Higher Order Tensor Decompositions for Machine Intelligence

Giuseppe Giovanni Calvi
Department of Electrical and Electronic Engineering
Imperial College London

A thesis submitted for the degree of
Doctor of Philosophy (Ph.D.)
December 2020

To Harrods's satay chicken salads.

Abstract

Recent years have witnessed a surge in the routine creation of large amounts of data. While traditional tools of linear algebra are still adequate for their analysis when the considered datasets are not particularly voluminous or complex, the unprecedented changes associated with *Big Data* have also highlighted the limitations of flat-view matrix models. This calls for the development of sophisticated tensor and multi-linear algebra algorithms that are naturally able to efficiently manipulate multi-dimensional data.

The overarching theme of this thesis is hence to develop novel, interpretable tensor algorithms and to show their practical utility in signal processing and machine learning applications. In the first part of the thesis, fundamental and advanced concepts commonly expressed with mathematical formulae are elucidated through graphical representations via Tensor Networks (TNs), which allow to visualize tensor equations as an interaction of nodes and edges. This has made it possible to provide new perspectives on core principles, as well as paving avenues for novel results, such as a method to perform tensor contractions through graphical representations of Tensor-Trains (TTs). Then, methods to ease the computational burden on the renowned Canonical Polyadic Decomposition (CPD) are presented. Firstly, it is shown how a prior TT-decomposition can reduce the CPD computational cost with respