the crowd* has been known for a very long time and was originally formulated by Aristotle. It simply states that the collective answer of a group of people to questions related to common world knowledge, spatial reasoning, and general estimation tasks, is often superior to the judgement of a particular person within this group.

With the advent of computer, the machine learning community have adopted this concept under the framework of *ensemble learning* [84]. This class of methods can be described as a collection of base learners whereby each learner generates a particular hypothesis about an

underlying process that governs the input data. This makes it possible to construct the final model as a strategic aggregation of the outputs from its constituent learners. Indeed, the “wisdom of the base learners” has been proven to be a powerful way to enhance the performance when solving both classification and regression types of problems [85, 86, 87]. For example, ensemble learning has been a key element in winning solutions to the Netflix Prize competition [88]. Generally speaking, ensemble methods can be considered within the three main groups:

- *Boosting*. The aim is to sequentially train a series of estimators whereby every subsequent estimator puts more emphasis on samples that were previously predicted incorrectly [89]. The most established variant of this strategy is the *AdaBoost* algorithm [90].
- *Stacking*. The gist of his strategy is that a set of base classifiers is first trained on the original input data, followed by “meta learning” in order to combine the outputs of base classifiers for enhanced performance [91].
- *Bagging*. This family relies on the creation of multiple “surrogate” training datasets from the original data, for example through resampling with replacement [92]. An independent base classifier is then applied to each new dataset and their outputs are combined to reach a consensus. To date, the most successful bagging approach is the Random Forest algorithm [93].

It is important to notice that common machine learning algorithms, such as Neural Networks (NN) or Support Vector Machines (SVM), operate on the basis of “one way” vector inputs, even if the original data is inherently multi-dimensional or multivariate, that is “multi-way”. Such “flat view” representation of multi-dimensional data arrays prevents learning models from taking full advantage of the underlying multi-way latent structure, as in the “matrix world” the cross-modal dependencies become obscure or even completely broken. This all calls for modern methodologies which can maintain the original structure in the data, an ideal avenue to explore multi-way analyses based on tensor decompositions [3]. Indeed, tensor based representations of multi-faceted data have been shown to be a promising tool for feature extraction [94, 95], data fusion [96, 97], anomaly detection [2] and classification [63, 66].

The research paradigm which lies at the intersection between ensemble learning and high dimensional data processing is known as *multi-view learning*. The concept of the multi-view ensemble learning was proposed in [98], however, despite its intrinsic multi-way structure, the data are still considered as single-way entities (vectors); this does not admit natural dimensionality reduction that would take advantage of the rich latent structure present in data. Furthermore, the analysis is limited to finding only relevant and irrelevant features. Other approaches under the framework of multiple-view multiple learners have been successfully applied to semi-supervised and active learning [99, 100], but still without employing multi-dimensional arrays (tensors) as a compact, rigorous and inherently structure-preserving model. The aim of this chapter is therefore to fill this void in the open literature and to introduce a novel framework which naturally incorporates physically meaningful tensor decompositions into ensemble learning of the intrinsically multi-way and multi-modal data.

4.2 Ensemble Learning: The Bagging Approach

Consider an ensemble of N independent classifiers, $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_N\}$, employed for a binary classification problem based on a dataset $\mathcal{D} : \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^M, y^M)\}$, where $y^m \in \{0, 1\}$ and $m = 1, \dots, M$. Provided that every classifier within the ensemble, $\mathcal{C}_n$, misclassifies the previously unseen sample, $\mathbf{x}^{new}$, with probability $p < 0.5$, we can write

$$\mathcal{P}(\mathcal{C}) = \sum_{n=\lceil \frac{N}{2} \rceil}^N \binom{N}{n} p^n (1-p)^{N-n} \quad (4.1)$$

$$\lim_{N \rightarrow \infty} \mathcal{P}(\mathcal{C}) = 0$$

4. Tensor Decompositions and Wisdom of the Crowd

where $\mathcal{P}(\cdot)$ is the probability of majority voting being incorrect. It has been proved that the operator $\mathcal{P}$ is monotonically decreasing with N [101], which simply implies that the more base learners participate in the majority vote the more accurate their collective decision. The two conditions that should be satisfied for this to hold true are:

1. Every predictor should perform better than a random guess;
2. The errors of individual predictors should be uncorrelated.

The latter condition has been the main challenge for all bagging classifiers. In principle, there are two ways to address this problem:

- Through a heterogeneous set of base learning algorithms in order to introduce a degree of diversity into the individual hypotheses, despite each being just a crude approximation of the true underlying processes that govern the data. In this way, the individual under-performance of a particular learning model is compensated for when all learning models are combined together into a unified global system;
- Utilising a homogeneous set of base classifiers and exposing each member to a subset of the training data. This data manipulation procedure is called *resampling*, and can significantly improve the overall generalisation ability of the model.

However, in real world scenarios, none of these methods is capable of completely removing correlation among the hypotheses generated by the base classifiers even if these are applied at the same time.

4.3 Mixture of Experts: Multidimensional Crowd

As described previously, current methods for generating ensembles of estimators rely on re-sampling the input data and/or using a set of different base classifiers for same data. Neither approach takes into account the original structure intrinsic to the data and the underlying multimodal dependencies. To this end, we propose a natural learning framework for multi-way data, referred to as *tensor ensemble learning* (TEL), which enables ensemble construction based on direct application of multi-linear analysis. In this way, the inherently rich structure of the input samples is fully exploited which equips the framework with the ability to extract otherwise indistinguishable latent information from multi-dimensional data. The proposed TEL method comprises the four main stages:

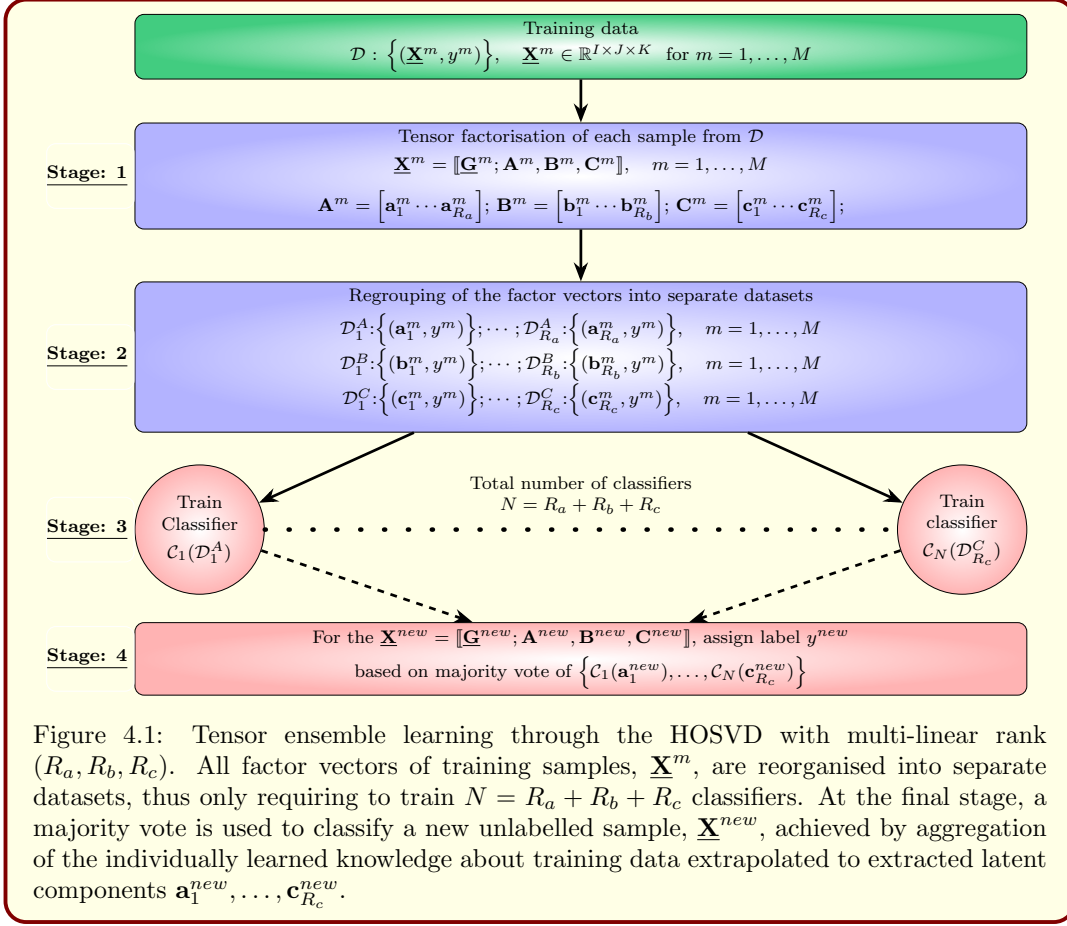
Stage 1. Apply a tensor decomposition to each multi-dimensional sample from the training set in order to find latent components. Every tensor decompositions considered in Section 2.3 can be utilised at this stage, in particular, Canonical Polyadic Decomposition (CPD) [84, 102], Higher Order Singular Value Decomposition (HOSVD) [24], Block Term Decomposition (BTD) [74] and the Tensor Train Decomposition (TT) [1, 9, 26].

Stage 2. Perform reorganisation of the components extracted in Stage 1 in order to generate multiple new datasets, each containing an incomplete information about the original sample. Given the different notions of existing tensor decompositions, there are multiple ways to accomplish this step.

Stage 3. Train an ensemble of base learners in order to generate a series of independent hypotheses about the underlying processes that govern the datasets constructed in Stage 2.

Stage 4. When facing a previously unseen data sample, aggregate the knowledge about its latent components based on the hypotheses from Stage 3. This can be achieved using any basic method, such as the weighted majority vote, or through more advanced machine learning algorithms.

4. Tensor Decompositions and Wisdom of the Crowd



Having established such a general framework of the tensor ensemble learning, we next demonstrate a successful implementation (depicted in Fig. 4.1) of the proposed concept in the context of supervised multi-class classification. For simplicity of presentation and ease of illustration we restrict ourselves to tensors of order 3, however, application of considered approach can be straightforwardly generalised to tensors of any arbitrary order.

4.4 Tensor Ensemble Learning for Classification

Consider the problem of training a classifier using a dataset, $\mathcal{D}$, of M labelled samples, which are naturally multi-dimensional, and come from K distinct categories, that is

$$\mathcal{D} : \{(\underline{\mathbf{X}}^m, y^m)\}, \quad m = 1, \dots, M \quad (4.2)$$

Here, $y^m \in [1, K]$ is a label associated with the training sample $\underline{\mathbf{X}}^m \in \mathbb{R}^{I \times J \times K}$. Recall that an important assumption behind the success of ensemble learning is that the hypotheses from each base learner are independent. This can be enforced by providing uncorrelated datasets to every base learner. Tensor decompositions that guarantee such uncorrelated factors have been presented in Section 2.3.2, namely, the HOSVD and the HOOI. Therefore, an implementation of Stage 2 of the TEL framework using HOSVD with the multi-linear rank (R_a, R_b, R_c) yields the following representation of the original data set

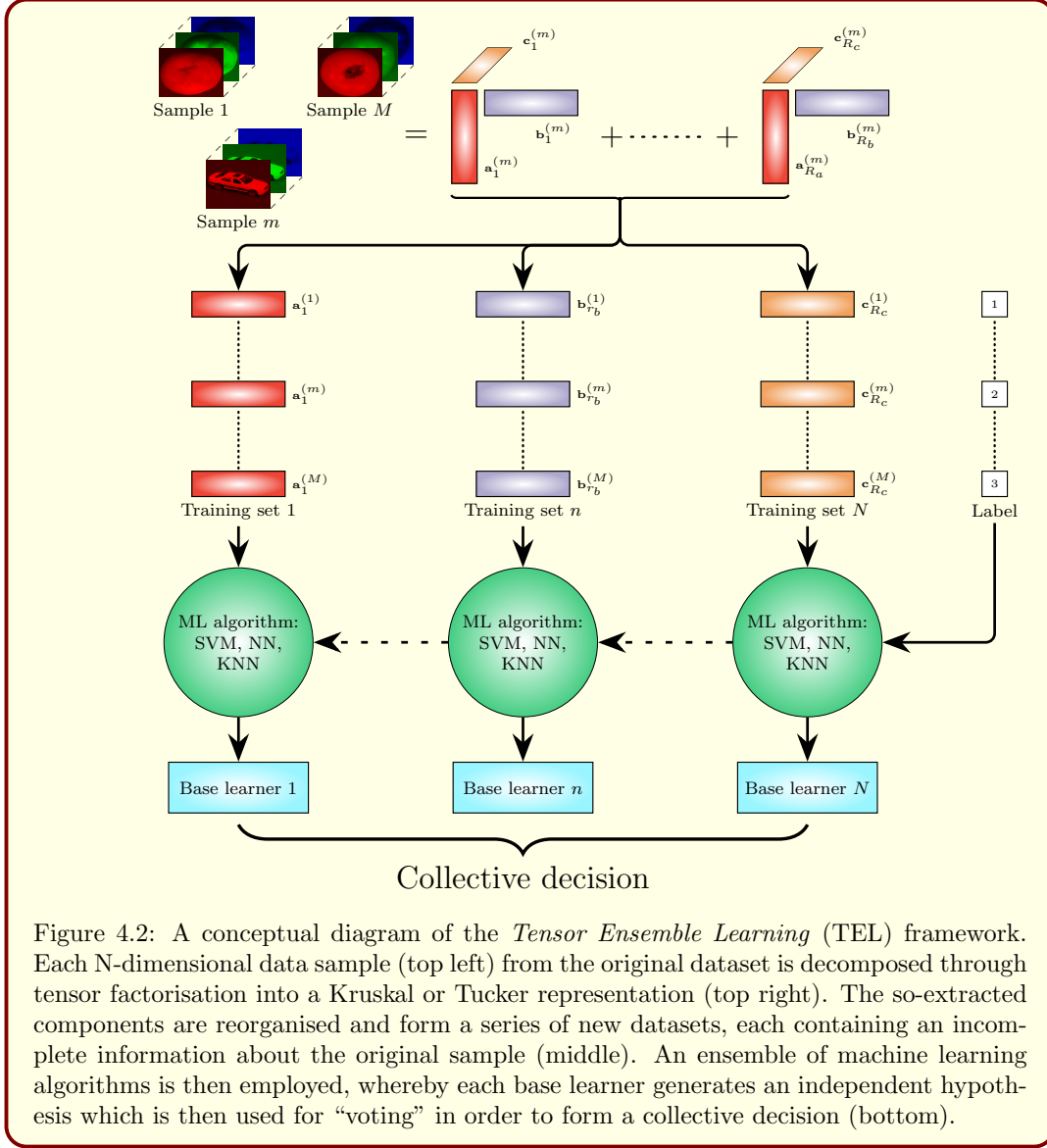


Figure 4.2: A conceptual diagram of the *Tensor Ensemble Learning* (TEL) framework. Each N-dimensional data sample (top left) from the original dataset is decomposed through tensor factorisation into a Kruskal or Tucker representation (top right). The so-extracted components are reorganised and form a series of new datasets, each containing an incomplete information about the original sample (middle). An ensemble of machine learning algorithms is then employed, whereby each base learner generates an independent hypothesis which is then used for “voting” in order to form a collective decision (bottom).

$$\mathcal{D}' : \left\{ \left(\llbracket \mathbf{G}^m; \mathbf{A}^m, \mathbf{B}^m, \mathbf{C}^m \rrbracket, y^m \right) \right\}, \quad m = 1, \dots, M \quad (4.3)$$

Here, the orthonormal factor matrices $\mathbf{A}^m \in \mathbb{R}^{I \times R_a}$, $\mathbf{B}^m \in \mathbb{R}^{J \times R_b}$, $\mathbf{C}^m \in \mathbb{R}^{K \times R_c}$ are uniquely defined by the corresponding column vectors $\mathbf{a}_{r_a}^m$, $\mathbf{b}_{r_b}^m$ and $\mathbf{c}_{r_c}^m$, where $r_a \in [1, R_a]$; $r_b \in [1, R_b]$ and $r_c \in [1, R_c]$. Thus, if a reorganisation is performed such that these factor vectors are considered by the base estimators independently, this would provide an ensemble learning framework which is practically feasible and yields enhanced performance after aggregation of the individual hypotheses. A detailed illustration of such reorganisation is given in Stage 2 of Fig. 4.1. Consequently, there are $N = R_a + R_b + R_c$ datasets which are separately fed into N base learners at the next stage. In this particular implementation of the TEL, the final stage can be accomplished using a simple majority vote but other aggregation methods could be equally utilised.

To summarise, the proposed Tensor Ensemble Learning – Vector Independent (TELVI) estimator is a machine learning model that belongs to the TEL framework, whereby the resampling

4. Tensor Decompositions and Wisdom of the Crowd

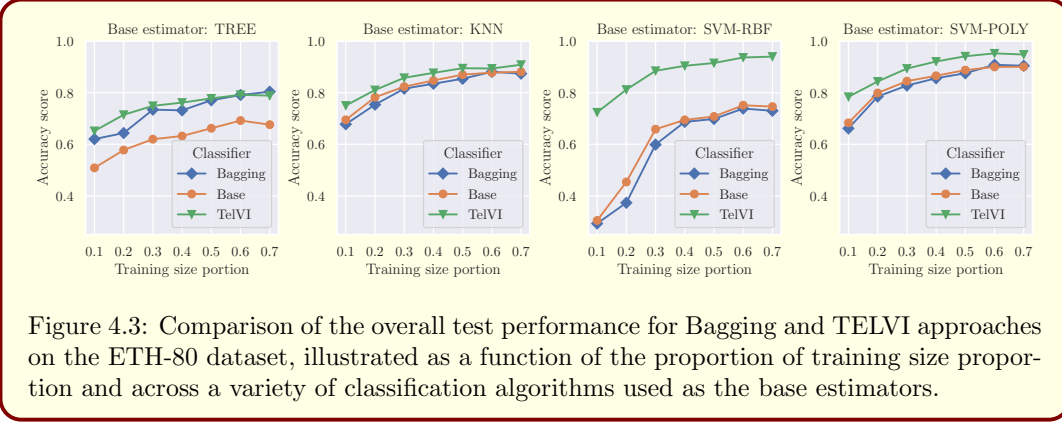


Figure 4.3: Comparison of the overall test performance for Bagging and TELVI approaches on the ETH-80 dataset, illustrated as a function of the proportion of training size proportion and across a variety of classification algorithms used as the base estimators.

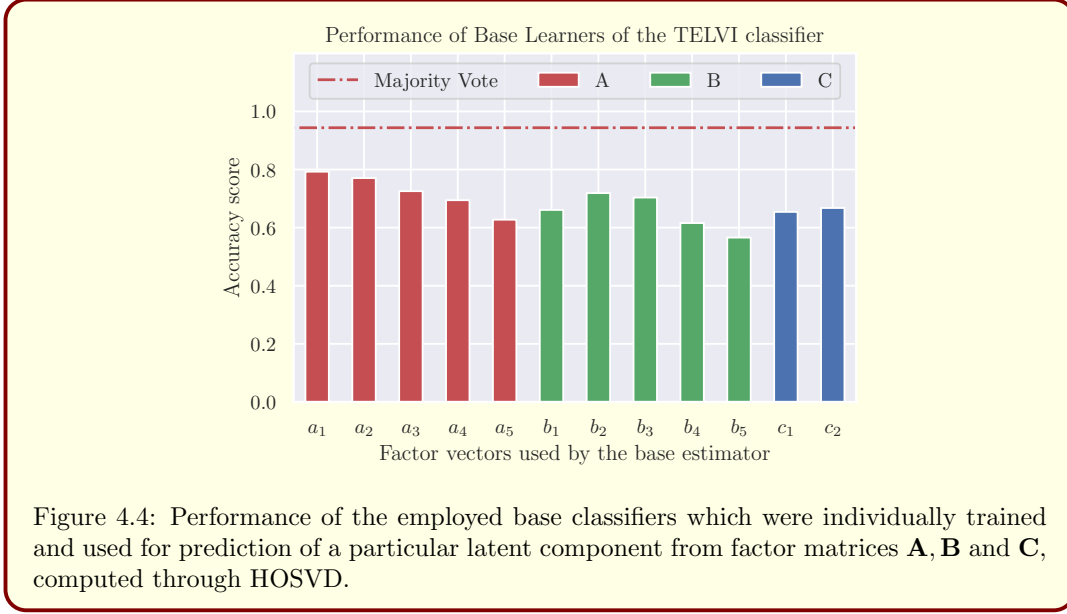
of the factor vectors is performed as depicted in Stage 2 of Fig. 4.1.

The proposed approach was employed for object classification from images within the benchmark ETH-80 dataset [80], while the whole procedure is illustrated on Fig. 4.2. The ETH-80 dataset consists of 3280 colour images (128×128 pixels) from 8 categories: apple, car, cow, cup, dog, horse, pear, tomato. Each category contains 10 different objects with 41 views per object, spaced equally over the viewing hemisphere. Observe that the samples of ETH-80 dataset are inherently multi-dimensional since any colour (RGB) image can be represented as a three-dimensional array $pixel_X \times pixel_Y \times 3$. For the simulations we used `Scikit-Learn` [103], a highly optimised library of classical machine learning algorithms, while the computation of the HOSVD and implementation of the TELVI classifier were implemented using our own recently developed Python software package for multi-dimensional data, `HOTTBOX`, publicly available at [104]. To assess performance of the proposed and classical approaches, as a performance metric we used the ratio of correctly categorised images to the total number of samples in the test set.

Prior to comparing the overall performances of the TELVI and Bagging approaches, classification accuracy scores of each chosen base classifier within the TELVI were computed in order to establish the behavioural correspondence between this implementation of TEL framework and classical ensemble learning. The classification pipeline for the TELVI classifier begins with splitting the images from the ETH-80 dataset into the training and test sets. This was followed by extracting the latent components from each image individually with the use of the HOSVD algorithm for the multi-linear rank (5, 5, 2), as it provided the best approximation error across all data samples. The latter should be treated as the hyper-parameter of the TELVI classifier. In our simulations, its values were chosen so as to give the optimal ratio between the compression rate and the approximation error of a particular tensor decomposition across all images. Next, the factor components of images from the training set were regrouped (as illustrated in Stage 2 of Fig. 4.1) into 12 new training datasets, thus requiring a total of 12 independent base learners to be employed.

After each base classifier had been trained, the TELVI classifier was ready to assign labels for the test samples that would undergo the same “preprocessing” steps (decomposing a test sample through HOSVD with multi-linear rank (5, 5, 2) and regrouping obtained components). Finally, a label was assigned based on the majority vote over a list of labels produced by the base classifiers.

Fig. 4.4 shows the individual test performance of each base classifier (Support Vector Machine with the polynomial kernel) trained with 50% of the original dataset. Observe that none of the base classifiers exhibited strong performance on the training set, with the highest reaching 80% in accuracy. However, when the individual classifiers were combined through a majority vote, the accuracy score of 94% was achieved. This is significantly higher than for any of the base estimators and provides conclusive evidence that the TELVI classifier operates in a manner



similar to classic ensemble learning approaches.

Next, the proposed method was rigorously assessed against the Bagging approach through four experiments. The images from the ETH-80 dataset were randomly split into the training and test data with the size of training proportion ranging from 10% to 70% of all available samples. In each experiment, all base learners of both TELVI and Bagging classifiers used the same machine learning algorithm: Support Vector Machines with both the Gaussian (SVM-RBF) and polynomial (SVM-POLY) kernel, Decision Trees (TREE) and K-nearest neighbours (KNN). Hyperparameters were tuned based on an exhaustive grid search with the 5-fold cross validation of the training data. The classification pipeline for the TELVI remained the same as described above. Recall that the Bagging approach requires to start with the vectorisation of all samples from the training and test data. Given the large number of features, dimensionality reduction through Principal Component Analysis (PCA) was used to reduce computational complexity during the training stage. For a fair comparison, the Bagging classifier also utilised 12 base learners that were combined by virtue of the majority vote.

The results of this set of experiments are presented in Fig. 4.3. Observe that TEL approach achieved significantly increased classification rates, independent of the nature of the particular base estimator at hand, with the highest scores reaching 95% for SVM-POLY, while Bagging performed at around the same level as the base classifier, except for the Decision Tree. This conclusively demonstrates that, even when resampling with replacement did not provide the desired increase in the performance, exploiting the inherent structure within the data is a promising way to create a series of datasets with enhanced discrimination ability, as a data manipulation part of ensemble learning.

4.5 Multi-linear Learning for Cardiovascular Diagnosis

Haemodialysis (HD) is an effective long-term renal replacement therapy that provides a lifeline for more than 30,000 people in the United Kingdom with end stage renal disease (ESRD) [105]. The Arteriovenous fistula (AVF) is a surgical anastomosis between an artery and a vein and is the preferred choice of vascular access for long-term HD [106, 107] as it is associated with longer access survival and lower complication rates as compared with catheters [108]. However, only 16% of patients initiate HD with AVFs due to a high primary failure rate [109]. The AVF failure

is an important cause of mortality, hospitalisation and excess healthcare cost [110, 111, 112]. The most important cause of AVF failure is progressive stenosis secondary to neointimal hyperplasia and fibrosis from consecutive cannulations [113]. The evaluation of AVF maturation and early detection of stenosis is important to allow for vascular interventions in order to maintain patency. Regular clinical examination and duplex ultrasound (DUS) assessments are non-invasive, inexpensive and are routinely used for AVF surveillance [106].

The DUS examination allows for the detection of areas with a significant velocity change by using a combination of B-mode (grey scale ultrasound), colour and spectral Doppler techniques [114, 115]. However, a number of studies have demonstrated that these measurements are poor at predicting AVF outcomes [116, 117, 118]. This may be related to the limitation of conventional Doppler techniques, which are angle dependent and therefore can only provide information when the flow is parallel to the beam direction, with inadequate estimation of the orthogonal components. Hence, DUS is significantly limited by tortuosity and complex multidirectional flow, which is inherent to AVF [119].

Over the recent years, various non-Doppler ultrasound approaches for flow estimation have been developed and are known as vector flow imaging (VFI) techniques [120]. Some VFI techniques are based on the tracking of speckle patterns on B-mode ultrasound caused by scattered ultrasound signals arising from moving blood cells. However, the interpretation of these complex patterns remains challenging and there is a need for the application of novel solutions to interpret the complex flow patterns associated with AVFs.

The B-mode ultrasound cine loops of AVF blood flow have immense volume, veracity, velocity and variety (the four V's of big data) and exhibit a large degree of structural richness. These data characteristics can restrict the application of classical analysis on the underlying features due to its "flat-view" nature. However, when re-arranged in multi-dimensional structures, the same data often admit much more convenient and mathematically tractable ways of analysis. Indeed, through multi-linear techniques and tensor decompositions data can be extrapolated providing highly informative multiway data representation. To this end, we will evaluate the feasibility of detecting AVF stenosis using a novel approach incorporating tensor decomposition and machine learning classification for the analysis of B-mode ultrasound cine loops.

For this experiment, gray-scaled video recordings of blood flow of 22 participants were collected whereby each subject underwent DUS surveillance of their AVF as per clinical protocol. All DUS examinations were performed with an Aixplorer ultrasound scanner, using a high frequency linear transducer (2-10 MHz). Patients were rested for 5 minutes in a seated position and then scanned with one arm leaning on the couch. All trials were performed with minimum pressure to avoid any deformation of the AVF. Consequently, all recordings were examined by a qualified vascular scientist with the use of B-mode ultrasound approach as well as colour and spectral Doppler technique at the arterial inflow, anastomosis, and venous outflow whereby the detection of stenosis was based on the assessment of blood flow volume, vessel diameter size and flow velocity. Thus, binary labels were assigned to every available data sample denoting either presence or absence of stenosis.

When applying multi-linear analysis, the first step was to reshape each video recording of intraluminal blood flow into an order-3 tensor whereby a set of consecutive frames were stacked together along the third, "frame", dimension as illustrated on Fig. 4.5. This allowed us to separately perform a multi-linear decomposition over every tensor. In particular, the CPD was chosen since it breaks down the initial tensor data into a linear combination of components, each of which is associated with an appropriate characteristic attributed to the B-mode ultrasound cine loop while carrying an inherent information about the recording as a whole. The so-obtained features were used as inputs to the TEL framework that adopts a multi-input neural network approach, consisting of:

- Three separate Long Short-Term Memory (LSTM) layers [121] with 8 cells that process each of the three extracted components act as the base learners (Stage 3 on Fig. 4.1);
- Majority vote (Stage 4 on Fig. 4.1), which is performed by a simple neural network of

4. Tensor Decompositions and Wisdom of the Crowd

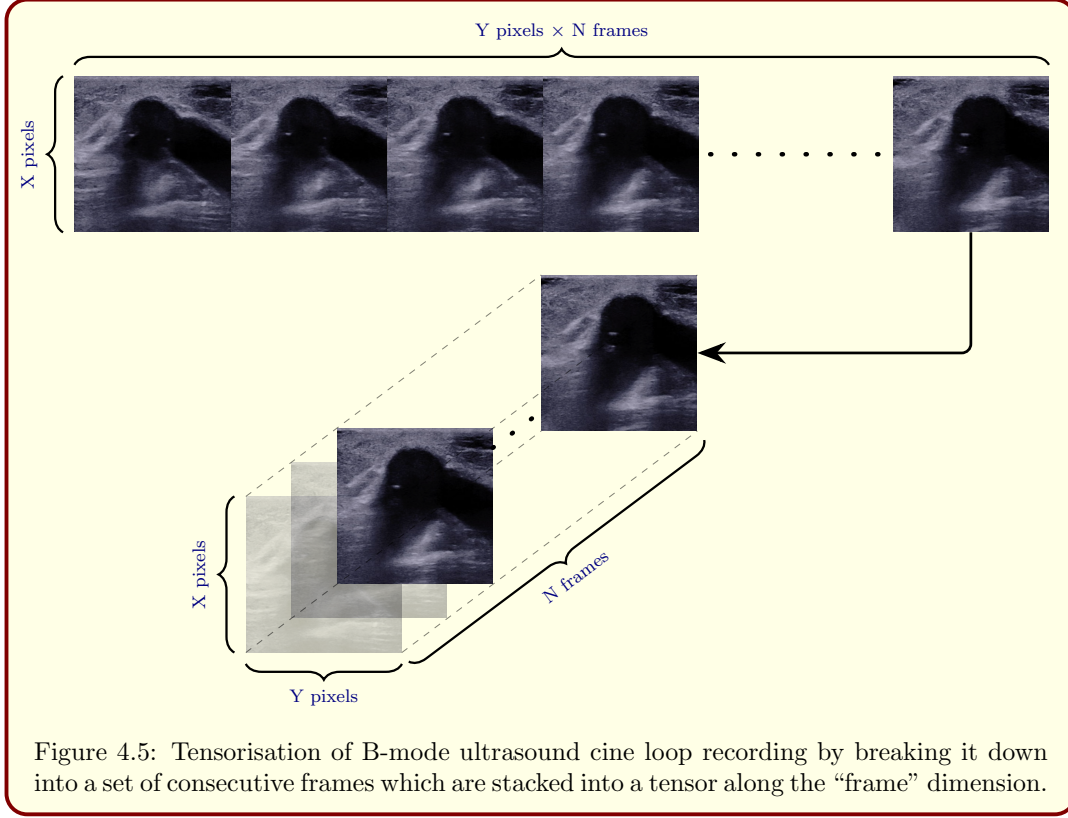


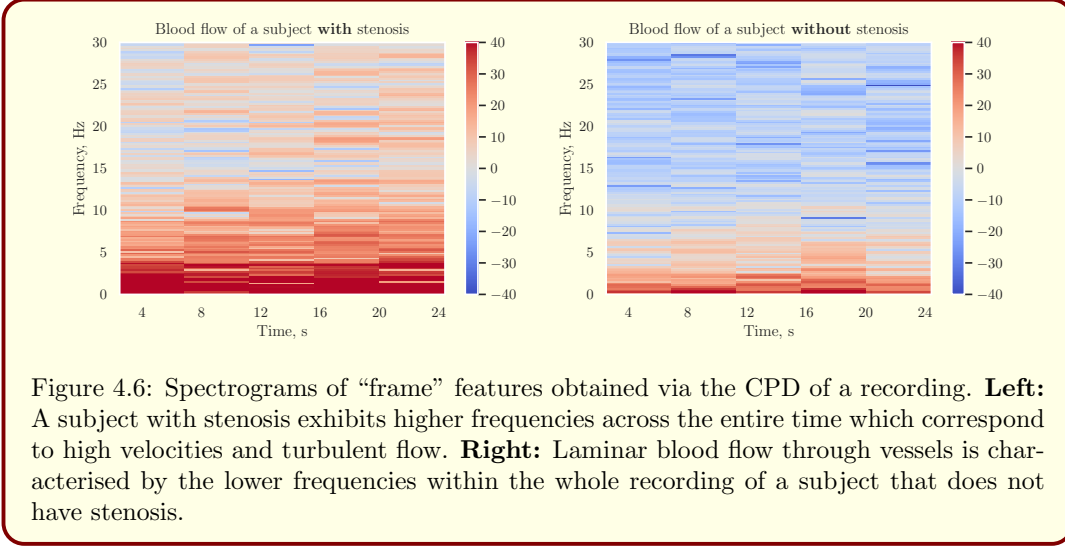
Figure 4.5: Tensorisation of B-mode ultrasound cine loop recording by breaking it down into a set of consecutive frames which are stacked into a tensor along the “frame” dimension.

single dense layer followed by an activation function which determines the probability of stenosis.

Prior to input to the neural network, data were Z-score normalized according to the metrics obtained from the training set. The component sequences were also zero-padded to be of uniform length per component, based on the length of the longest sequence of the component at hand in the training set. The network was trained using binary cross-entropy loss for a maximum of 100 epochs on mini-batches of size 1, with training being terminated early if the validation loss failed to improve for five consecutive epochs. The weights of the LSTM layers were regularised during training using both the L1 and L2 regularisation penalties with a value of 0.01.

To gain a better idea of how well the model generalises to new data and how much the performance varies based on different data samples, we generated multiple training sets by the bootstrap method. This was achieved through 1,000 iterations, with each iteration drawing “n” samples with replacements from the dataset to be used as a training set, where “n” is 80% of the total number of data points. We also ensured that samples related to all blood flows (arterial inflow, venous outflow, anastomosis) were present in every training set. The remaining out-of-bag samples were used as a test set to assess the performance of the algorithm at each iteration. The Area Under the Receiver Operating Characteristic (AUROC) curve of the model predictions on the out-of-bag samples was recorded for each iteration and the mean AUROC and corresponding confidence intervals (CI) for the mean were calculated. Thus, the full classification model achieved a median AUROC value of 0.69 (CI 0.68 to 0.70), thus showing promising predictive power despite the small size of the dataset.

Recall, that features extracted through tensor decompositions carry physical meaning. Therefore, components associated with the “frame” dimension of the original tensor data (see Fig. 4.5) can be interpreted as a time evolution of blood flow within the spatial region of the recording



throughout its duration. In other words, it is absolutely reasonable to apply the Short Time Fourier Transform in order to access feasibility of multi-linear analysis prior to feeding the obtained information into machine learning algorithms. Indeed, spectrograms of “frame” features of subjects with stenosis demonstrated a prominent presence of high frequencies (left side of Fig. 4.6) which correlates with the high velocities and turbulent blood flow as opposed to subjects without stenosis whose spectrograms only have low frequencies (right side of Fig. 4.6), effectively corresponding to normal heart beat and laminar blood flow through arteries and vessels.

Remark 16. *It is important to highlight that all neural network models were trained without using a graphical processing unit (GPU)-acceleration despite a huge amount of the original data. This is attributed to the effective representation of it in a lower-dimensional space by virtue of the tensor decompositions.*

4.6 Chapter Summary

In this chapter, we have introduced tensor ensemble learning (TEL), a novel framework for generating ensembles of base estimators for multi-dimensional data. This has been achieved by virtue of tensor decompositions, thus making TEL highly parallelisable and suitable for large-scale problems. As one of the possible implementations of TEL, we have presented the TELVI algorithm that has been derived in accordance with the definition of classical ensemble learning. The conducted simulations have shown that when compared with the popular approach of *bootstrap aggregating*, TELVI is capable of improving test performance of a multi-class image classification task. This success is partially due to its ability to obtain uncorrelated surrogate datasets that are generated by HOSVD and partially owing to the dimensionality reduction inherent to tensor decompositions which takes into account the original structure of the data. Next, we have incorporated the TEL concept into a cardiovascular diagnosis scenario and the problem of automatic identification of abnormal blood flow. We have demonstrated that the approach of training classification models on features extracted through tensor decomposition of B-mode ultrasound videos of blood flow, shows great promise at predicting AVF stenosis despite the small number of data-points. These results are particularly encouraging in the clinical scenario of AVF surveillance, which remains a challenging area where the novel solutions are necessary to improve clinical outcomes.

Chapter 5

Computational Efficiency and Practical Implementations

It's so cheap to store all data. It's cheaper to keep it than to delete it. You can have a ridiculously enormous and complex data set, but if you have the right tools and methodology, then it's not a problem.

Andrew McAfee and Aaron Koblin

Chapter Overview

Tensors have been gaining interest in the past decade as a powerful data analysis tool, owing to their characteristic of inherently preserving intra-data dependencies and correlations which carry additional valuable information. Hence, with the rapid emergence of big data, tensor decompositions have become an indispensable and natural tool to analyse datasets of high dimensionality and variety. Despite the obvious advantages, multi-linear algebra is not as intuitive as classical linear algebra, its counterpart for matrices and vectors. For example, tensor expressions are not so easily manipulated, and notation can be overwhelming. Moreover, tensor decompositions operate in a block fashion, yet the existing demand for Internet of Things, mobile and remote monitoring applications require computationally efficient and low-latency real-time implementations.

In this chapter, we take a deep dive into graphical representation of multi-linear operations to explore how this approach can be used to elucidate even on the most basic procedures. More specifically, we focus on inner products, tensor mode- n unfoldings in the context of the Tucker decomposition, and contraction products. By gaining deeper insight and better understanding of the field, we devise a new algorithm to efficiently perform contraction products, the Tensor Train Contraction Product. Next, we consider a novel approach for the computation of Tucker decomposition, based on the recursive updates of the factor matrices that take place only upon the arrival of a new tensor-variate data sample. The efficacy of the proposed algorithms opens new avenues for next generation of real-time streaming technology and applications. Finally, we introduce the Higher Order Tensors ToolBOX (HOTTBOX), an open source toolbox for tensor decompositions, statistical analysis, visualisation, feature extraction, regression and non-linear classification of multi-dimensional data. This toolbox was conceived with the aim to serve as a self-contained library which allows for seamless integration with other popular data science packages for data wrangling and visualisation, such as *Pandas* [122], *Matplotlib* [123], *Plotly*

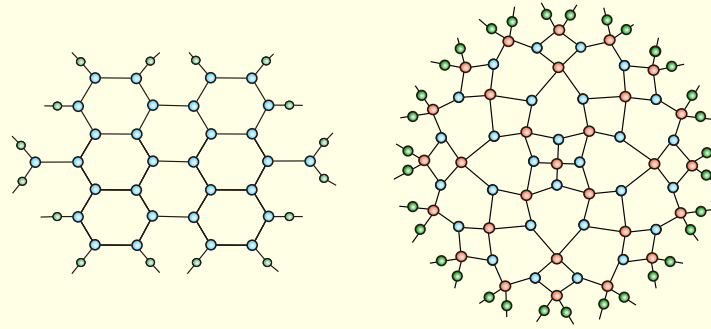


Figure 5.1: Example of two relatively involved tensor operations, but easily comprehensible through a graphical representation. Left: A TN representing a 16-th order tensor, with contraction nodes shown in blue, and physical nodes shown in green. Right: A TN representing a 32-th order tensor, with contraction nodes shown in blue (3-rd order) and orange (4-th order), and physical nodes in green. Image adapted from [1].

[124], *Dash* [125].

5.1 Tensor Contraction Product via Tensor Train Decomposition

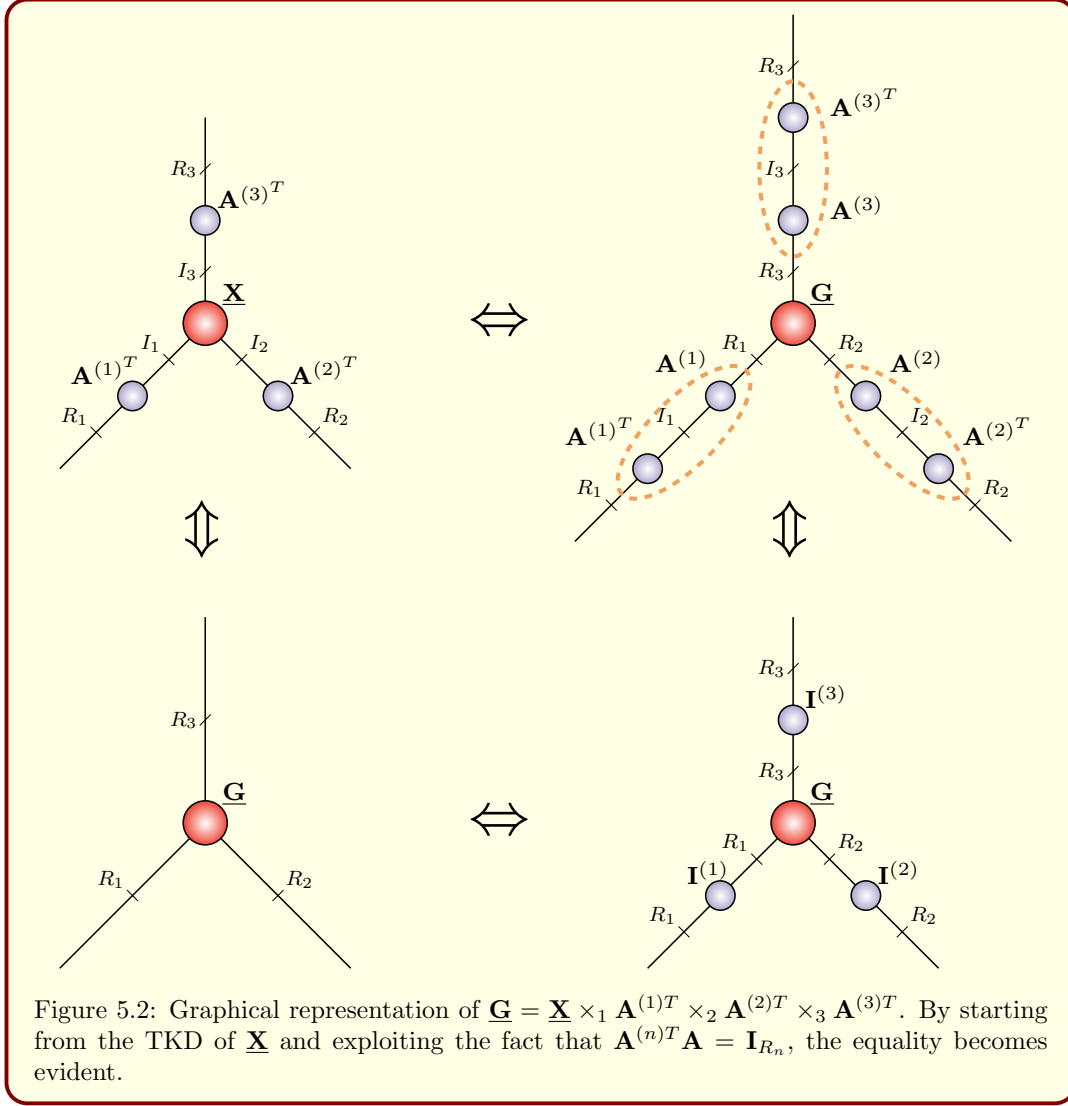
5.1.1 Introduction

Tensor operations expressed in a general form are often cumbersome even with the standard conventions and notation like those presented in Table 2.1 and Table 2.2. Even relatively concise multilinear operations, such as (2.57) for the Tucker (TKD) and (2.64) for the Tensor Train (TT) decompositions, may be difficult to comprehend and gain insight into. Consequently, tensor decompositions may not possess sufficient amount of intuition for multi-dimensional arrays of high order.

Taking inspiration from quantum physics and quantum chemistry [1], approaches have been developed to illustrate mathematical operations with the use of diagrams. Indeed, tensors, and hence TNs, can be represented as a connection of nodes and edges, in a way that resembles mathematical graphs. Any tensor of any order, including scalars (0-th order), vectors (1-st order), matrices (2-nd order) and higher order tensors, can be represented by circles, or nodes. Observe Fig. 2.3 where the edges represent the tensor modes, and, conventionally, they are labelled from the “top” of a node (north), then follow in an anti-clockwise direction; the labels designate the dimensionality of the mode. The number of edges connected to the node determines the order of the tensor. Hence, *contraction mode* is an edge that connects two tensors of equal size, while an edge which is linked from only one side represents a *physical mode* of the underlying tensors, or TN.

The basics of TNs operations have already been summarised on Fig. 2.4. This, in turn, allows for intuitive ways to correctly portray even the most complex operations. For instance, Fig. 5.1 illustrates two interactions among tensors which would otherwise translate in an extremely exhausting and hard to follow mathematical expression, as numerous matrices, together with 3-rd order, and 4-th order tensors are present.

Moreover, graphical representations may be exploited to deepen the understanding of the fundamental concepts, thanks to a solid grasp and understanding of the basics of such visualisation. This inevitably leads to novel approaches to efficient implementation of the tensor contraction product.



5.1.2 Common multi-linear operations through nodes and edges

Graphical representation of the Tucker Decomposition

In the following we set out to demonstrate common TKD expressions via the power of tensor graphical representations. Consider, for example, the TKD of a 3-rd order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. If the orthogonality constraints are imposed [24], then the following holds:

$$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \iff \underline{\mathbf{G}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \mathbf{A}^{(3)T} \quad (5.1)$$

where $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. The above derives from the fact that $\mathbf{A}^{(n)T} \mathbf{A} = \mathbf{I}_{R_n}$, but it can be difficult to grasp an immediate full intuitive understanding of (5.1). However, if the problem is viewed through a lens of diagrammatical approach, as shown in Fig. 5.2, it becomes immediately clear that the right-hand side of (5.1) reduces to $\underline{\mathbf{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ being multiplied by an identity matrix $\mathbf{I}_{R_n}$ along each mode.

Graphical representations may be employed as a tool to elucidate even more complicated mathematical problems. For instance, consider the optimisation problem for the Higher-Order

5. Computational Efficiency and Practical Implementations

Orthogonal Iteration (HOOI), shown in (2.61). This is equivalent to minimising the square of the objective function, given by

$$\begin{aligned}
\left\| \underline{\mathbf{X}} - \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket \right\|^2 &= \|\underline{\mathbf{X}}\|^2 - 2\langle \underline{\mathbf{X}}, \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket \rangle + \|\llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket\|^2 \\
&= \|\underline{\mathbf{X}}\|^2 - 2\langle \llbracket \underline{\mathbf{X}}; \mathbf{A}^{(1)T}, \dots, \mathbf{A}^{(N)T} \rrbracket, \underline{\mathbf{G}} \rangle + \|\underline{\mathbf{G}}\|^2 \\
&= \|\underline{\mathbf{X}}\|^2 - 2\langle \underline{\mathbf{G}}, \underline{\mathbf{G}} \rangle + \|\underline{\mathbf{G}}\|^2 \\
&= \|\underline{\mathbf{X}}\|^2 - \|\underline{\mathbf{G}}\|^2 \\
&= \|\underline{\mathbf{X}}\|^2 - \|\underline{\mathbf{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \dots \times_N \mathbf{A}^{(N)T}\|^2
\end{aligned} \tag{5.2}$$

The above can be reformulated as a series of subproblems and solved via an ALS approach [4]. Next, we shall concentrate on the mathematical steps in (5.2), and how the employed inner products can be expressed with the use of graphical approach to the representation of the operations involved. First, consider the equivalence

$$\langle \underline{\mathbf{X}}, \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket \rangle = \langle \llbracket \underline{\mathbf{X}}; \mathbf{A}^{(1)T}, \dots, \mathbf{A}^{(N)T} \rrbracket, \underline{\mathbf{G}} \rangle \tag{5.3}$$

From Fig. 5.3, it becomes obvious why this holds, even though a 3-rd order tensor, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, is considered for simplicity. An inner product between any two tensors equals the contraction along all their modes [4], as represented in the top-right corner of Fig. 5.3. It is possible to manipulate these graphical representations: instead of considering $\underline{\mathbf{G}}$ and factor matrices $\mathbf{A}^{(n)}$ together, we consider $\underline{\mathbf{X}}$ and $\mathbf{A}^{(n)T}$ (lower portion of Fig. 5.3). Notice that in the case of $\underline{\mathbf{X}}$, the contracting modes are I_n , while for $\underline{\mathbf{G}}$ contractions are carried out on modes R_n . As a result, the lower-left section of Fig. 5.3 corresponds to (5.3). The same approach can be used to justify

$$\|\llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket\|^2 = \|\underline{\mathbf{G}}\|^2 \tag{5.4}$$

Using the property $\langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle = \|\underline{\mathbf{X}}\|^2$ [1], we can repeat the procedure depicted on Fig. 5.3 with

$$\langle \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket, \llbracket \underline{\mathbf{G}}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket \rangle \tag{5.5}$$

Upon combining the matrices, $\mathbf{A}^{(n)}$, and by contracting them along their common modes, I_n , results in N products $\mathbf{A}^{(n)T} \mathbf{A}^{(n)} = \mathbf{I}_{R_n}$. Therefore, this leads to

$$\langle \llbracket \underline{\mathbf{G}}; \mathbf{I}^{(1)}, \dots, \mathbf{I}^{(N)} \rrbracket, \underline{\mathbf{G}} \rangle = \langle \underline{\mathbf{G}}, \underline{\mathbf{G}} \rangle = \|\underline{\mathbf{G}}\|^2 \tag{5.6}$$

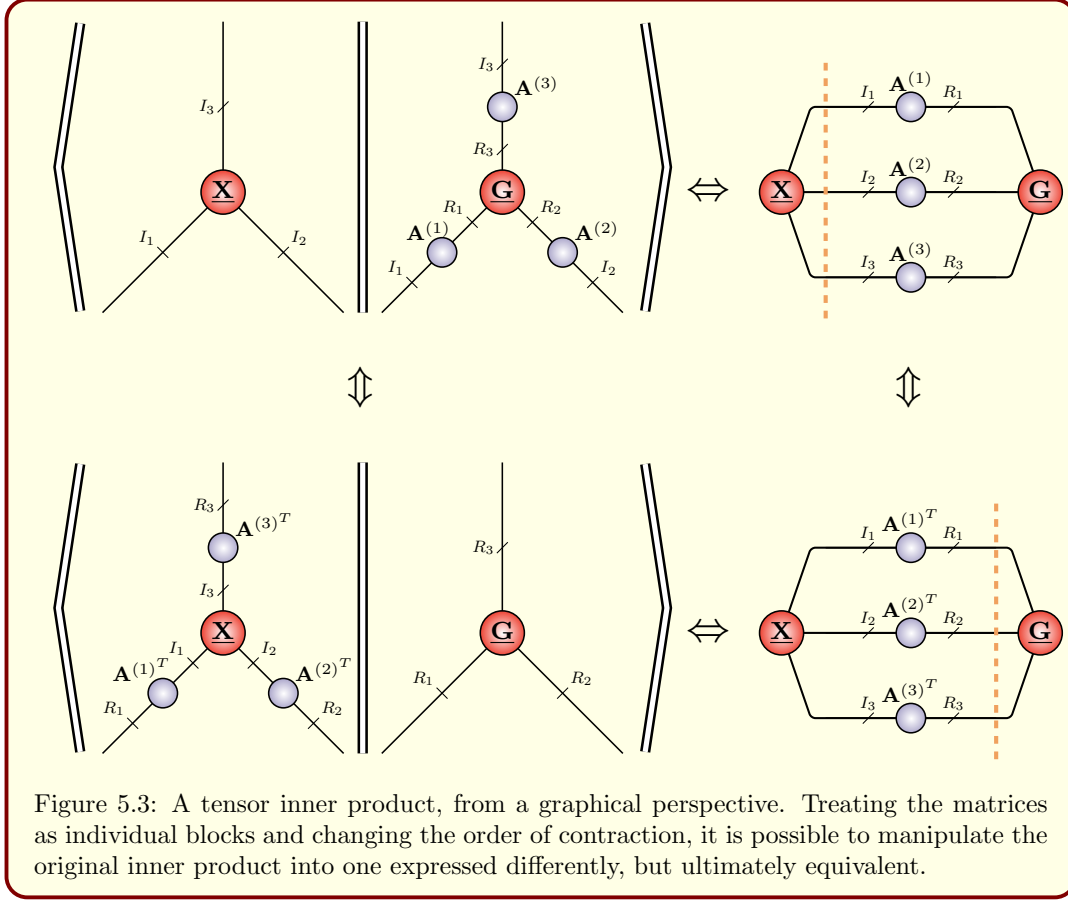
thus justifying (5.4).

Remark 17. For convenience, the above properties about inner products were presented in the light of the TKD, but they are valid for any tensor. This way clearly shows some of the benefits of graphical representations of TNs.

A Note on Tensor Unfoldings

Unfolding tensor (represented in the TKD format) into a matrix is a common technique. This allows us to comfortably operate within the realms of the well-known matrix algebra, while at the same time preserving the advantages and structure of multi-linear algebra, as shown in (2.56) and (2.58). Although not particularly complicated, these equations might be difficult to take for granted if no justification is provided. To this end, observe Fig. 5.4. The question is how do matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ take part in the operation. The answer is that the second and third dimension of $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, I_1 and I_2 , are multiplied, so a Kronecker product between $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ follows intuitively. This results in a matrix $\mathbf{K}_* \in \mathbb{R}^{I_2 I_3 \times R_2 R_3}$. To further maintain dimensionality compatibility and allow for applying of the contraction, it also becomes natural

5. Computational Efficiency and Practical Implementations



to unfold $\underline{\mathbf{G}}$ along its 1-st mode, yielding $\mathbf{G}_{(1)} \in \mathbb{R}^{R_1 \times R_2 R_3}$. By observing the resulting final graph, it is now clear that $\mathbf{X}_{(1)} = \mathbf{A}^{(1)} \mathbf{G}_{(1)} \mathbf{K}$ where $\mathbf{K} = \mathbf{K}_*^T = (\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T \in \mathbb{R}^{R_2 R_3 \times I_2 I_3}$.

Remark 18. Note that the same approach is valid to justify unfoldings in the light of the CPD, as it can be considered a special case of the TKD.

A Note on Topology Equivalence

If two TNs are represented through different schemas but have the same topology and elements, they are equal. In other words, the topology of a TN defines the TN itself. This result is intuitive and has a foundation in Graph Theory [126]. However it is possible to express the same graphical representation of a tensor with different equations.

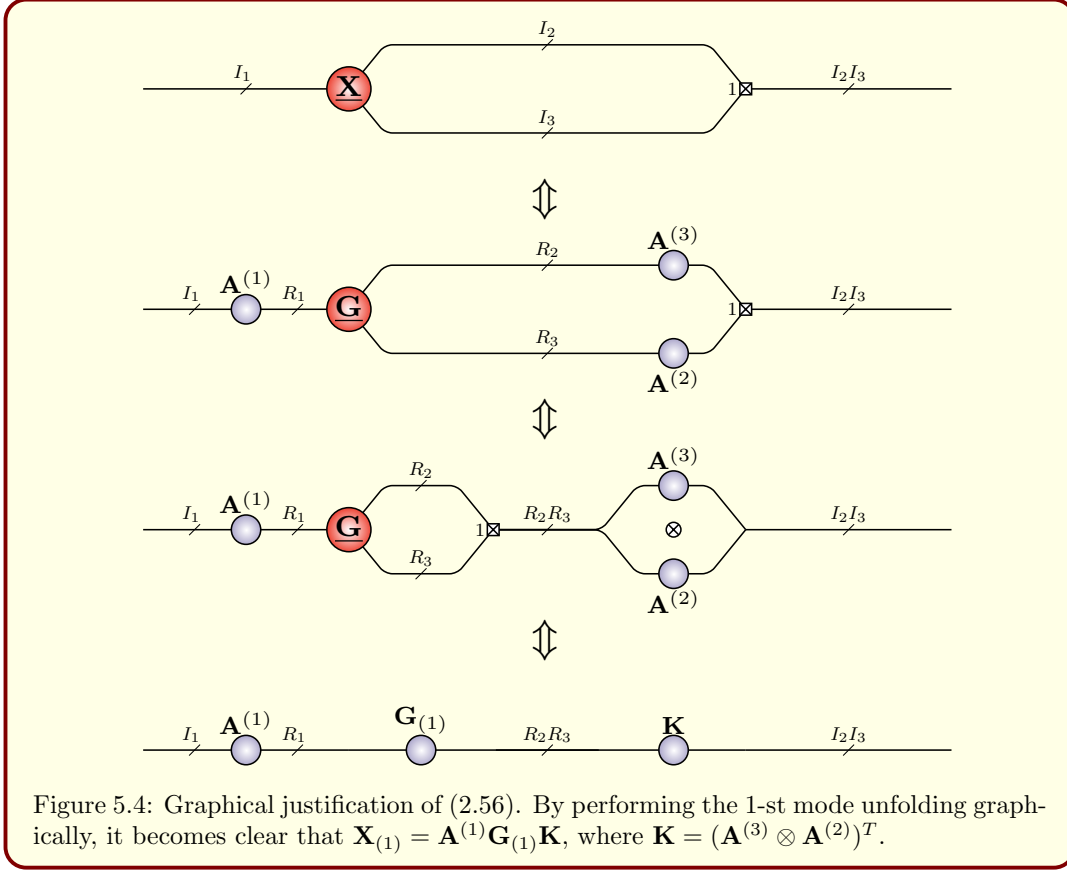
Remark 19. Any TN has several mathematical representations, which are all equivalent. To illustrate this, consider a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format, that is

$$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \quad (5.7)$$

The above is equivalent to

$$\underline{\mathbf{X}} = \mathbf{A}^{(1)} \times_2^1 (\underline{\mathbf{G}} \times_2 \mathbf{A}^{(2)}) \times_3^1 \mathbf{A}^{(3)T} \quad (5.8)$$

The equality of (5.7) and (5.8) is evident upon the inspection of Fig. 5.5, which shows a TKD expressed with the notation that is normally used for a TT, hence the requirement of the transpose on $\mathbf{A}^{(3)}$.



It is possible to manipulate its mathematical expression in some formats which may be more convenient than others, inevitably leading to a conclusion that the actual topology that defines a TN is a very crucial aspect. For example, (5.7) has the advantage over (5.8) in that it can make use of the relations shown in (2.56). This inherent duality is a powerful tool that is achieved via the graphical representation of tensors, emphasising once again their need in this branch of mathematics.

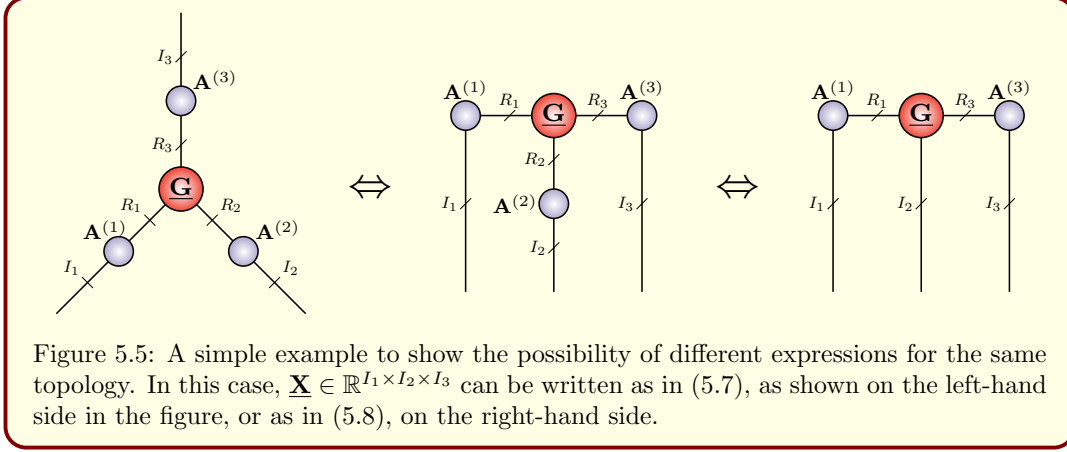
5.1.3 Graphical approach to contraction of tensors in the TT-format

It is extremely powerful to represent a tensor as a set of nodes and edges. This gives us an immediate flexibility and versatility in mathematical operations. It is possible to split links, contract cores, rotate parts, and merge TNs contracting specific cores when an original tensor is treated as a graph. In other words, the sole topology of a TN is sufficient to obtain insights which would otherwise be difficult to access as long as the mathematical rigour is preserved.

To illustrate this, a novel method to perform tensor contractions is derived, stemming solely from an intuition gained through the graphical representation. Recall the definition of contraction products provided in (2.24). A direct application of the iterative definition can be computationally expensive, as for higher order tensors it would involve a significant amount of iterative procedures. On the other hand, the same problem could be tackled using TTs which is suggested by the graphical representation of the TT illustrated on the right hand side of Fig. 2.8, as each core has information regarding the corresponding original tensor mode [48].

Given that, the first and last cores in a TT are always matrices. The question is hence whether one could exploit this to reduce an $\times_n^m$ contraction product to a simple matrix multi-

5. Computational Efficiency and Practical Implementations



plication. For simplicity, suppose it is required to perform a $\times_3^2$ contraction between a 4th-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ and a 3rd-order tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, where $I_3 = J_2$. This would result in a tensor $\underline{\mathbf{Z}}$, given by:

$$\underline{\mathbf{X}} \times_3^2 \underline{\mathbf{Y}} = \underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times I_2 \times I_4 \times J_1 \times J_3} \quad (5.9)$$

The tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ can also have their indices permuted to yield $\underline{\mathbf{X}}^\rho \in \mathbb{R}^{I_3 \times I_1 \times I_2 \times I_4}$ and $\underline{\mathbf{Y}}^\rho \in \mathbb{R}^{J_2 \times J_1 \times J_3}$, such that the contracting modes, I_3 for $\underline{\mathbf{X}}$, and J_2 for $\underline{\mathbf{Y}}$ become the first modes. Upon, performing a TT decomposition on $\underline{\mathbf{X}}^\rho$ and $\underline{\mathbf{Y}}^\rho$ respectively, this yields the cores

$$\begin{aligned} \underline{\mathbf{G}}_x^{(n)} &\in \mathbb{R}^{R_{(n-1)} \times I_n \times R_n}, \quad n = 1, 2, 3, 4 \\ \underline{\mathbf{G}}_y^{(m)} &\in \mathbb{R}^{P_{(m-1)} \times J_m \times P_m}, \quad m = 1, 2, 3 \\ P_0 &= P_3 = 1 \\ R_0 &= R_4 = 1 \end{aligned} \quad (5.10)$$

This implies that, since $I_3 = J_2$, the desired contraction in (5.9) can be obtained as a simple matrix product $\underline{\mathbf{G}}_x^{(1)} = \mathbf{A}_x \in \mathbb{R}^{I_1 \times R_1}$ and $\underline{\mathbf{G}}_y^{(1)} = \mathbf{A}_y \in \mathbb{R}^{J_2 \times P_1}$, in the form

$$\mathbf{A}_x^T \mathbf{A}_y = \mathbf{K} \in \mathbb{R}^{R_1 \times P_1} \quad (5.11)$$

This is the centre of the proposed Tensor-Train Contraction Product (TTCP) [48], a method to perform contractions between tensors that are represented in the TT-format. Observe from Fig. 5.6 that first, the cores $\mathbf{A}_x$ and $\mathbf{A}_y$ of tensors $\underline{\mathbf{X}}^\rho$ and $\underline{\mathbf{Y}}^\rho$ are linked by a standard matrix multiplication $\mathbf{A}_x^T \mathbf{A}_y$, connecting the two TTs of $\underline{\mathbf{X}}^\rho$ and $\underline{\mathbf{Y}}^\rho$. This results in the newly formed TT where the link introduced by the contraction product is consolidated into the matrix $\mathbf{K} \in \mathbb{R}^{R_1 \times P_1}$. Thus the task of reconstructing the tensor represented by the union of the two TTs yields $\underline{\mathbf{Z}}^\rho \in \mathbb{R}^{I_4 \times I_2 \times I_1 \times J_1 \times J_3}$, which is a permuted version of the tensor $\underline{\mathbf{Z}}$ in (5.9). In order to obtain the latter it is sufficient to permute $\underline{\mathbf{Z}}^\rho$ such that the resulting dimensions follow the natural anti-clockwise order obtained by the original contraction. The procedure is summarised in Algorithm 7, where the operator $(\cdot)^{T_{[1;3]}}$ permutes any given tensor such that the first and third modes are exchanged. The benefits of the TTCP are two-fold:

1. The resulting tensor $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_3^2 \underline{\mathbf{Y}}$ is automatically given in the TT-format, suitable for efficient distributed storage of then underlying information, whereby its cores still contain the modal information with respect to the individual tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$.
2. It provides a more efficient alternative to the actual contraction, in comparison to (2.24);

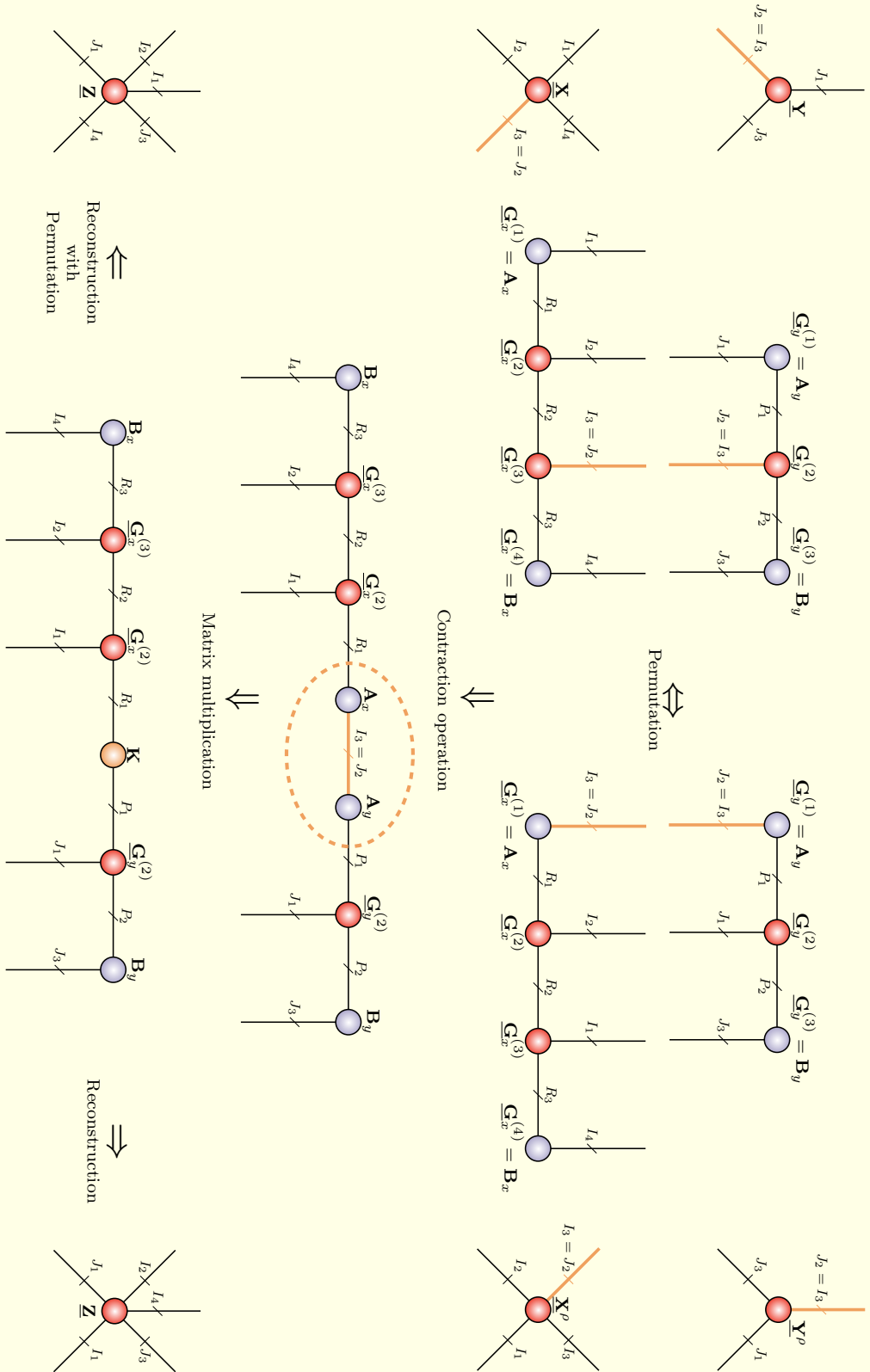


Figure 5.6: Graphical illustration of TTCP. The contraction is performed on a 4th-order tensor $\underline{\mathbf{X}}$ and a 3rd-order tensor $\underline{\mathbf{Y}}$ along common modes $I_3 = J_2$. This is achieved by exploiting the topology of TTs. The TT decompositions of the permuted versions of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ are obtained, allowing for the reduction of the contraction to a single matrix multiplication.

5. Computational Efficiency and Practical Implementations

Algorithm 7: Tensor-Train Contraction Product (TTCP)

Input : $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \dots \times J_M}$, n and m
Output: $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_n^m \underline{\mathbf{Y}}$ s.t.
 $\underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N \times J_1 \times \dots \times J_{m-1} \times J_{m+1} \times \dots \times J_M}$

- 1 Perform permutation of an original $\underline{\mathbf{X}} \rightarrow \underline{\mathbf{X}}^\rho$ such that
$$\underline{\mathbf{X}}^\rho \in \mathbb{R}^{I_n \times I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N}$$
- 2 Represent $\underline{\mathbf{X}}^\rho$ in TT format in accordance with Algorithm 4 to obtain
$$\underline{\mathbf{X}}^\rho = \mathbf{A}_x \times_2^1 \underline{\mathbf{G}}_x^{(2)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}_x^{(N-1)} \times_3^1 \mathbf{B}_x$$
- 3 Repeat steps 1 and 2 for $\underline{\mathbf{Y}}$ to obtain
$$\underline{\mathbf{Y}}^\rho = \mathbf{A}_y \times_2^1 \underline{\mathbf{G}}_y^{(2)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}_y^{(M-1)} \times_3^1 \mathbf{B}_y$$
- 4 Perform contraction $\times_n^m$ through matrix multiplication
$$\mathbf{K} = \mathbf{A}_x^T \mathbf{A}_y$$
- 5 Obtain reconstructed version of $\underline{\mathbf{Z}}^\rho$

$$\underline{\mathbf{Z}}^\rho = \mathbf{B}_x \times_2^1 \underline{\mathbf{G}}_x^{(N-1)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}_x^{(2)} \times_3^1 \mathbf{K} \times_2^1 \underline{\mathbf{G}}_y^{(2)} \times_3^1 \dots \times_3^1 \underline{\mathbf{G}}_y^{(M-1)} \times_3^1 \mathbf{B}_y$$
- 6 Perform permutation of $\underline{\mathbf{Z}}^\rho \rightarrow \underline{\mathbf{Z}}$ such that
$$\underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N \times J_1 \times \dots \times J_{m-1} \times J_{m+1} \times \dots \times J_M}$$
- 7 return $\underline{\mathbf{Z}}$

5.1.4 Complexity analysis and numerical simulations

Generally, if all tensors were stored in the TT format, instead of performing an expensive $\times_n^m$ tensor contraction product between $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \dots \times J_M}$ according to the standard definition in (2.24), which incurs $\mathcal{O}(I_1 \dots I_{n-1} I_{n+1} \dots I_N I_n J_1 \dots J_{m-1} J_{m+1} \dots J_M)$ operations, the proposed TTCP only requires $\mathcal{O}(R_1 I_n P_1)$ operations, where R_1 is the first TT rank of $\underline{\mathbf{X}}$ and P_1 is the first TT rank of $\underline{\mathbf{Y}}$.

For illustration on the savings in computational complexity by the proposed TTCP, let $I_n = I$ and $J_m = I$ for $n = 1, \dots, N$ and $m = 1, \dots, N$, and $R_1 = P_1 = R$. Then, the number of operations required (so called *time complexity*) of the tensor contraction product reduces from the original $\mathcal{O}(I^{2N-1})$, which is *exponential in the tensor order*, N , to $\mathcal{O}(IR^2)$ in the TTCP (step 4 of Algorithm 7), which is *independent of the tensor order*, and is highly efficient for a low TT rank, R , as illustrated in Section 5.1.4. Hence, the computational advantage of TTCP over standard tensor contraction product is $\mathcal{O}(I^{2N-1})/\mathcal{O}(IR^2) \approx \mathcal{O}(I^{2N-2})$.

Notice that if the original tensors need first to be represented in the TT format, then this would incur an additional computational complexity of $\mathcal{O}(RI^N)$ [127] (steps 2 and 3 of Algorithm 7). In this case, the computational advantage of TTCP would still be of the order $\mathcal{O}(I^{N-1})$. Finally, if the resulting contraction product needs to be reconstructed from the TT-format to the full tensor form, this would then incur an additional $\mathcal{O}(I^{2N-2})$ complexity, whereby the computational advantage of $\mathcal{O}(I)$ remains. In practice, however, it is desirable to store tensors in a distributed TT-format to reduce the storage and computational overheads associated with large-scale tensor networks, and only reconstruct the final tensor after perform-

5. Computational Efficiency and Practical Implementations

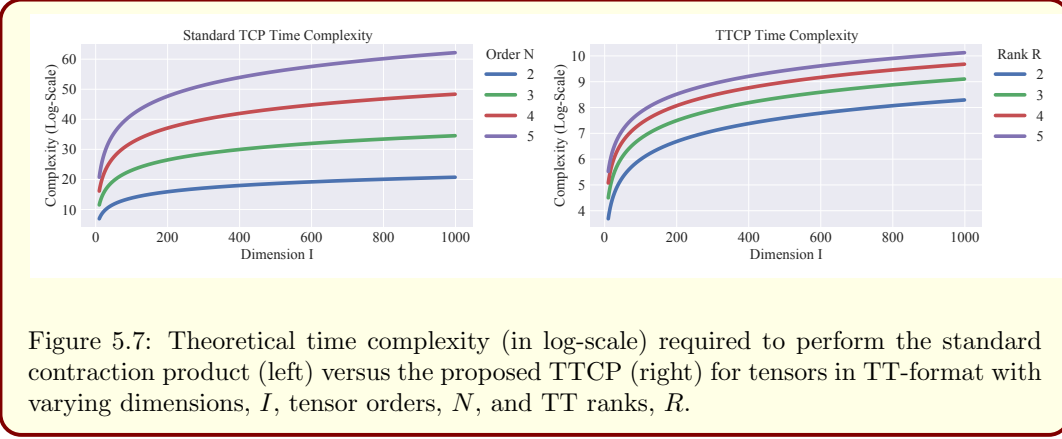


Figure 5.7: Theoretical time complexity (in log-scale) required to perform the standard contraction product (left) versus the proposed TTCP (right) for tensors in TT-format with varying dimensions, I , tensor orders, N , and TT ranks, R .

ing numerous contraction products, and ideal scenario for the proposed TTCP to provide large complexity gains over standard contractions.

The computational efficiency of TTCP can be observed in Fig. 5.8, where the two tensors, $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$, are drawn from a zero-mean, unit-variance Gaussian distribution, and three different cases are considered, that is: (i) $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{20 \times 20 \times 20}$; (ii) $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{20 \times 20 \times 20 \times 5}$; and (iii) $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{20 \times 20 \times 20 \times 5 \times 4}$. For each case, the contraction $\underline{\mathbf{X}} \times_1^1 \underline{\mathbf{Y}}$ is computed via (2.24) and via the TTCP algorithm, and the computational times are recorded for both cases. The TTCP exhibits a significant computational efficiency advantage in all cases, especially for the 4-th and 5-th order scenarios. This suggests that for higher-order tensors, expression (2.24) tends to become intractable, and the advantages of the TTCP become apparent.

The TT representation is a low-rank approximation of the original tensor, which inevitably comes with a cost of an approximation error. Generally, the higher the rank of the actual TT decomposition, the less pronounced the approximation error. However, the determination of the exact rank for tensor decompositions is generally NP-hard [128], which is prohibitive to providing a theoretical guarantee on the approximation error. Instead, in practice, the choice of rank is determined empirically for a given error tolerance, and it varies significantly across different datasets. As a rule of thumb, we find that a low TT-rank of $R_n = 4$ for $n = 1, \dots, N-1$ is often sufficient to approximate more than 90% of the original TCP (measured in explained variance) in some of the most popular machine learning benchmark datasets, as shown in Section 5.1.4.

In the same fashion which shows that a contraction product can be represented as a matrix multiplication, in specific cases contractions along multiple modes can be carried out in a single operation. Consider, for example, two N -th order tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \dots \times J_N}$. Assuming that $I_k = J_k, \forall k \neq n$, a contraction along all modes except the n -th one (“all-but- n ”) is denoted by [129]

$$\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle_{-n} = \mathbf{X}_{(n)} \mathbf{Y}_{(n)}^T \in \mathbb{R}^{I_n \times J_n} \quad (5.12)$$

To graphically justify this, consider the 3-rd order case for two general tensors, $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $I_1 = J_1, I_2 = J_2$, as illustrated on Fig. 5.9. The unfolding is performed on the 3-rd mode, i.e. the mode not taking part in the contraction, for both $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$, and the transpose on $\mathbf{Y}_{(3)}$ is taken in order to avoid a dimension mismatch. It is now not difficult to extend and accept this result for any pair of higher order tensors.

5. Computational Efficiency and Practical Implementations

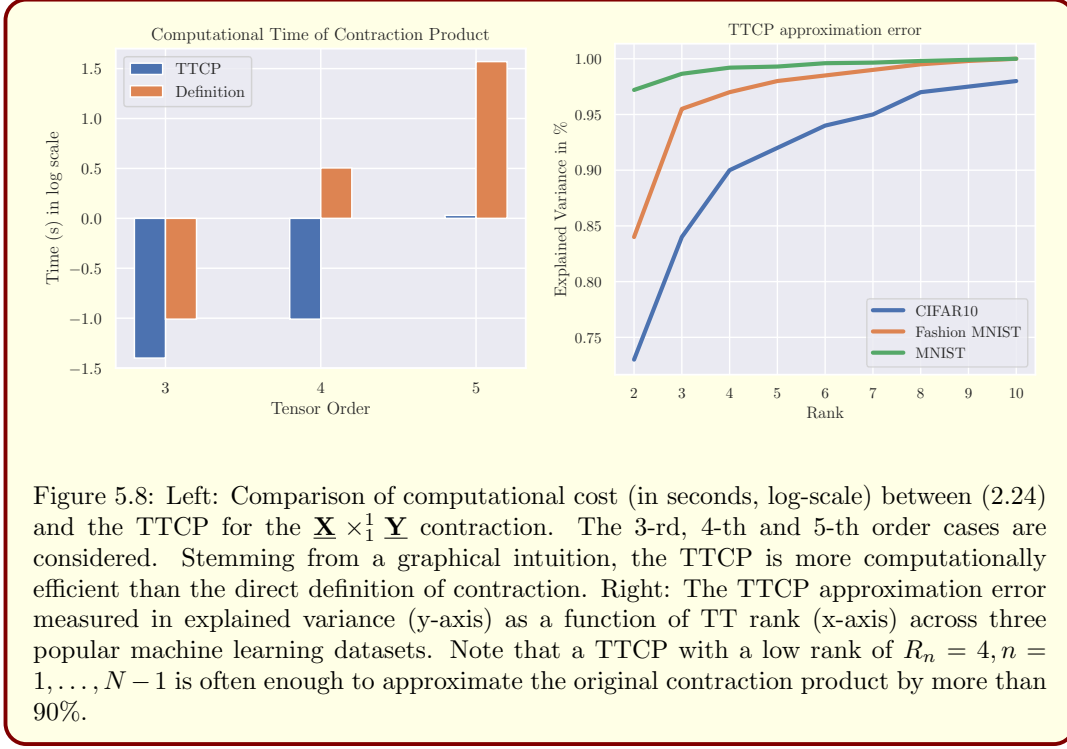


Figure 5.8: Left: Comparison of computational cost (in seconds, log-scale) between (2.24) and the TTCP for the $\underline{\mathbf{X}} \times_1^1 \underline{\mathbf{Y}}$ contraction. The 3-rd, 4-th and 5-th order cases are considered. Stemming from a graphical intuition, the TTCP is more computationally efficient than the direct definition of contraction. Right: The TTCP approximation error measured in explained variance (y-axis) as a function of TT rank (x-axis) across three popular machine learning datasets. Note that a TTCP with a low rank of $R_n = 4, n = 1, \dots, N - 1$ is often enough to approximate the original contraction product by more than 90%.

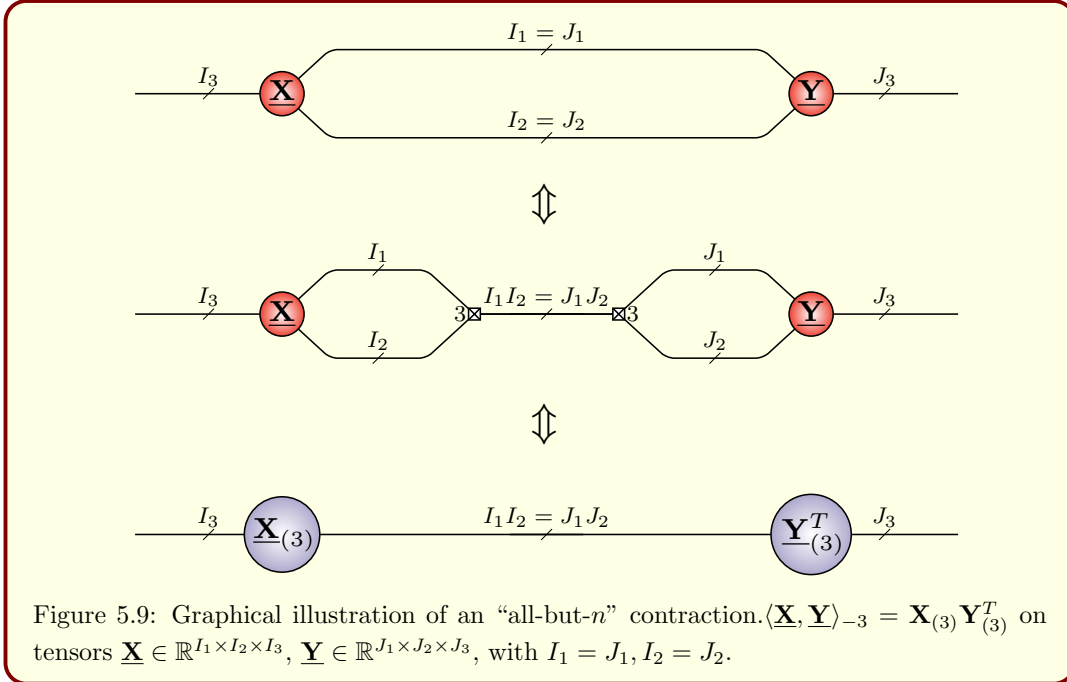


Figure 5.9: Graphical illustration of an “all-but- n ” contraction. $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle_{-3} = \underline{\mathbf{X}}_{(3)} \underline{\mathbf{Y}}_{(3)}^T$ on tensors $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $I_1 = J_1, I_2 = J_2$.

5.2 Recursive Tensor Decompositions for Streaming Big Data

5.2.1 Introduction

The ever increasing and already widespread use of multisensor technology and the associated data quantities of exceedingly large dimensionality (big data) have highlighted the limitations of standard “flat-view” matrix and vector models both in terms of accuracy and computational requirements. This calls for new, sophisticated, tools capable of coping with the sheer complexity of modern datasets [3]. With the name stemming from the Latin word *tendere*, i.e. “to stretch”, tensors are multi-dimensional generalisations of matrices and vectors, which benefit from the power of multi-linear algebra to flexibly and efficiently account for multi-way relationships. In addition, owing to their ability to exploit the underlying data structures in factor analysis and feature extraction [4, 63, 95, 97], they have found application in a wide range of disciplines, from very theoretical ones, such as numerical analysis and physics [23, 24, 39, 130], to the more applied areas of signal processing and machine learning [1, 3, 9, 131, 132, 133].

However, the natural high dimensionality of tensors implies that the applications of common algorithms on data in raw tensor format may be intractable, owing to the curse of dimensionality [26], a phenomenon whereby storage memory and the required number of operations in computation grow exponentially with the tensor order. To tackle this issue, tensor decompositions have been developed to represent tensors as a series of lower complexity multi-linear operations over latent factors. The most commonly used decompositions are the Canonical Polyadic [134], the Tensor Train [26], and the Tucker [22] decompositions (CPD, TT, and TKD, respectively). In particular, the TKD can be thought of as a higher-order form of the matrix Singular Value Decomposition (HOSVD) [24], and has found its original applications in chemical sciences [135, 136]. More recently, TKD has been widely employed in numerous studies of multi-dimensional data, owing to its straightforward physical interpretability and intuition stemming from the link with SVD. Indeed, the HOSVD represents a direct generalisation of SVD, and, as such, the large volume of SVD-related work is gradually being extended to higher order tensors. Examples of applications of HOSVD include the identification of handwritten digits and clustering [28, 137].

However, despite its advantages, due to the curse of dimensionality, computing HOSVD can be prohibitively expensive for large volume tensor data, or, more specifically, for tensors with large dimensions. One stumbling block is that HOSVD can only be solved once the entire input data is known, which, in turn, means that analytical methods require extensive multi-linear algebra operations, and are therefore unsuitable for real-time applications. Solutions have been proposed to tackle this problem, such as subspace methods and incremental SVD [138, 139], but these are still costly.

To address this issue, inspired by the work in [140], we propose a perturbation-based approach for the computation of HOSVD, referred to as Recursive HOSVD (R-HOSVD). This is achieved by a recursive estimation of mode-wise eigenvalues and eigenvectors, through the perturbation of the associated mode-wise covariance matrices [141]. Empowered by the flexibility of multi-linear algebra, the proposed method makes full use of the underlying structure in the original data which makes it possible to not only preserve intra- and inter-modal dependencies, but also to operate for online streaming data in the tensor format. The R-HOSVD is shown to be computationally efficient, as each iteration involves only simple matrix multiplications. For rigour, it is conclusively demonstrated that the proposed R-HOSVD resolves two outstanding issues with HOSVD:

- It grossly alleviates original computational costs associated with the HOSVD;
- It provides a badly needed online study of multimodal data.

Numerical examples demonstrate the validity and efficacy of the proposed framework.

5. Computational Efficiency and Practical Implementations

5.2.2 Preliminaries

Consider an order- $(N+1)$ data tensor, $\underline{\mathbf{Y}}$, constructed from order- N tensor samples, $\underline{\mathbf{X}}[m]$, as in Eq. (2.38). The HOSVD of the data tensor, $\underline{\mathbf{Y}}$, is then given by

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \underset{n=1}{\times}^{N+1} \mathbf{U}^{(n)} \quad (5.13)$$

where $\underline{\mathbf{G}} \in \mathbb{R}^{I_1 \times \dots \times I_{N+1}}$ is the core data tensor, and $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is the mode- n orthonormal factor matrix. The principle of the HOSVD operator is illustrated in Fig. 2.7 for an order-3 tensor, $\underline{\mathbf{X}}$.

The HOSVD exhibits the following properties of interest to this work, which stem from the properties of the Kronecker product

$$\begin{aligned} \text{vec}(\underline{\mathbf{X}}) &= \left(\underset{n=N}{\otimes} \mathbf{U}^{(n)} \right) \text{vec}(\underline{\mathbf{G}}) \\ \underline{\mathbf{X}}_{(n)} &= \mathbf{U}^{(n)} \underline{\mathbf{G}}_{(n)} \left(\underset{\substack{i=N+1 \\ i \neq n}}{\otimes} \mathbf{U}^{(i)^T} \right) \end{aligned} \quad (5.14)$$

Remark 20. The HOSVD of an order- $(N+1)$ tensor in Eq. (2.38) is equivalent to a sequence of principal component analyses on each mode- n unfolding of the constituent order- N sample tensors $\underline{\mathbf{X}}[1], \dots, \underline{\mathbf{X}}[M]$.

Denote the covariance matrix of the mode- n unfolding of the tensor samples by $\mathbf{R}^{(n)} \in \mathbb{R}^{I_n \times I_n}$, to yield

$$\begin{aligned} \mathbf{R}^{(n)} &= \frac{1}{M} \sum_{m=1}^M \underline{\mathbf{X}}_{(n)}[m] \underline{\mathbf{X}}_{(n)}^T[m] \\ &= \frac{1}{M} \underline{\mathbf{X}}_{(n)} \underline{\mathbf{X}}_{(n)}^T \\ &= \mathbf{U}^{(n)} \left(\frac{1}{M} \underline{\mathbf{G}}_{(n)} \underline{\mathbf{G}}_{(n)}^T \right) \mathbf{U}^{(n)^T} \\ &= \mathbf{U}^{(n)} \mathbf{\Lambda}^{(n)} \mathbf{U}^{(n)^T} \end{aligned} \quad (5.15)$$

where $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is the orthonormal eigenvector matrix and $\mathbf{\Lambda}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ the diagonal eigenvalue matrix. Therefore, the eigenvector matrix of $\mathbf{R}^{(n)}$ is equal to the mode- n factor matrix of the HOSVD of the data tensor $\underline{\mathbf{Y}}$.

Additional intuition to HOSVD can be provided by exploiting the intricate relationship between the principal component analysis and singular value decomposition. Consider the vector-variate samples, $\mathbf{x}[1], \dots, \mathbf{x}[M] \in \mathbb{R}^I$. We can compute their PCA, by the eigendecomposition of their covariance matrix, as

$$\frac{1}{M} \sum_{m=1}^M \mathbf{x}[m] \mathbf{x}^T[m] = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T \quad (5.16)$$

Alternatively, we can combine $\mathbf{x}[1], \dots, \mathbf{x}[M]$ into the data matrix $\mathbf{X} \in \mathbb{R}^{I \times M}$ and compute its SVD as

$$\mathbf{X} = [\mathbf{x}[1] \dots \mathbf{x}[M]] = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \quad (5.17)$$

From the above, it can be shown that

$$\begin{aligned}
 \mathbf{X}\mathbf{X}^T &= \sum_{m=1}^M \mathbf{x}[m]\mathbf{x}^T[m] \\
 &= \mathbf{U}(M\mathbf{\Lambda})\mathbf{U}^T \\
 &= \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T
 \end{aligned} \tag{5.18}$$

Then, the following equivalence holds, which is in essence the Cholesky decomposition

$$\mathbf{\Lambda} = \frac{1}{M}\mathbf{\Sigma}\mathbf{\Sigma}^T \tag{5.19}$$

Now, consider the vector-variate samples which are generated as vectorisations of order- N samples of an order- $(N+1)$ data tensor $\underline{\mathbf{X}}$. Then, the vectorisations take the form, $\mathbf{x}[m] = \text{vec}(\underline{\mathbf{X}}[m])$, while with the PCA of such vectors we arrive at

$$\frac{1}{M} \sum_{m=1}^M \text{vec}(\underline{\mathbf{X}}[m]) \text{vec}(\underline{\mathbf{X}}[m])^T = \left(\bigotimes_{n=N}^1 \mathbf{U}^{(n)} \mathbf{\Lambda}^{(n)} \mathbf{U}^{(n)T} \right) \tag{5.20}$$

whereas the SVD of the data matrix yields

$$\begin{aligned}
 \left[\text{vec}(\underline{\mathbf{X}}[1]), \dots, \text{vec}(\underline{\mathbf{X}}[M])^T \right] &= \underline{\mathbf{X}}_{(N+1)}^T \\
 &= \left(\bigotimes_{n=N}^1 \mathbf{U}^{(n)} \right) \underline{\mathbf{G}}_{(N+1)}^T \mathbf{U}^{(N+1)T}.
 \end{aligned} \tag{5.21}$$

We therefore obtain

$$\left(\bigotimes_{n=N}^1 \mathbf{\Lambda}^{(n)} \right) = \frac{1}{M} \underline{\mathbf{G}}_{(N+1)}^T \underline{\mathbf{G}}_{(N+1)} \tag{5.22}$$

Under the assumption that $\mathbf{\Lambda}^{(n)}$ admits the Cholesky decomposition, $\mathbf{\Lambda}^{(n)} = \mathbf{\Sigma}^{(n)}\mathbf{\Sigma}^{(n)T}$, the core data tensor is given by

$$\underline{\mathbf{G}}_{(N+1)} = \sqrt{M} \left(\bigotimes_{n=N}^1 \mathbf{\Sigma}^{(n)T} \right) \mathbf{\Theta} \tag{5.23}$$

where $\mathbf{\Theta}$ is an arbitrary rotation matrix. In practice this arbitrary rotation matrix can effectively be ignored, that is, we can choose $\mathbf{\Theta} = \mathbf{I}$, as suggested in [142, 143].

5.2.3 Recursive HOSVD

It is important to note that the HOSVD of the data tensor in online streaming scenarios results in two sets of factor matrices:

- the entity-space factor matrices of fixed dimensionality $\mathbf{U}^{(n)}$ for $n = 1, \dots, N$;
- the sample-space factor matrix $\mathbf{U}^{(N+1)} \in \mathbb{R}^{t \times t}$ which grows in dimensionality with each newly arrived sample.

Entity-space update scheme

Consider the recursion equation for the mode- n covariance matrix

$$\begin{aligned}
 \mathbf{R}^{(n)} &= \sum_{m=1}^M \alpha[m] \underline{\mathbf{X}}_{(n)}[m] \underline{\mathbf{X}}_{(n)}^T[m] \\
 &= (1 - \alpha[m])\mathbf{R}^{(n)}[m-1] + \alpha[m] \underline{\mathbf{X}}_{(n)}[m] \underline{\mathbf{X}}_{(n)}^T[m]
 \end{aligned} \tag{5.24}$$

5. Computational Efficiency and Practical Implementations

where $\alpha[m] \in (0, 1)$ is the forgetting factor, for which the following holds $\sum_{m=1}^M \alpha[m] = 1$. Without loss of generality, we can drop the time dependency notion for α . Then the matrix $\mathbf{R}^{(n)}[m]$ admits the eigendecomposition

$$\mathbf{R}^{(n)}[m] = \mathbf{U}^{(n)}[m] \mathbf{\Lambda}^{(n)}[m] \mathbf{U}^{(n)T}[m] \quad (5.25)$$

where $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ and $\mathbf{\Lambda}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ denote the orthonormal eigenvector and diagonal eigenvalue matrices, respectively.

Next, define the core data tensor based on the projection of the current sample, $\underline{\mathbf{X}}[m]$, using the preceding factor matrices, $\mathbf{U}^{(n)}[m-1]$, that is

$$\underline{\mathbf{A}} = \underline{\mathbf{X}}[m] \times_{n=1}^N \mathbf{U}^{(n)T}[m-1] \quad (5.26)$$

and introduce the following variables

$$\underline{\mathbf{A}}_{(n)} = \mathbf{U}^{(n)T}[m-1] \underline{\mathbf{X}}_{(n)}[m] \left(\bigotimes_{\substack{i=1 \\ i \neq n}}^N \mathbf{U}^{(i)T}[m-1] \right) \quad (5.27)$$

$$\mathbf{B} = \underline{\mathbf{A}}_{(n)} \underline{\mathbf{A}}_{(n)}^T$$

For computational efficiency, we can calculate $\mathbf{B}$ directly using

$$\begin{aligned} \mathbf{A} &= \mathbf{U}^{(n)T}[m-1] \underline{\mathbf{X}}_{(n)}[m] \\ \mathbf{B} &= \mathbf{A} \mathbf{A}^T \end{aligned} \quad (5.28)$$

since $\mathbf{U}^{(n)T} \mathbf{U}^{(n)} = \mathbf{I}, \forall n$. Upon substituting these definitions into Eq. (5.25), the recursive formula for the eigenvectors and eigenvalues can be written as

$$\mathbf{U}^{(n)}[m] \mathbf{\Lambda}^{(n)}[m] \mathbf{U}^{(n)T}[m] = \mathbf{U}^{(n)}[m-1] \left[(1-\alpha) \mathbf{\Lambda}^{(n)}[m-1] + \alpha \mathbf{B} \right] \mathbf{U}^{(n)T}[m-1] \quad (5.29)$$

Further, by introducing the eigendecomposition

$$\left[(1-\alpha) \mathbf{\Lambda}^{(n)}[m-1] + \alpha \mathbf{B} \right] = \mathbf{V} \mathbf{D} \mathbf{V}^T \quad (5.30)$$

where $\mathbf{V} \in \mathbb{R}^{I_n \times I_n}$ is orthonormal and $\mathbf{D} \in \mathbb{R}^{I_n \times I_n}$ is diagonal, the expression in Eq. (5.29) becomes

$$\mathbf{U}[m] \mathbf{\Lambda}[m] \mathbf{U}^T[m] = \mathbf{U}[m-1] \mathbf{V} \mathbf{D} \mathbf{V}^T \mathbf{U}^T[m-1] \quad (5.31)$$

It is now immediately obvious that

$$\begin{aligned} \mathbf{U}[m] &= \mathbf{U}[m-1] \mathbf{V} \\ \mathbf{\Lambda}[m] &= \mathbf{D} \end{aligned} \quad (5.32)$$

Remark 21. In spite of the fact that the matrix $[(1-\alpha)\mathbf{\Lambda}[m-1] + \alpha\mathbf{B}]$ has a special structure which is much simpler than that of a general covariance matrix, determining the eigendecomposition $\mathbf{V} \mathbf{D} \mathbf{V}^T$ analytically is difficult. As shown in the next section, if m is large, the problem can be solved in a much simpler way using a matrix perturbation analysis approach. The assumption of a large m underpins the streaming big data paradigm.

Consider the matrix $[(1-\alpha)\mathbf{\Lambda}[m-1] + \alpha\mathbf{B}]$. For a small α , this matrix becomes diagonally dominant and therefore its eigenvalues are close to $[(1-\alpha)\mathbf{\Lambda}[m-1] + \alpha\mathbf{B}]$, according to the Gershgorin theorem [139, 144], while its eigenvector matrix will be close to identity. The problem reduces to finding the eigendecomposition of a matrix $[(1-\alpha)\mathbf{\Lambda}[m-1] + \alpha\mathbf{B}]$, that is, an update of the diagonal matrix, $\mathbf{\Lambda}$, using the following approximations:

5. Computational Efficiency and Practical Implementations

$$\begin{aligned}\mathbf{D} &= (1 - \alpha)\mathbf{\Lambda}[m - 1] + \mathbf{P}_\Lambda \\ \mathbf{V} &= \mathbf{I} + \mathbf{P}_\mathbf{V}\end{aligned}\tag{5.33}$$

where $\mathbf{P}_\Lambda$ and $\mathbf{P}_\mathbf{V}$ are small perturbation matrices. Observe that the eigenvalue perturbation matrix $\mathbf{P}_\Lambda$ is naturally diagonal. With these definitions, we have

$$\begin{aligned}\mathbf{V}\mathbf{D}\mathbf{V}^T &= (\mathbf{I} + \mathbf{P}_\mathbf{V})\left((1 - \alpha)\mathbf{\Lambda}[m - 1] + \mathbf{P}_\Lambda\right)(\mathbf{I} + \mathbf{P}_\mathbf{V})^T \\ &= (1 - \alpha)\mathbf{\Lambda}[m - 1] + \mathbf{P}_\Lambda + \mathbf{D}\mathbf{P}_\mathbf{V}^T + \mathbf{P}_\mathbf{V}\mathbf{D} \\ &\quad + \mathbf{P}_\mathbf{V}(1 - \alpha)\mathbf{\Lambda}[m - 1]\mathbf{P}_\mathbf{V}^T + \mathbf{P}_\mathbf{V}\mathbf{P}_\Lambda\mathbf{P}_\mathbf{V}^T\end{aligned}\tag{5.34}$$

Assuming that the second-order perturbation terms are negligible, upon equating Eq. (5.34) to $[(1 - \alpha)\mathbf{\Lambda} + \alpha\mathbf{B}]$, we have

$$\alpha\mathbf{B} = \mathbf{P}_\Lambda + \mathbf{D}\mathbf{P}_\mathbf{V}^T + \mathbf{P}_\mathbf{V}\mathbf{D}\tag{5.35}$$

Since we know that $\mathbf{V}$ is orthonormal we can therefore substitute $\mathbf{V} = \mathbf{I} + \mathbf{P}_\mathbf{V}$ in $\mathbf{V}\mathbf{V}^T = \mathbf{I}$, and upon using the assumption $\mathbf{P}_\mathbf{V}\mathbf{P}_\mathbf{V}^T \approx 0$, this reveals the skew-symmetry $\mathbf{P}_\mathbf{V} = -\mathbf{P}_\mathbf{V}^T$.

The fact that $\mathbf{P}_\Lambda$ and $\mathbf{D}$ are diagonal can be used to find the solution to the perturbation matrices from Eq. (5.35) in the form

$$\begin{aligned}\{\mathbf{P}_\Lambda\}_{i,i} &= b_{i,i} \\ \{\mathbf{P}_\mathbf{V}\}_{i,j} &= \begin{cases} \frac{b_{i,j}}{\lambda_j + b_{j,j} - \lambda_i - b_{i,i}} & i \neq j \\ 0 & i = j \end{cases}\end{aligned}\tag{5.36}$$

where $b_{i,j} = \{\alpha\mathbf{B}\}_{i,j}$ and $\lambda_i = \{(1 - \alpha)\mathbf{\Lambda}[m - 1]\}_{i,i}$

Sample-space update scheme

Once all entity-space factor matrices $\mathbf{U}^{(n)}[m]$ for $n = 1, \dots, N$ are obtained, we can use the relationship in Eq. (5.23) to retrieve the core data tensor using

$$\underline{\mathbf{G}}_{(N+1)}[m] = \sqrt{m} \left(\bigotimes_{n=N}^1 \mathbf{\Sigma}^{(n)T}[m] \right)\tag{5.37}$$

where $\mathbf{\Sigma}^{(n)}[m] = \mathbf{\Lambda}^{(n)\frac{1}{2}}[m]$, observe that this is computationally cheap since $\mathbf{\Lambda}^{(n)}[m]$ is diagonal. Then, the sample-space factor matrix can be directly obtained from

$$\mathbf{U}^{(N+1)} = \underline{\mathbf{X}}_{(N+1)}[m] \left(\bigotimes_{n=N}^1 \mathbf{U}^{(n)}[m] \right) \underline{\mathbf{G}}_{(N+1)}^{-1}[m]\tag{5.38}$$

where $\underline{\mathbf{Y}} = \text{stack}_{N+1}(\underline{\mathbf{X}}[1], \dots, \underline{\mathbf{X}}[m])$ is the data tensor which contains all samples up to and including time instant m . Algorithm 8 outlines the end-to-end procedure of performing recursive HOSVD for a given stream of multi-dimensional data samples.

5.2.4 Implementation considerations

The role of the forgetting factor α

When dealing with stationary data, the individual samples should be weighted equally, and as such the forgetting factor is set to $\alpha[m] = \frac{1}{m}$. In a non-stationary environment, a first-order dynamical forgetting strategy could be employed by selecting a fixed decay rate. We can therefore set a small $\alpha \in (0, 1)$. Considering that the average memory depth of this recursion is $\frac{1}{\alpha}$ samples, the choice of the forgetting factor presents a trade-off between tracking capability and estimation variance.

5. Computational Efficiency and Practical Implementations

Algorithm 8: Recursive HOSVD for streaming multi-dimensional data

Input : Stream of tensor data samples $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$
Output: $\underline{\mathbf{G}}[m], \mathbf{U}^{(n)}[m]$ updated upon arrival of a new tensor data sample $\underline{\mathbf{X}}[m]$

- 1 Initialise $\mathbf{U}^{(n)}[0]$ and $\mathbf{\Lambda}^{(n)}[0]$ for $n = 1, \dots, N$
- 2 **for** $m=1, \dots$ **do**
- 3 Wait for a new input data sample $\underline{\mathbf{X}}[m]$
- 4 **for** $n = 1, \dots, N$ **do**
- 5 $\mathbf{A} \leftarrow \mathbf{U}^{(n)T}[m-1] \underline{\mathbf{X}}_{(n)}[m]$
- 6 $\mathbf{B} \leftarrow \mathbf{A} \mathbf{A}^T$
- 7 Find perturbation matrices $\mathbf{P}_V$ and $\mathbf{P}_\Lambda$ from Eq. (5.36)
- 8 $\hat{\mathbf{U}} \leftarrow \mathbf{U}^{(n)}[m-1](\mathbf{I} + \mathbf{P}_V)$
- 9 $\hat{\mathbf{\Lambda}} \leftarrow (1 - \alpha) \mathbf{\Lambda}^{(n)}[m-1] + \mathbf{P}_\Lambda$
- 10 $\mathbf{T} \leftarrow \text{Diag}(\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_{I_n})$ where $\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1 \dots \hat{\mathbf{u}}_{I_n}]$
- 11 $\mathbf{U}^{(n)}[m] \leftarrow \hat{\mathbf{U}} \mathbf{T}^{-1}$
- 12 $\mathbf{\Lambda}^{(n)}[m] \leftarrow \hat{\mathbf{\Lambda}} \mathbf{T}^2$
- 13 **end for**
- 14 $\underline{\mathbf{Y}} \leftarrow \text{stack}_{N+1}(\underline{\mathbf{X}}[0], \dots, \underline{\mathbf{X}}[m])$
- 15 $\mathbf{\Sigma}^{(n)}[m] \mathbf{\Sigma}^{(n)T}[m] \leftarrow \mathbf{\Lambda}^{(n)}[m]$; // Cholesky factorisation
- 16 $\underline{\mathbf{G}}_{(N+1)}[m] \leftarrow \left(\bigotimes_{n=N}^1 \mathbf{\Sigma}^{(n)T}[m] \right)$
- 17 $\mathbf{U}^{(N+1)}[m] \leftarrow \underline{\mathbf{Y}}_{(N+1)}[m] \left(\bigotimes_{n=N}^1 \mathbf{U}^{(n)}[m] \right) \underline{\mathbf{G}}_{(N+1)}^{-1}[m]$
- 18 **end for**
- 19 **return** $\underline{\mathbf{G}}[m], \mathbf{U}^{(n)}[m] \quad \forall m, n$

Initialisation

In practice, we can initialize the operation, even without performing an HOSVD, by setting the estimated eigenvector matrix to identity, $\mathbf{U}^{(n)}[m] = \mathbf{I} \in \mathbb{R}^{I \times I}$ and the eigenvalues to the sample variances of each input entry over M samples, $\mathbf{\Lambda}^{(n)}[0] = \text{diag}(\mathbf{R}^{(n)}[M])$, for $n = 1, \dots, N$. In order to guarantee the accuracy of the first order perturbation approximation, we need to choose the forgetting factor α to be very close to 0 such that the time-average defining $\mathbf{R}^{(n)}[m]$ becomes very close to the deterministic ensemble average for $(1 - \alpha) \approx 1$. This key assumption relies on the property

$$\lim_{m \rightarrow \infty} \mathbf{R}^{(n)-1}[m] \mathbf{R}^{(n)}[m] = (1 - \alpha) \mathbf{I} \quad (5.39)$$

where $\mathbf{R}^{(n)} = E\{\mathbf{X}_{(n)} \mathbf{X}_{(n)}^T\}$.

Computational complexity analysis

Upon combining the individual contributions outlined in Table 5.1, and neglecting the scaling constants, it follows that the mode- n computational complexity is $\sim \mathcal{O}(\sum_{n=1}^N I_n(P + 1 + I_n + I_n^2))$. Assuming that each mode is comparable in size, i.e. $I_1 \approx I_2 \approx \dots \approx I_N$, or, more generally, $I_n \approx I$, then a single update step requires $\sim \mathcal{O}(NI(P + 1 + I + I^2))$ computations. By the same assumption, the HOSVD of an order- N tensor has computational complexity of $\sim \mathcal{O}(NPI)$. Recall that the data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N \times M}$ is of order $(N + 1)$, then the HOSVD of $\underline{\mathbf{X}}$ at a time-step m exhibits the complexity $\sim \mathcal{O}((N + 1)PI m)$. Therefore, the requirement for the proposed R-HOSVD to be more computationally efficient can be formalised as

5. Computational Efficiency and Practical Implementations

$$NI(P + 1 + I + I^2) < (N + 1)PI m \quad (5.40)$$

Considering the dominant terms only, the requirement reduces to $m > I$, and the computational benefits of the recursive HOSVD become obvious.

Table 5.1: Computational complexity breakdown of R-HOSVD	
Algorithm Step	Complexity
$\mathbf{A} \leftarrow \mathbf{U}^{(n)T} [m - 1] \mathbf{X}_{(n)} [m]$ $\mathbf{B} \leftarrow \mathbf{A} \mathbf{A}^T$	$\mathcal{O}(2I_n P)$
Find $\mathbf{P}_V$ and $\mathbf{P}_\Lambda$	$\mathcal{O}(4I_n^2)$
$\hat{\mathbf{U}} \leftarrow \mathbf{U}^{(n)} [m - 1] (\mathbf{I} + \mathbf{P}_V)$ $\hat{\Lambda} \leftarrow (1 - \alpha) \mathbf{\Lambda}^{(n)} [m - 1] + \mathbf{P}_\Lambda$	$\mathcal{O}(I_n^2 + I_n^3) + \mathcal{O}(2I_n)$
$\mathbf{U}^{(n)} [m] \leftarrow \hat{\mathbf{U}} \mathbf{T}^{-1}$ $\mathbf{\Lambda}^{(n)} [m] \leftarrow \hat{\Lambda} \mathbf{T}^2$	$\mathcal{O}(I_n^2) + \mathcal{O}(I_n)$

5.2.5 Simulations

To demonstrate the validity of the proposed framework in a regime-changing environment, we generated two order-4 data tensors, $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^{3 \times 4 \times M}$, in the form of a vector of M order-3 tensor-valued samples, each drawn from a different tensor-variate Gaussian distribution [145] with arbitrarily chosen parameters. These were then stacked together so as to form a single data tensor of order-4, $\mathbf{Y} = \text{stack}_4(\mathbf{X}_1, \mathbf{X}_2) \in \mathbb{R}^{3 \times 4 \times 5 \times 2M}$. A total of 1000 Monte Carlo simulations were carried out with the forgetting factor set to a constant $\alpha = 10^{-3}$. The accuracy was measured in terms of the principal angle [146] between the estimated and actual subspaces for each mode- n factor matrix, that is, if the mode- n actual ($\mathbf{U}^{(n)}$) and estimated ($\hat{\mathbf{U}}^{(n)}$) factor matrices comprise of the columns $\mathbf{u}_i^{(n)}$ and $\hat{\mathbf{u}}_i^{(n)}$, respectively, then the angular error at mode- n is given by

$$e_n = \min \left\{ \frac{\hat{\mathbf{u}}_i^{(n)T} \mathbf{u}_i^{(n)}}{\|\hat{\mathbf{u}}_i^{(n)}\| \|\mathbf{u}_i^{(n)}\|} \mid \hat{\mathbf{u}}_i^{(n)} \in \hat{\mathbf{U}}^{(n)}, \mathbf{u}_i^{(n)} \in \mathbf{U}^{(n)} \right\} \quad (5.41)$$

The results of 1000 independent Monte Carlo runs were averaged to obtain the convergence curves for the factor matrix, as shown in Fig. 5.10. Convergence is observed for all modes, even in the event of the regime-change, as indicated by the change in distribution at $m = 10000$. The algorithm was implemented and using Python Higher-Order Tensor ToolBOX (HOTTBOX) [81].

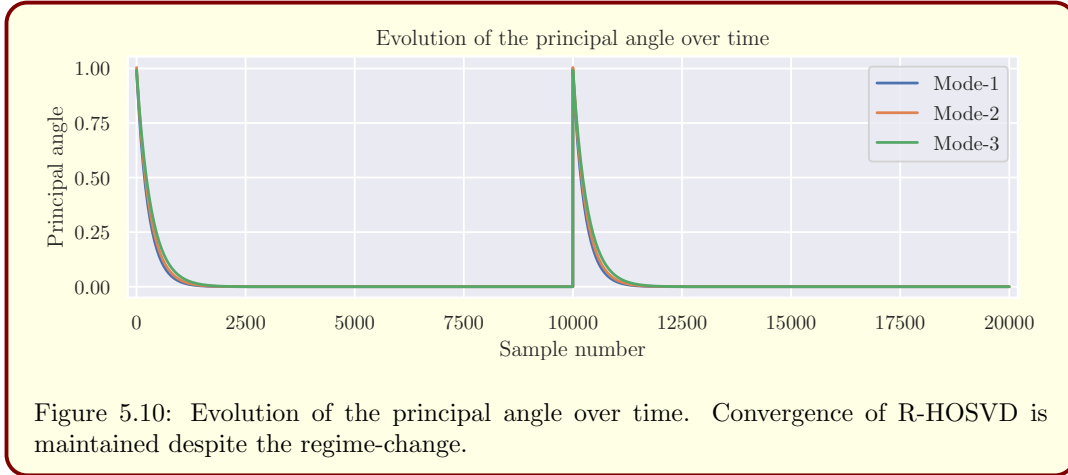
5.3 Higher Order Tensors ToolBOX

5.3.1 Introduction

Over the past years, several libraries for dealing with multi-dimensional data have emerged:

- *Tensor Toolbox* [43] is one of the first attempts to provide users with classes and functions for manipulating dense, sparse and structured tensors as well as fundamental tensor decompositions using MATLAB's object-oriented features;

5. Computational Efficiency and Practical Implementations



- Another popular choice is *Tensorlab* [44] which offers users various tools for tensor computations and complex optimisations. In addition, it also implements a number of algorithms dealing with large-scale datasets and more advanced factorisation, such as the LL1 and Block Term Decomposition (BTD) [74, 147], and supports quite intuitive tools for basic visualisations of N-dimensional data.

Both these toolboxes are well structured and provide extensive documentation supported with various examples; they also require a proprietary MATLAB platform. However, in the world of free software, a fair share of packages are written in Python, and include

- The most well received library by the community is *TensorLy* [45]. It was developed with simplicity in mind and has a flexible backend system which allows to perform tensor decomposition algorithms at scale and over a range of hardware setups, from a single CPU through to a distributed execution on multiple GPUs. *TensorLy* also comprises implementations of some machine learning approaches, but its main focus remains on optimised and efficient performance;
- The *TensorD* toolbox [46] was designed as a highly modular piece of software, specifically tailored to the most prominent machine learning framework, the TensorFlow. Although it allows us to transform an idea into a result very quickly, it supports implementations of only CPD and TKD.

Several other libraries exist which primary focus on the Tensor Train Decomposition (TTD), a factorisation originally proposed by a research group from the Skolkovo Institute of Science and Technology in 2011. The same group also released a Python implementation of their algorithms within a dedicated toolbox [148] whose functionalities were then extended to support utilisation of GPUs for efficient computations, batch processing and automatic differentiation [149]. Another package for tensor network learning with PyTorch [150] takes a somewhat orthogonal approach to establishing a common interface, where the existing multi-linear models, i.e. CPD, TKD, TT, are represented as a particular case of tensor networks [1]. Thus, decomposing, manipulating, and reconstructing tensors can be (to some extent) abstracted away from the particular decomposition format.

In recent years, we have also witnessed a tendency to open-source implementation of newly proposed algorithms or computational optimisation advances in the field of multi-linear analysis. This has brought to light a number of publicly available stand-alone projects¹²³ for which the

¹https://github.com/cohenjer/Tensor_codes

²<https://github.com/andrewssobral/mctc4bmi>

³https://github.com/yuanlonghao/T3C_tensor_completion

5. Computational Efficiency and Practical Implementations

documentation or explanations of installation steps are often incomplete, so the community generally struggles to incorporate these into a custom experiment.

To summarise, all of the aforementioned libraries and projects have found their niche within tensor decompositions and applications, and exhibit the following commonalities:

- They utilise tensors only as N-dimensional arrays and do not exploit complementary meta-information about the data a tensor represents;
- Lack of visualisation tools, which are extremely beneficial for analysis, multi-disciplinary efforts or educational purposes;
- No sufficient focus on unifying the existing machine learning approaches which are specific to inherently multi-dimensional data.

In order to address these shortcomings, a self-contained open source library, **HOTTBOX**, has been developed. It is written in Python, as one of the top three popular languages world-wide and the most friendly and easy to get started with even for the newcomers. Our journey of exploring available functionalities of this package will be supported with short code snippets that serve as guiding points for those who wish to enter a fascinating field of tensor decomposition and their applications. All that is required is to have a working installation of Python 3 and the following packages: *NumPy* [151], *Scikit-Learn* [82] and *HOTTBOX* [81]. All these can be installed directly from the Python Package Index with:

```
pip install numpy scikit-learn hottbox
```

5.3.2 Ecosystem of HOTTBOX

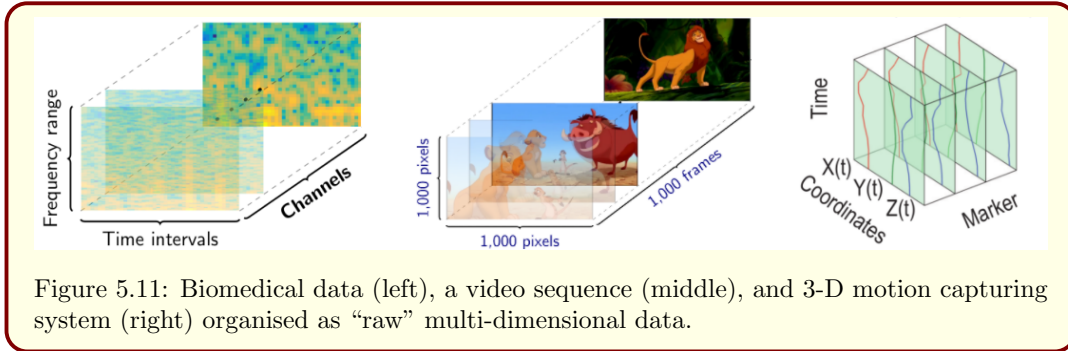


Figure 5.11: Biomedical data (left), a video sequence (middle), and 3-D motion capturing system (right) organised as “raw” multi-dimensional data.

Various types of data can be represented in the form of an N-dimensional array. Most often, this is achieved through a combination of mathematical construction and experimental design. For example:

- By stacking time-frequency representations of each channel that records brain electrical activity during a particular task or stimulus (each slice on the left-most panel in Fig. 5.11);
- A video clip can be broken down into a set of consecutive frames which are stacked into a tensor (middle panel in Fig. 5.11);
- Optical motion capture system contains several markers, each recording information about position change in a 3-D space over a period of time (right most-panel in Fig. 5.11).

5. Computational Efficiency and Practical Implementations

The HOTTBOX builds on top of NumPy and provides a top level API for data representation and manipulation operations that are specific to multi-linear analysis. Creation of a **Tensor** object is straightforward and requires only an N-dimensional array which should be passed as an input:

```
1 import numpy as np
2 from hottbox.core import Tensor
3
4 # Create 3-d array of data
5 array_3d = np.arange(24).reshape((2, 3, 4))
6
7 # Create tensor
8 X = Tensor(array_3d)
9
10 print(X)
11 # This tensor is of order 3 and consists of 24 elements.
12 # Sizes and names of its modes are (2, 3, 4) and ['mode-0', 'mode-1', 'mode-2'] respectively.
```

The HOTTBOX is now ready to perform basic tensor operations from Section 2.1, as it is equipped with **Tensor** objects which implement the corresponding methods. For example, the most common transformations of mode- n unfolding Eq. (2.2) and generic folding Eq. (2.30) are available through an intuitive abstraction API layer. Moreover, it is also possible to chain methods together, which is extremely convenient for performing sequential mode- n product transformations Eq. (2.15).

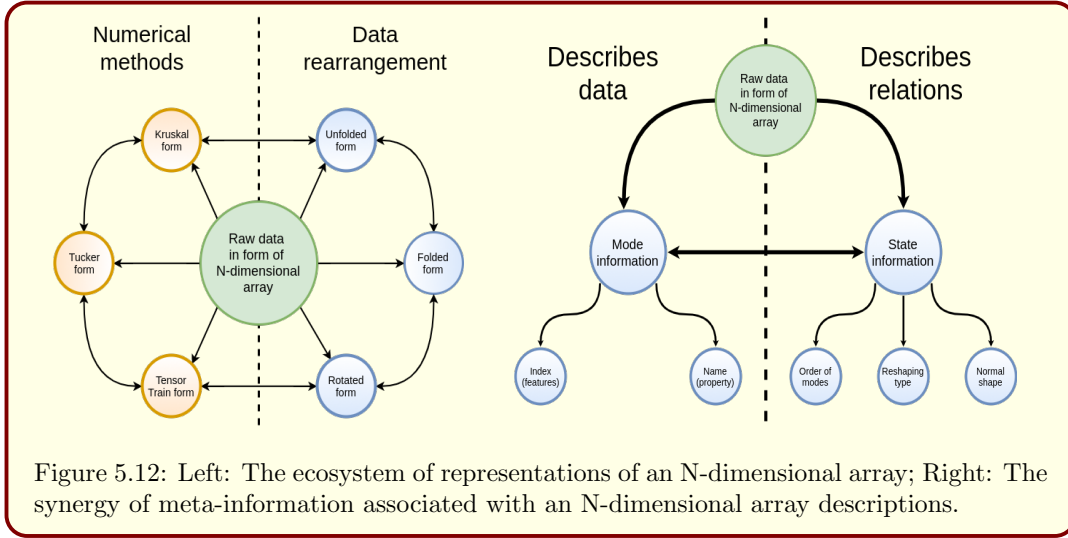
```
14 X.unfold(mode=0)
15 X.data
16 # array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11],
17 #        [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]])
18
19 X.fold()
20 X.data
21 # array([[[ 0,  1,  2,  3],
22 #          [ 4,  5,  6,  7],
23 #          [ 8,  9, 10, 11]],
24 #        [[12, 13, 14, 15],
25 #          [16, 17, 18, 19],
26 #          [20, 21, 22, 23]]])
27
28
29 A = np.arange(15).reshape(5, 3)
30 B = np.arange(24).reshape(6, 4)
31 X.mode_n_product(A, mode=1).mode_n_product(B, mode=2)
32 print(X)
33 # This tensor is of order 3 and consists of 60 elements.
34 # Sizes and names of its modes are (2, 5, 6) and ['mode-0', 'mode-1', 'mode-2'] respectively.
```

By default, all operations directly change the underlying data array of a tensor. However, in some cases it might be beneficial to obtain result as a separate data structure which merely requires specifying an additional parameter, i.e. `inplace=False`.

Remark 22. *In order to be consistent with Python indexing, the count of modes and elements within data structures starts from zero.*

5.3.3 Core Structural Components

Each dimension within an N-dimensional array can be associated with a certain property, or mode of the raw data. At the same time, this characteristic is described by a set of distinct



features. To show this, assume that a simple data rearrangement procedure (e.g. folding, unfolding) of the raw data was performed which yielded a different view of data at hand. However, this manipulation did not change the original properties of the underlying data array, but instead it changed the relationships between modes. We shall refer to such a transformation of a “raw” N-dimensional data as a *change of state*. To summarise:

- *Mode* of the tensor is defined by the name of the property it represents and names of the features that describe this property;
- *State* of the tensor is defined by transformations applied to the data array;
- *Meta-data* about a tensor is a record of information about the modes and the state it is currently in.

Oftentimes, N-dimensional data contains a considerable amount of repetitive and redundant information. By applying different numerical methods, i.e. tensor decompositions, this information can be represented in a more compact and efficient way. There are three major conceptually different factorisation types and computational algorithms associated with tensors:

- *Kruskal form* represents the original data as a list of matrices, each of which corresponds to a particular dimension of an array. It also imposes a one-to-one relation between column vectors of these factor matrices;
- *Tucker form* is similar to Kruskal form, but every vector of one factor matrix is related to all vectors of the other factor matrices through a core tensor;
- *Tensor Train form* represents a tensor as a set of sparsely interconnected matrices and core tensors of low order, and therefore only adjacent components have explicit relation.

To this end, the core structural components of HOTTBOX are the `Tensor`, `TensorCPD`, `TensorTKD` and `TensorTT` classes which bring together the raw, Kruskal, Tucker, and Tensor Train formats respectively. Examples of how to interact with these representations within the HOTTBOX are given in the code snippets below.

```
1 import numpy as np
2 from hottbox.core import TensorCPD
```

5. Computational Efficiency and Practical Implementations

```
3
4 # Create Kruskal form
5 tensor_cpd = TensorCPD(fmat=[A, B, C], core_values=Lambda)
6
7 # Convert to Tucker form
8 tensor_tkd = tensor_cpd.to_tucker()
9
10 # Add metadata
11 tensor_tkd.set_mode_name(["Year", "Month", "Day"])
12
13 # Reconstruct to original form
14 tensor = tensor_tkd.reconstruct()
15
16 print(tensor)
17 # This tensor is of order 3 and consists of 210 elements.
18 # Sizes and names of its modes are (5, 6, 7) and ['Year', 'Month', 'Day'] respectively.
```

It is important to understand that the characteristics of modes of N-dimensional data remain the same regardless of which form they currently represent. Therefore, to suit the inherent characteristics of tensors, it is highly advantageous to have a unified API, which is provided within HOTTBOX and is capable of:

- Allowing us to switch between different tensor forms seamlessly;
- Preserving information about the underlying tensor characteristics;
- Giving an opportunity to utilise meta-information with ease.

5.3.4 Supported Algorithms

Another aim of the HOTTBOX library is to facilitate the usage of a wide range of existing numerical methods which are available for analysis of inherently multi-dimensional data. The HOTTBOX splits implementation of such algorithms into several modules dedicated to different aspects of multi-way analysis.

Tensor factorisation module

The *tensor factorisation module* consists of implementations of fundamental tensor decompositions which include the Canonical Polyadic (Algorithm 1), Higher Order Singular Value (Algorithm 2) and Tensor Train decompositions (Algorithm 4), that can be used to represent a “raw” multi-modal data as one of the structures covered in Section 5.3.3. A set of these algorithms is further extended with different variations which have been proposed in the literature on optimised processing of multi-dimensional arrays. For example, the Higher Order Orthogonal Iteration Decomposition (Algorithm 3) aids the actual computation of the best multi-linear approximation, while the Randomised Canonical Polyadic Decomposition [152] significantly reduces both memory requirements and computational speed, without sacrificing on the precision of CP factorisation. The HOTTBOX aims to follow the same standards adopted by the main projects of the Python scientific community and exposes only a single functional method, **decompose**, that hides complexity and all implementation details. Thus, even not very experienced or computing savvy users can incorporate multi-linear analysis into their research with ease. An example of the straightforward way HOTTBOX deals with tensor decompositions is illustrated in the code snippet below.

```
1 import numpy as np
2 from hottbox.core import Tensor
3 from hottbox.algorithms.decomposition import CPD, RandomisedCPD
4
```

5. Computational Efficiency and Practical Implementations

```
5 array_3d = np.arange(30000).reshape((20, 30, 50))
6 tensor = Tensor(array_3d.astype(np.float))
7
8 # Unified interface to perform decomposition
9 for algorithm in [RandomisedCPD(), CPD()]:
10     tensor_cpd = algorithm.decompose(tensor, rank=(3,))
```

The most common way to evaluate performance of an algorithm is by computing the residual error of approximation:

$$\frac{\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2}{\|\mathbf{X}\|_F^2} \quad (5.42)$$

where $\mathbf{X}$ is the original multi-dimensional array and $\hat{\mathbf{X}}$ in the above example is reconstruction of the Kruskal form of a tensor, obtained by $\mathbf{A} \times_1 \mathbf{B} \times_2 \mathbf{C}$. Since this metric is frequently employed throughout different stages of multi-linear analysis, the HOTTBOX provides the corresponding utility function for user convenience, as listed in the code snippet below.

```
12 from hottbox.metrics.decomposition import residual_rel_error
13
14 rel_error = residual_rel_error(tensor, tensor_cpd)
15
16 print(f'Relative error of Kryskal form = {rel_error}')
17 # Relative error of Kryskal form = 0.0005035
```

Another subset of tensor decomposition methods within HOTTBOX is the *data fusion and feature extraction* module, which is composed of the Coupled Matrix and Tensor factorisation [153] and PARAFAC2 [154] that provide an interface identical to the one considered above. Both algorithms can be employed to jointly analyse a multi-way array with a matrix or a collection of matrices that share a common mode (which describes similar data properties) in order to extract a latent structure that governs the underlying processes. In spite of the latter model being able to simultaneously factorise a collection of only 2-dimensional data structures, PARAFAC2 is still attributed to multi-way analysis as a variant of the CP representation with relaxed constraints.

Classification and regression module

The *classification and regression* module includes the implementations of the Least Squares Support Tensor Machine (LSSTM) and Tensor Ensemble Learning (TEL) algorithms (illustrated on Fig. 4.1) which respectively generalise concepts and ideas of the existing popular “flat-view” matrix counterparts, i.e. the Support Vector Machine and Ensemble Learning frameworks [1, 155]. Thus, the LSSTM and TEL can be seen as a natural extension to the case of tensor-valued data samples.

Several different TEL algorithms have been incorporated into the HOTTBOX, all sharing a similar interface. Here, we consider a particular implementation, **TelVI**, that was employed for simulations presented in Section 4.4. First in order to extract latent components, every raw multi-dimensional sample has to be represented in the Tucker form through the HOSVD algorithm with the fixed multi-linear rank- (R_a, R_b, R_c) , as shown in the code snippet below.

```
1 from hottbox.algorithms.decomposition import HOSVD
2
3 # Generic function to obtain data samples and labels
4 raw_data, labels = get_data()
5
6 # Represent each sample in Tucker form and store it in a list
7 hosvd = HOSVD()
```

5. Computational Efficiency and Practical Implementations

```
8 R_a, R_b, R_c = 5, 5, 2
9 data_decomposed = list()
10 for sample in raw_data:
11     tensor_tkd = hosvd.decompose(sample, rank=(R_a, R_b, R_c))
12     data_decomposed.append(tensor_tkd)
```

The above code snippet corresponds to the top right of the Fig. 4.2. Since the HOSVD is applied to every sample individually, it makes no difference whether all samples are processed as a batch or after splitting them into the train and test subsets. Next, a `TelVI` classifier needs to be initialised, which expects a collection of separate base learners, each associated with a particular factor vector of the Tucker form, as depicted at the bottom of the Fig. 4.2. In other words, the required number is determined by the sum of multi-linear rank values used for the HOSVD. We will proceed with the implementation of the standard Support Vector Machine (SVM) provided by the `scikit-learn`, one of the most popular Python package for classical ML [82], as described in the code snippet below.

```
16 from sklearn.svm import SVC
17 from hottbox.algorithms.classification import TelVI
18
19 # Prepare base learners
20 base_classifiers = list()
21 for _ in range(R_a + R_b + R_c):
22     base_classifiers.append(SVC())
23
24 # Initialise classifier
25 telvi = TelVI(base_clf=base_classifiers, probability=True)
```

Here, we have created a set of homogeneous base classifiers, however, the parameter `base_clf` can accommodate a mix of heterogeneous objects representing ML algorithms, e.g. logistic regression, neural network etc. The only restriction imposed on the members of `base_clf` is that each should contain an implementation of methods `fit`, `predict` and `predict_proba`, i.e. they should adhere scikit-learn type API [156].

Finally, the components of the Tucker representations of raw data through HOSVD need to be reorganised in a set of new surrogate datasets, each containing an incomplete information about the original samples. Next, these datasets are feed into the corresponding base learners (see bottom of Fig. 4.2) during the training stage. The so-obtained series of independent hypotheses about the underlying processes are then utilised to classify the previously unseen data samples which is achieved by aggregating the individually learned knowledge through the weighted majority vote. For user convenience, all steps of this procedure are encapsulated and available via the methods `fit` and `score` for the training and testing stages respectively; see the code snippet below.

```
30 from sklearn.model_selection import train_test_split
31
32 # Split into train and test datasets
33 data = train_test_split(data_decomposed, labels, test_size=0.3)
34
35 # Train classifier and validate performance
36 X_train, X_test, y_train, y_test = data
37 telvi.fit(X_train, y_train)
38 score = telvi.score(X_test, y_test)
```

It is a common practice to perform hyper-parameter tuning of ML algorithms in order to enhance their performance. Recall that `TelVI` employs a series of independent base classifiers. Thus, a grid search for optimal parameter can be conducted for each of them with the use of

5. Computational Efficiency and Practical Implementations

`grid_search` method. This requires us to specify a list of key value pairs, each defining a name of the hyper-parameter and a set of corresponding values to try, as depicted in the code snippet below.

```
42 hyper_params = list()
43 for _ in base_classifiers:
44     hyper_params.append({"kernel": ["rbf", "linear", "poly"]})
45
46 telvi.grid_search(X_train, y_train, hyper_params)
47
48 # Train classifier with tuned parameters and obtain new results
49 telvi.fit(X_train, y_train)
50 score = telvi.score(X_test, y_test)
```

It was our top priority to make API as close as possible to standards established within the scientific community so that anyone familiar with existing machine learning libraries would be able to utilise HOTTBOX for rapid development and quick iteration from an idea to an initial functioning prototype, in an effortless manner.

5.3.5 Interactive Visualisations

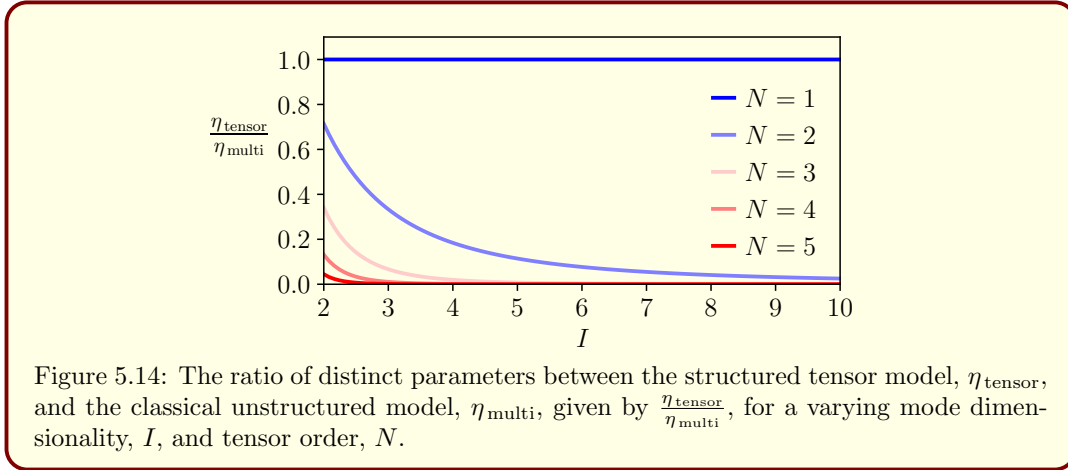
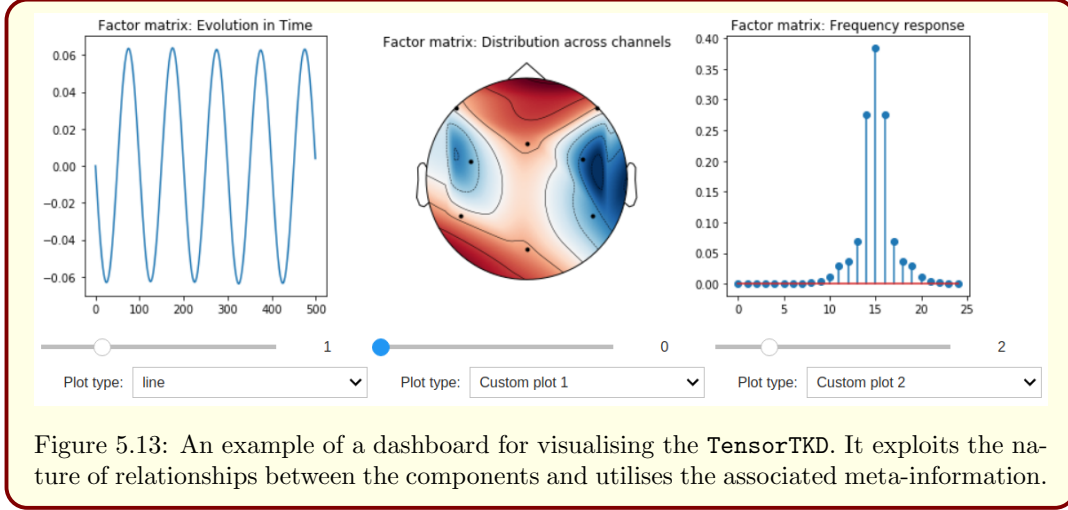
A key benefit of having a common API in our HOTTBOX library is that it provides us with the means to integrate the multi-linear structures with external popular libraries that serve completely orthogonal purposes, e.g. data wrangling and visualisation. The latter is particularly important for exploratory data analysis, and when displaying various forms of tensor representations. This is completely different from commonly used approaches, as current approaches require a considerable amount of repetitive work to achieve an acceptable and effective illustration result. To this end, in order to accommodate many versatile forms of efficient representations, within the HOTTBOX we provide users with an interactive dashboard. For example, Fig. 5.13 illustrates the user interface for a “raw” N-dimensional array factorised into the Tucker form. Due to the nature of this representation, different modes and components correspond to different properties that can be plotted using various types of graphs available in the dropdown menu. Although a set of options is predefined, it can be easily extended to suit a particular need of a user. In this way, a huge number of cross-modal relations intrinsic to the components of Tucker form (or other forms of efficient representations) can be explored with the use of sliders. Finally, meta-information is extracted and embedded as figure titles, to keep track of the design sequence.

Remark 23. *Fig. 5.13 merely displays synthetic data for illustrational purposes, where each panel visualises a single column vector from three different factor matrices that would have been obtained through either CPD or TKD of brain activity represented in the form of a third order tensor.*

5.3.6 Statistical Analysis Tools

HOTTBOX also provides classes and functions for the estimation of *Tensor-Valued Gaussian* statistical models [157], a functionality which does not have analogues in other fully featured libraries. The inherit notion of tensor-valued probability density function makes this module extremely suitable for a large variety of use-cases that aim to model interactions between real-world tensor data samples that are typically causal and probabilistic. Our implementation also benefits from the Kronecker separability of the first and second order moments of the tensor-valued random variable, as elaborated in [157]. In turn, this property of tensor-valued statistical models provides an immediate reduction in parameters required to represent a random process compared to the standard multivariate Gaussian model, as can be observed in Fig. 5.14.

5. Computational Efficiency and Practical Implementations



The so-enabled exploitation of tensor-valued models is essential for alleviating the *Curse of Dimensionality*, that is, an exponential growth in data volume with an increase in the order of the tensor. At the same time, implementation of tensor-valued Gaussian statistical models equips researches and data analysts with the essential tool for:

- Utilising Bayesian inference methods;
- Straightforward consideration of a mixture of probabilistic models;
- Statistical and hypothesis testing through the likelihood function;
- Employing class-conditional densities for classification tasks.

5.4 Chapter Summary

This chapter has covered the basics of graphical representation of tensors. We have revised the standard building blocks for Tensor Networks, together with showing how they are employed to graphically represent operations within Tensor Decompositions, such as the Tucker, and the Tensor Train. The material has been presented through the lens of graphical representations

5. Computational Efficiency and Practical Implementations

so that the results are highly accessible even to new researchers in the field. Our emphasis has been on providing better insight and intuition in some fundamentals of tensor algebra. In particular, it was shown how diagrams composed of edges and nodes can provide intuitive justifications to tensor unfoldings and how they offer greater flexibility in working with contraction products. This has led to a novel and computationally efficient method for tensor contraction, referred to as the Tensor Train Contraction Product (TTCP), that is easy to grasp. The benefits of the TTCP over the standard definition of contraction have been illustrated on Fig. 5.8. Next, we have introduced a recursive procedure for computing the higher-order singular value decomposition (R-HOSVD) of tensor-valued data streams in real-time. Numerical evaluations have demonstrated the efficacy of the proposed technique for online basis detection, even in the critical case of regime-changing environments as depicted on Fig. 5.10. The distinguishing advantage of the proposed framework is the extensive saving in computation, which arises from the relaxations employed on the factor matrix perturbation model, thereby opening new avenues for the next generation of streaming big data scenarios where computational affordability is of paramount importance. Finally, we have presented the Higher Order Tensor ToolBOX (HOTTBOX), a library dedicated to handling multi-dimensional data, which at its core employs a user-friendly approach to statistical analysis, visualisation, feature extraction, regression and non-linear classification paradigms. This makes it possible for HOTTBOX to be used by the multi-disciplinary communities which operate with inherently N-dimensional data arrays. Moreover, the toolbox is equipped with a unit tests suite in order to meet the high requirements on the integrity of the codebase of the modern software.

Chapter 6

Conclusions and Future Work

We have the duty of formulating, of summarizing, and of communicating our conclusions, in intelligible form, in recognition of the right of other free minds to utilize them in making their own decisions.

Ronald Fisher

6.1 Conclusions

This thesis has addressed the problem of joint analysis of multi-block data, whereby a particular focus was given to data preprocessing and the identification of common and individual information. The key motivator for this work has been the sheer capabilities and scalability of multi-dimensional representation of data through tensors, as only a limited portion of the computational methods has been readily available to researchers across a diverse range of data science fields. Drawing upon ideas evolved from the physical meaning of the fundamental multi-linear linear operations and inspired by the strong need for interpretability of tensor decompositions, we have been able to propose several algorithms for feature extraction from tensor valued data and enhancing computational performance of the associated tensor decompositions. In the process, several new results in the field of ensemble learning have been introduced which were accompanied by the practical implementations within publicly available software. This chapter presents a brief overview of the contributions of the thesis and outlines the directions for future work.

- **Chapter 1.** Introduction.

This first chapter serves as the general overview for approaches to the examination of the ever-increasing amount of real-world data and applications in signal processing and machine learning. The quest to make sense from information with such large volume and variety has highlighted both the limitations of standard flat-view matrix models and the necessity to move towards more versatile data analysis tools. This thesis was conceived with the aim of developing practical algorithms which build upon a solid foundation and physical meaning, while motivated by the need for new approaches and implementations within multi-linear analysis. The outline of every subsequent chapter is provided where details of the considered ideas and objectives are consequently elaborated.

- **Chapter 2.** Prerequisites: Multilinear Algebra.

Several important concepts of tensor analysis which are relevant to this thesis are thoroughly reviewed, such as folding and unfolding, mode- n , and contraction products. This

has allowed us to introduce the cornerstone tensor decompositions and corresponding implementation of the computation algorithms for the CPD-ALS, HOSVD, HOOI and TT-SVD as well as to seamlessly manipulate with the associated fundamental data representations in either Kruskal, Tucker or Tensor Train forms accompanied with the intuitive and self explanatory examples. Moreover, we also cover conditions of the efficient tensor representations that are sufficient to manifest uniqueness of the decomposition at hand. Finally, a notion of TN framework is introduced before continuing our journey into various tensorisation techniques. This allows us to appreciate tensor generation procedures by virtue of rearrangement of lower-dimensional structures, mathematical construction or experimental design that are indispensable tools for analysis and unravelling mysteries of latent components of multi-dimensional data.

- **Chapter 3.** Feature Extraction and Data Fusion.

A novel framework for feature extraction is introduced, which employs the CPD and LL1 decomposition with non-negativity constraint in order to separate common and individual subspaces. This was achieved through a realisation that the outer product is inherently capable of splitting the multi-modal relations, which paves the way for the elimination of the shared information from the multi-block data. Thus, machine learning algorithms can greatly benefit from operating over the much lower-dimensional and significantly more discriminative individual features. The choice of the CPD and LL1 decompositions for the proposed approach is justified by their intrinsic characteristic which allows us to establish a one-to-one connection between tensor components and facilitates intuitive interpretation. Unlike the matrix methods, the established framework is very flexible and is not restricted to input data of a specific shared structure. The problem of an effective approach to feature fusion when dealing with multi-dimensional data is also considered. Drawing inspiration from the fundamental methods and properties of the linear algebra, we introduce a mathematical formalism behind the summation of TNs with general properties of the equivalent matrix operations generalised to the case of multi-dimensional arrays of any arbitrary high order. This naturally led to a physically meaningful interpretation of the summation operation as a means for information blending that is possible only when TN representations of different information are considered as unified entities. Algorithms for feature extraction and data fusion presented in this chapter are supported by intuitive examples while practical advantages are demonstrated through an image classification application despite, the small size of the dataset and a high number of distinct classes.

- **Chapter 4.** Tensor Decompositions and Wisdom of the Crowd.

In this chapter, a novel framework that generalises classic flat-view ensemble learning to multi-dimensional tensor-valued data is introduced. This was achieved by virtue of tensor decompositions, whereby the proposed method, referred to as Tensor Ensemble Learning, decomposes every input data sample into multiple uncorrelated factors, feature matrices which in turn allows for flexibility in the choice of multiple learning algorithms in order to reduce the bias of base estimators and improve test performance. The TELVI algorithm takes full advantage of the intrinsic multi-way properties of the derived framework, thus, being highly parallelisable and suitable for large-scale problems. To validate the developed approach, several classification tasks were employed, where the most important results were obtained in the field of cardiovascular diagnosis. In particular, it is shown that TELVI exhibits substantial power in the blood flow analysis and carries a great promise at predicting AVF stenosis based on the feature extracted from B-mode ultrasound videos of patients' blood flow even when only a small number of data points per subject were available to the algorithm. These results are particularly encouraging for the clinical scenario of AVF surveillance, which remains a challenging area where the application of novel solutions is necessary to improve clinical outcomes.

- **Chapter 5.** Computational Efficiency and Practical Implementations.

6. Conclusions and Future Work

This chapter focuses on:

- Improving the performance and optimisation of the existing algorithms and operations in order to alleviate the computational burden associated with increasing amounts of data;
- Demystifying multi-dimensional decompositions for various data analysis communities active in the area by bridging the gap between the theory of the multi-linear paradigm and the direct implementation.

The standard multi-linear operations, tensor decompositions are revised through a lens of graphical representation in the form of TNs notation. Deeper insights and intuition were obtained through a combination of illustrative diagrams composed of nodes and edges, which then served as a basis for efficient calculation of the contraction product that employs the novel approach formalised in the TTCP algorithm. The chapter also provides material to strengthen one of the fundamental tensor decomposition, namely HOSVD, by overcoming a major stumbling block that requires the entire input data to be known beforehand. The considered method for recursive updates of the factor matrices completely eliminates such a restrictive requirement, while the efficacy of the proposed algorithm opens new avenues for the next generation of real-time streaming technology and applications in the current era of online processing. Finally, this chapter presented the HOTTBOX, an open source software for tensor decompositions, statistical analysis, visualisation, feature extraction, regression and non-linear classification of tensor-valued information. An abstraction layer placed between the user interface and the underlying computational algorithms for tensor decompositions reduced complex multi-way operations to a single functional call allowing for an enjoyable experience and quick iteration from an idea to a minimal valuable product in a matter of minutes. The ease of use of HOTTBOX was demonstrated through a series of illustrative examples. A particular emphasis is given to a unique, self-contained ecosystem of internal structures that establish a foundation for seamless integration with popular data science libraries utilised for visualisation and data manipulation. Thus, the HOTTBOX offers a simple and consistent platform to perform tensor signal processing and machine learning in an accessible and straightforward fashion, as the result, it lowers the entry barrier into the fascinating field of multi-dimensional arrays.

6.2 Future Work

While we have demonstrated the potential of using tensor decompositions and multilinear operation and have charted the way for machine learning algorithms to benefit from TDs, many opportunities are yet to be explored for extending the scope of this thesis. We shall briefly outline some pertinent ideas and survey several questions that allow for the generalisation of the concepts explored thus far.

- **Further practical applications of the CIFE framework.**

The theory for the extraction of common and individual features was developed in Chapter 3, and the technique was applied to image classification. A particularly interesting area for further applications would be processing physiological signals for person authentication systems. Following recent advances in sensor technology, it is now possible to capture EEG responses by utilising a portable in-ear device that does not require skilled assistance or cumbersome preparation for data collection [158]. At the same time, it is becoming a standard practice to protect sensitive or confidential information vulnerable to fraud with multi-factor authentication methods for an additional layer of security. While there are commonalities in EEG responses across people, biometric recognition techniques rest upon the unique characteristics of a person. It has been shown that the brain activity

satisfies all the requirements (universality, permanence, uniqueness, and collectability) to be utilised in real-world scenarios when data is obtained from a portable in-ear device [159]. Thus, an imposter attempting to gain unlawful access is faced with a security challenge (for example, picturing a dog riding a bike), whereby EEG signals generated as a result of mental activity would contain both common information related to the general knowledge about the verification task and individual features attributed to differences in perception and imagination. In such a situation, correct separation of common from individual characteristics plays a decisive role in whether or not access is granted.

- **Optimal selection of components for Tensor Ensemble Learning.** The research paradigm which bridges the gap between the classical ensemble learning and high dimensional data processing has been introduced in Chapter 4. Despite the strong notion of interpretability and intuition behind the proposed framework, combined with extreme flexibility of multi-linear analysis, it does not make the TEL an ultimate tool that is suitable for all ML problems associated with the multi-way data. Moreover, it is still unclear what is the optimal strategy that should be employed for creating surrogate datasets from the results of tensor decomposition of the raw data samples. Another interesting avenue to explore is whether the core tensor has any noticeable and meaningful effect during the training stage of base learner.

- **Further investigation and tensor methods into AVF.**

While work presented in Section 4.5 was the first attempt to employ multi-linear analysis for predicting AVF stenosis, there are a number of challenges for the employed workflow that require further development and evaluation. Currently, this technique can be applied to B-mode videos of blood flow following image acquisition. However, to maximise clinical value and for ease of use, real-time interpretation capability would be ideal. Furthermore, like vector flow imaging techniques, this technique also relies on frame by frame detection of speckle patterns. Therefore, further work is necessary to evaluate the optimal B-mode imaging settings for identifying speckle patterns scattered by red blood cells. An alternative strategy would be to use microbubble contrast agents which enhance ultrasound signals from blood [160] and have been used in a range of clinical applications [161]. This may improve the accuracy by enhancing the detection of speckles scattered from within blood by microbubbles. Moreover, apart from the considered application to AVF imaging, it may also prove useful in various cardiovascular ultrasound applications such as lower limb imaging and echocardiography.

- **Extension of the HOTTBOX functionality.**

Having established the solid foundation of core components within HOTTBOX in Section 5.3, it is important to keep extending the range of algorithms for multi-dimensional data available to end users. With the number of proposed multi-linear methods constantly growing, it takes a considerable period of time for them to become widely expected and readily accessible for experiments. Block Term Decomposition that borrows the ideas of linear separability and maximising multi-modal variance from CPD and TKD [74] is one of such example despite the expressiveness potential associated with the underlying algorithm. Tensor Ring Decomposition is another promising candidate, as it offers the circular dimensional permutation invariance which can be viewed as a linear combination of TT decompositions, thus providing the powerful and generalised representation abilities [162]. In addition, it is highly beneficial to incorporate methods for clustering, such as that proposed in [163], and the generalised multi-linear regression approach [131] suitable for inherently N-dimensional information. At the same time, software development is one of the most rapidly evolving practical fields where optimisations to low level operations are always work in progress. This opens a new avenue for implementing transparent support of hardware acceleration through utilising graphical process units (GPUs) and distributed computational engines in a divide-and-conquer manner, whereby an initial tensor data

6. Conclusions and Future Work

is split into a sequence of smaller blocks that are processed in parallel and consequently merged into a unified structure [164].

6. Conclusions and Future Work

References

- [1] A. Cichocki, N. Lee, I. Oseledets, A. H. Phan, Q. Zhao, and D. P. Mandic, “Tensor networks for dimensionality reduction and large-scale optimization. Part 1: Low-rank tensor decompositions,” *Foundations and Trends in Machine Learning*, vol. 9, no. 4-5, pp. 249–429, 2016.
- [2] H. Fanaee-T and J. Gama, “Tensor-based anomaly detection: An interdisciplinary survey,” *Knowledge-Based Systems*, vol. 98, pp. 130–147, 2016.
- [3] A. Cichocki, D. P. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, “Tensor decompositions for signal processing applications: From two-way to multiway component analysis,” *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.
- [4] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
- [5] R. Bellman, “Curse of dimensionality,” *Adaptive control processes: A guided tour*, vol. Princeton University Press, Princeton, NJ, 1961.
- [6] S.V. Dolgov and Savostyanov D.V., “Alternating minimal energy methods for linear systems in higher dimensions,” *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. A2248–A2271, 2014.
- [7] O. Debals, M. Van Barel, and L. De Lathauwer, “Blind signal separation of rational functions using Löwner-based tensorization,” *In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4145–4149, 2015.
- [8] M. A. Hasan and A. A. Hasan, “Hankel matrices of finite rank with applications to signal processing and polynomials,” *Journal of Mathematical Analysis and Applications*, vol. 208, no. 1, pp. 218–242, 1997.
- [9] A. Cichocki, A. H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, and D. P. Mandic, “Tensor networks for dimensionality reduction and large-scale optimization. Part 2: Applications and future perspectives,” *Foundations and Trends in Machine Learning*, vol. 9, no. 6, pp. 431–673, 2017.
- [10] F. Cong, Q. Lin, L. Kuang, X. Gong, P. Astikainen, and T. Ristaniemi, “Tensor decomposition of EEG signals: A brief review,” *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015.
- [11] F. Miwakeichi, E. Martinez-Montes, P. A. Valdés-Sosa, N. Nishiyama, H. Mizuhara, and Y. Yamaguchi, “Decomposing EEG data into space–time–frequency components using parallel factor analysis,” *NeuroImage*, vol. 22, no. 3, pp. 1035–1045, 2004.
- [12] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener, “Multiway analysis of epilepsy tensors,” *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

References

- [13] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.
- [14] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Studies in Applied Mathematics*, vol. 6, no. 1-4, pp. 164–189, 1927.
- [15] R. B. Cattell, "Parallel proportional profiles and other principles for determining the choice of factors by rotation," *Psychometrika*, vol. 9, no. 4, pp. 267–283, 1944.
- [16] J. D. Carroll and J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.
- [17] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory Multimodal Factor Analysis," *University of California at Los Angeles Los Angeles, CA*, 1970.
- [18] A. Smilde, R. Bro, and P. Geladi, "Multi-way analysis: Applications in the chemical sciences," *New York: John Wiley & Sons Ltd.*, 2004.
- [19] P. M. Kroonenberg, "Applied multiway data analysis," *John Wiley & Sons*, vol. 702, 2008.
- [20] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in measuring change*, vol. 15, pp. 122–137, 1963.
- [21] L. R. Tucker, "The extension of factor analysis to three-dimensional matrices," *Contributions to Mathematical Psychology*, pp. 110–127, 1964.
- [22] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
- [23] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank- $(r_1, r_2, \dots, r_n)$ approximation of higher-order tensors," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.
- [24] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
- [25] G. Beylkin and M. J. Mohlenkamp, "Algorithms for numerical analysis in high dimensions," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 2133–2159, 2005.
- [26] I. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.
- [27] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2377–2388, 2000.
- [28] B. Savas and L. Eldén, "Handwritten digit classification using higher order singular value decomposition," *Pattern Recognition*, vol. 40, no. 3, pp. 993–1003, 2007.
- [29] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear image analysis for facial recognition," *In Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, vol. 2, pp. 511–514, 2002.
- [30] T. Kolda and B. Bader, "The TOPHITS model for higher-order web link analysis," *In Proceedings of the International Workshop on Link Analysis, Counterterrorism and Security*, vol. 7, pp. 26–29, 2006.

- [31] D. Tao, X. Li, W. Hu, S. Maybank, and X. Wu, "Supervised tensor learning," *Knowledge and Information Systems*, vol. 13, no. 1, pp. 1–42, 2007.
- [32] P. Comon, "Tensor decompositions: State of the art and applications, keynote address in IMA Conf.," *Mathematics in Signal Processing, Warwick, UK*, 2000.
- [33] D. Chen, X. Li, L. Wang, S. U. Khan, J. Wang, K. Zeng, and C. Cai, "Fast and scalable multi-way analysis of massive neural data," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 707–719, 2014.
- [34] A. H. Andersen and W. S. Rayens, "Structure-seeking multilinear methods for the analysis of fMRI data," *NeuroImage*, vol. 22, no. 2, pp. 728–739, 2004.
- [35] K. P. Singh, A. Malik, S. Sinha, D. Mohan, and V. K. Singh, "Exploring groundwater hydrochemistry of alluvial aquifers using multi-way modeling," *Analytica Chimica Acta*, vol. 596, no. 1, pp. 171–182, 2007.
- [36] S. C. Lee, H. Liu, M. J. Kim, J. T. Kim, and C. K. Yoo, "Online monitoring and interpretation of periodic diurnal and seasonal variations of indoor air pollutants in a subway station using parallel factor analysis (PARAFAC)," *Energy and buildings*, vol. 68, pp. 87–98, 2014.
- [37] H. Kim, S. Lee, X. Ma, and C. Wang, "Higher-order PCA for anomaly detection in large-scale networks," *In Proceedings of the 2009 IEEE International Conference on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 85–88, 2009.
- [38] K. Maruhashi, F. Guo, and C. Faloutsos, "Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis," *In Proceedings of the 2011 IEEE International Conference on Advances in Social Networks Analysis and Mining*, pp. 203–210, 2011.
- [39] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, 2006.
- [40] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," *In Proceedings of the European Conference on Computer Vision*, pp. 447–460, 2002.
- [41] P. A. Chew, B. W. Bader, T. G. Kolda, and A. Abdelali, "Cross-language information retrieval using PARAFAC2," *In Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, pp. 143–152, 2007.
- [42] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 24–40, 2011.
- [43] B. W. Bader, T. G. Kolda, et al., "MATLAB Tensor Toolbox Version 3.1," Available online at <https://www.tensortoolbox.org>.
- [44] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0 – Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization," *In Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers*, pp. 1733–1738, 2016.
- [45] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in Python," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 925–930, 2019.

References

- [46] L. Hao, S. Liang, J. Ye, and Z. Xu, “TensorD: A tensor decomposition library in Tensor-Flow,” *Neurocomputing*, vol. 318, pp. 196–200, 2018.
- [47] A. Cichocki, “Era of big data processing: A new approach via tensor networks and tensor decompositions,” *arXiv preprint arXiv:1403.2048*, 2014.
- [48] G. G. Calvi, I. Kisil, A. Cichocki, Anh Huy Phan, and D. P. Mandic, “A tale of nodes and edges: Visualizing tensors and their basic operations,” *Submitted to IEEE Signal Processing Magazine*, 2020.
- [49] L. Stankovic, D. P. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides, “Understanding the basis of graph signal processing via an intuitive example-driven approach,” *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 133–145, 2019.
- [50] W. Hackbusch, “Tensor spaces and numerical tensor calculus,” *Springer Science & Business Media*, vol. 42, 2012.
- [51] B. N. Khoromskij, “Tensors-structured numerical methods in scientific computing: Survey on recent advances,” *Chemometrics and Intelligent Laboratory Systems*, vol. 110, no. 1, pp. 1–19, 2012.
- [52] L. Grasedyck, D. Kressner, and C. Tobler, “A literature survey of low-rank tensor approximation techniques,” *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.
- [53] M. Bouslé, O. Debals, and L. De Lathauwer, “A novel deterministic method for large-scale blind source separation,” *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, pp. 1890–1894, 2015.
- [54] R. M. Gray, “Toeplitz and circulant matrices: A review,” *Foundations and Trends in Communications and Information Theory*, 2006.
- [55] S. A. Billings, “Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains,” *John Wiley & Sons*, 2013.
- [56] D. Koutra, E. E. Papalexakis, and C. Faloutsos, “Tensorsplat: Spotting latent anomalies in time,” *In Proceedings of the 16th Panhellenic Conference on Informatics*, pp. 144–149, 2012.
- [57] J. B. Kruskal, “Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics,” *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.
- [58] N. D. Sidiropoulos and R. Bro, “On the uniqueness of multilinear decomposition of N-way arrays,” *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 3, pp. 229–239, 2000.
- [59] J. B. Kruskal, “Rank, decomposition, and uniqueness for 3-way and N-way arrays,” *Multway Data Analysis*, pp. 7–18, 1989.
- [60] I. Oseledets and E. Tyrtshnikov, “Breaking the curse of dimensionality, or how to use SVD in many dimensions,” *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3744–3759, 2009.
- [61] S. Holtz, T. Rohwedder, and R. Schneider, “The alternating linear scheme for tensor optimization in the tensor train format,” *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. 683–713, 2012.
- [62] J. A. Bengua, H. N. Phien, H. D. Tuan, and M. N. Do, “Efficient tensor completion for color image and video recovery: Low-rank tensor train,” *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2466–2479, 2017.

- [63] A. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *Nonlinear Theory and its Applications*, vol. 1, no. 1, pp. 37–68, 2010.
- [64] H. A. Phan, A. Cichocki, P. Tichavský, G. Luta, and A. Brockmeier, "Tensor completion through multiple Kronecker product decomposition," *In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3233–3237, 2013.
- [65] T. Varidddhisai and D. P. Mandic, "Online multilinear dictionary learning," *arXiv preprint arXiv:1703.02492*, 2017.
- [66] R. Zink, B. Hunyadi, S. Van Huffel, and M. De Vos, "Tensor-based classification of an auditory mobile BCI without a subject-specific calibration phase," *Journal of Neural Engineering*, vol. 13, no. 2, pp. 1–10, 2016.
- [67] Y. Guo and G. Pagnoni, "A unified framework for group independent component analysis for multi-subject fMRI data," *NeuroImage*, vol. 42, no. 3, pp. 1078–1093, 2008.
- [68] A. R. Groves, C. F. Beckmann, S. M. Smith, and M. W. Woolrich, "Linked independent component analysis for multimodal data fusion," *NeuroImage*, vol. 54, no. 3, pp. 2198–2217, 2011.
- [69] W. W. Chin, "The partial least squares approach to structural equation modeling," *Modern Methods for Business Research*, vol. 295, no. 2, pp. 295–336, 1998.
- [70] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3, pp. 321–377, 1936.
- [71] G. Zhou, A. Cichocki, Y. Zhang, and D. P. Mandic, "Group component analysis for multiblock data: Common and individual feature extraction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2426–2439, 2016.
- [72] G. Zhou, A. Cichocki, and D. P. Mandic, "Common components analysis via linked blind source separation," *In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2150–2154, 2015.
- [73] E. F. Lock, K. A. Hoadley, J. S. Marron, and A. B. Nobel, "Joint and individual variation explained (JIVE) for integrated analysis of multiple data types," *The Annals of Applied Statistics*, vol. 7, no. 1, pp. 523–550, 2013.
- [74] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms. Part II: Definitions and uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1033–1066, 2008.
- [75] M. Sørensen and L. De Lathauwer, "Canonical polyadic decompositions and coupled decompositions in multilinear rank- $(L_r, L_r, 1)$ terms. Part I: Uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 2, pp. 496–522, 2015.
- [76] M. Sørensen, I. Domanov, and L. De Lathauwer, "Canonical polyadic decompositions and coupled decompositions in multilinear rank- $(L_r, L_r, 1)$ terms. Part II: Algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 3, pp. 1015–1045, 2015.
- [77] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *In Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 556–562, 2001.
- [78] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," *In Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pp. 138–142, 1994.

References

- [79] O. Morris, M. Lee, and A. Constantinides, “A unified method for segmentation and edge detection using graph theory,” *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 11, pp. 2051–2054, 1986.
- [80] B. Leibe and B. Schiele, “Analyzing appearance and contour based methods for object categorization,” *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 409–415, 2003.
- [81] I. Kisil, G. G. Calvi, B. Scalzo Dees, A. Moniri, and D. P. Mandic, “HOTTBOX: Higher Order Tensors ToolBOX,” <https://github.com/hottbox>.
- [82] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [83] A. H. Phan and A. Cichocki, “Tensor decompositions for feature extraction and classification of high dimensional datasets,” *Nonlinear Theory and its Applications, IEICE*, vol. 1, no. 1, pp. 37–68, 2010.
- [84] T. G. Dietterich, “Ensemble methods in machine learning,” *In Proceedings of International Workshop on Multiple Classifier Systems*, pp. 1–15, 2000.
- [85] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.
- [86] D. Opitz and R. Maclin, “Popular ensemble methods: An empirical study,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.
- [87] S. Džeroski and B. Ženko, “Is combining classifiers with stacking better than selecting the best one?,” *Machine Learning*, vol. 54, no. 3, pp. 255–273, 2004.
- [88] Y. Koren, “The BellKor solution to the Netflix grand prize,” *Netflix Prize Documentation*, vol. 81, pp. 1–10, 2009.
- [89] R. E. Schapire, “The strength of weak learnability,” *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.
- [90] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [91] D. H. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
- [92] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [93] A. Liaw, M. Wiener, et al., “Classification and regression by randomForest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002.
- [94] W. K. Wong, Z. Lai, Y. Xu, J. Wen, and C. P. Ho, “Joint tensor feature analysis for visual object recognition,” *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2425–2436, 2015.
- [95] I. Kisil, G. G. Calvi, A. Cichocki, and D. P. Mandic, “Common and individual feature extraction using tensor decompositions: A remedy for the curse of dimensionality?,” *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6299–6303, 2018.

- [96] E. Acar, M. A. Rasmussen, F. Savorani, T. Næs, and R. Bro, "Understanding data fusion within the framework of coupled matrix and tensor factorizations," *Chemometrics and Intelligent Laboratory Systems*, vol. 129, pp. 53–63, 2013.
- [97] G. G. Calvi, I. Kisil, and D. P. Mandic, "Feature fusion via tensor network summation," *In Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, pp. 2623–2627, 2018.
- [98] V. Kumar and S. Minz, "Multi-view ensemble learning: An optimal feature set partitioning for high-dimensional data classification," *Knowledge and Information Systems*, vol. 49, no. 1, pp. 1–59, 2016.
- [99] Q. Zhang and S. Sun, "Multiple-view multiple-learner active learning," *Pattern Recognition*, vol. 43, no. 9, pp. 3113–3119, 2010.
- [100] S. Sun and Q. Zhang, "Multiple-view multiple-learner semi-supervised learning," *Neural Processing Letters*, vol. 34, no. 3, pp. 229, 2011.
- [101] L. Lam and S. Suen, "Application of majority voting to pattern recognition: An analysis of its behaviour and performance," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 27, no. 5, pp. 553–568, 1997.
- [102] H. A. Kiers and I. V. Mechelen, "Three-way component analysis: Principles and illustrative application," *Psychological Methods*, vol. 6, no. 1, pp. 84–110, 2001.
- [103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [104] I. Kisil, A. Moniri, G. G. Calvi, B. Scalzo Dees, and D. P. Mandic, "HOTTBOX: Higher Order Tensors ToolBOX," <https://github.com/hottbox>.
- [105] J. T. Daugirdas, T. A. Depner, J. Inrig, R. Mehrotra, M. V. Rocco, R. S. Suri, et al., "KDOQI clinical practice guideline for hemodialysis adequacy," *American Journal of Kidney Diseases*, vol. 66, no. 5, pp. 884–930, 2015.
- [106] H. Adequacy et al., "Clinical practice guidelines for hemodialysis adequacy, update 2006," *American Journal of Kidney Diseases*, vol. 48, no. 1, pp. S2–S90, 2006.
- [107] S. M. Sever and W. Meichelboeck, "European best practice guidelines on vascular access," *Journal of Vascular Access*, vol. 12, no. 2, pp. 137–181, 2011.
- [108] A. Besarab, "Resolved: Fistulas are preferred to grafts as initial vascular access for dialysis," *Journal of the American Society of Nephrology*, vol. 19, no. 9, pp. 1629–1633, 2008.
- [109] N. Gjorgjievski, P. Dzekova-Vidimliski, V. Gerasimovska, S. Pavleska-Kuzmanovska, J. Gjorgjevska, P. Dejanov, A. Sikole, and N. Ivanovski, "Primary failure of the arteriovenous fistula in patients with chronic kidney disease stage 4/5," *Macedonian Journal of Medical Sciences*, vol. 7, no. 11, pp. 1782, 2019.
- [110] H. I. Feldman, S. Kobrin, and A. Wasserstein, "Hemodialysis vascular access morbidity," *Journal of the American Society of Nephrology*, vol. 7, no. 4, pp. 523–535, 1996.
- [111] B. Manns, M. Tonelli, S. Yilmaz, H. Lee, K. Laupland, S. Klarenbach, V. Radkevich, and B. Murphy, "Establishment and maintenance of vascular access in incident hemodialysis patients: a prospective cost analysis," *Journal of the American Society of Nephrology*, vol. 16, no. 1, pp. 201–209, 2005.

References

- [112] W. Metcalfe, I.H. Khan, G.J. Prescott, K. Simpson, and A. M. MacLeod, “Hospitalization in the first year of renal replacement therapy for end-stage renal disease,” *QJM: An International Journal of Medicine*, vol. 96, no. 12, pp. 899–909, 2003.
- [113] P. Roy-Chaudhury, “Endothelial progenitor cells, neointimal hyperplasia, and hemodialysis vascular access dysfunction: Novel therapies for a recalcitrant clinical problem,” *Circulation*, vol. 12, pp. 3–5, 2005.
- [114] T. Lopez and F. Casino, “The management of arteriovenous fistula for dialysis patients,” *Italian Journal of Nephrology*, vol. 31, no. 1, 2014.
- [115] Guedes Marques, M. and Ibeas, J. and Botelho, C. and Maia, P. and Ponce, P., “Doppler ultrasound: A powerful tool for vascular access surveillance,” *Seminars in Dialysis*, vol. 28, no. 2, pp. 206–210, 2015.
- [116] C. S. Bavare, J. Bismuth, H. F. El-Sayed, T. T. Huynh, E. K. Peden, M. G. Davies, A. B. Lumsden, and J. J. Naoum, “Volume flow measurements in arteriovenous dialysis access in patients with and without steal syndrome,” *International Journal of Vascular Medicine*, vol. 2013, 2013.
- [117] A. Masengu, J. McDaid, A. P. Maxwell, and J. B. Hanko, “Preoperative radial artery volume flow is predictive of arteriovenous fistula outcomes,” *Journal of Vascular Surgery*, vol. 63, no. 2, pp. 429–435, 2016.
- [118] A. Asif, P. Roy-Chaudhury, and G. A. Beathard, “Early arteriovenous fistula failure: A logical proposal for when and how to intervene,” *Clinical Journal of the American Society of Nephrology*, vol. 1, no. 2, pp. 332–339, 2006.
- [119] I. Fiorina, M. V. Raciti, A. Goddi, V. Cantisani, C. Bortolotto, S. Chu, and F. Calliada, “Ultrasound vector flow imaging: Could be a new tool in evaluation of arteriovenous fistulas for hemodialysis?,” *Journal of Vascular Access*, vol. 18, no. 4, pp. 284–289, 2017.
- [120] Hansen, K. L. and Nielsen, M. B. and Jensen, J. A., “Vector velocity estimation of blood flow: A new application in medical ultrasound,” *Ultrasound*, vol. 25, no. 4, pp. 189–199, 2017.
- [121] Greff, K. and Srivastava, R. K. and Koutník, J. and Steunebrink, B. R. and Schmidhuber, J., “LSTM: A search space odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [122] W. McKinney et al., “Data structures for statistical computing in python,” *In Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, 2010.
- [123] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [124] “Plotly: An open-source, interactive graphing library for Python,” Available online at <https://github.com/plotly/plotly.py>.
- [125] “Dash: Python framework for building analytical web applications,” Available online at <https://github.com/plotly/dash>.
- [126] R. J. Trudeau, “Introduction to Graph Theory,” *Courier Corporation*, 2013.
- [127] Y. Zniyed, R. Boyer, A. De Almeida, and G. Favier, “A TT-based hierarchical framework for decomposing high-order tensors,” *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. A822–A848, 2020.
- [128] H. Johan, “Tensor rank is NP-complete,” *Journal of Algorithms*, vol. 4, no. 11, pp. 644–654, 1990.

- [129] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, “Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation,” *John Wiley & Sons*, 2009.
- [130] T. G. Kolda, “Orthogonal tensor decompositions,” *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 243–255, 2001.
- [131] Q. Zhao, C. F. Caiafa, D. P. Mandic, Z. C. Chao, Y. Nagasaka, N. Fujii, L. Zhang, and A. Cichocki, “Higher order partial least squares (HOPLS): A generalized multilinear regression method,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1660–1673, 2012.
- [132] P. Comon, “Tensors: A brief introduction,” *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 44–53, 2014.
- [133] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, Evangelos E. P., and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” *arXiv preprint arXiv:1607.01668*, 2016.
- [134] R. Bro, “PARAFAC. Tutorial and applications,” *Chemometrics and intelligent laboratory systems*, vol. 38, no. 2, pp. 149–171, 1997.
- [135] A. Smilde, R. Bro, and P. Geladi, “Multi-way analysis: Applications in the chemical sciences,” *Tecnometrics*, pp. 1–380, 2005.
- [136] P. M. Kroonenberg, “Applied multiway data analysis,” *John Wiley & Sons*, vol. 702, 2008.
- [137] P. Symeonidis, “ClustHOSVD: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD,” *IEEE Transactions on Systems, Man, and Cybernetics Systems*, vol. 46, no. 9, pp. 1240–1251, 2015.
- [138] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang, “Incremental tensor subspace learning and its applications to foreground segmentation and tracking,” *International Journal of Computer Vision*, vol. 91, no. 3, pp. 303–327, 2011.
- [139] X. Ma, D. Schonfeld, and A. Khokhar, “Dynamic updating and downdating matrix SVD and tensor HOSVD for adaptive indexing and retrieval of motion trajectories,” *In Proceedings of the 2009th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1129–1132, 2009.
- [140] D. Erdogmus, Y. N. Rao, H. Peddaneni, A. Hegde, and J. C. Principe, “Recursive principal components analysis using eigenvector matrix perturbation,” *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 13, pp. 2034–2041, 2004.
- [141] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, “MPCA: Multilinear principal component analysis of tensor objects,” *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 18–39, 2008.
- [142] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.
- [143] M. E. Tipping and C. M. Bishop, “Mixtures of probabilistic principal component analyzers,” *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.
- [144] C. F. Van Loan and G. H. Golub, “Matrix computations,” *Johns Hopkins University Press Baltimore*, 1983.

References

- [145] P. D. Hoff et al., “Separable covariance arrays via the Tucker product, with applications to multivariate relational data,” *Bayesian Analysis*, vol. 6, no. 2, pp. 179–196, 2011.
- [146] A. V. Knyazev and M. E. Argentati, “Principal angles between subspaces in an A-based scalar product: Algorithms and perturbation estimates,” *SIAM Journal on Scientific Computing*, vol. 23, no. 6, pp. 2008–2040, 2002.
- [147] L. Sorber, M. Van Barel, and L. De Lathauwer, “Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank- $(L_r, L_r, 1)$ terms, and a new generalization,” *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, 2013.
- [148] “Python implementation of the Tensor Train Decomposition,” Available online at <https://github.com/oseledets/ttpy>.
- [149] A. Novikov, P. Izmailov, V. Khrulkov, M. Figurnov, and I. Oseledets, “Tensor Train decomposition on TensorFlow (T3F),” *arXiv preprint arXiv:1801.01928*, 2018.
- [150] “Tensor network learning with PyTorch,” Available online at <https://github.com/rballester/tntorch>.
- [151] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22, 2011.
- [152] C. Battaglino, G. Ballard, and T. G. Kolda, “A practical randomized CP tensor decomposition,” *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 876–901, 2018.
- [153] B. Jeon, I. Jeon, L. Sael, and U. Kang, “Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries,” *In Proceedings of the 32nd IEEE International Conference on Data Engineering (ICDE)*, pp. 811–822, 2016.
- [154] H. A. L. Kiers, J. M. F. Ten Berge, and R. Bro, “PARAFAC2 Part I: A direct fitting algorithm for the PARAFAC2 model,” *Journal of Chemometrics*, vol. 13, no. 3-4, pp. 275–294, 1999.
- [155] I. Kisil, A. Moniri, and D. P. Mandic, “Tensor ensemble learning for multidimensional data,” *In Proceedings for the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1358–1362, 2018.
- [156] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: Experiences from the scikit-learn project,” *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.
- [157] B. Scalzo Dees and D. P. Mandic, “A statistically identifiable model for tensor-valued Gaussian random variables,” *arXiv:1911.02915*, 2019.
- [158] V. Goverdovsky, D. Looney, P. Kidmose, and D. P. Mandic, “In-ear EEG from viscoelastic generic earpieces: Robust and unobtrusive 24/7 monitoring,” *IEEE Sensors Journal*, vol. 16, no. 1, pp. 271–277, 2015.
- [159] T. Nakamura, V. Goverdovsky, and D. P. Mandic, “In-ear EEG biometrics for feasible and readily collectable real-world person authentication,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 648–661, 2017.

-
- [160] C. H. Leow, E. Bazigou, R. J. Eckersley, C. H. Alfred, P. D. Weinberg, and M. X. Tang, “Flow velocity mapping using contrast enhanced high-frame-rate plane wave ultrasound and image tracking: Methods and initial in vitro and in vivo evaluation,” *Ultrasound in Medicine & Biology*, vol. 41, no. 11, pp. 2913–2925, 2015.
- [161] J. R. Lindner, “Microbubbles in medical imaging: current applications and future directions,” *Nature Reviews Drug Discovery*, vol. 3, no. 6, pp. 527–533, 2004.
- [162] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, “Tensor ring decomposition,” *arXiv preprint arXiv:1606.05535*, 2016.
- [163] W. W. Sun and L. Li, “Dynamic tensor clustering,” *Journal of the American Statistical Association*, pp. 1–28, 2019.
- [164] S. K. Suter, M. Makhynia, and R. Pajarola, “Tamresh – Tensor approximation multiresolution hierarchy for interactive volume visualization,” *In Proceedings of Computer Graphics Forum*, vol. 32, no. 3, pp. 151–160, 2013.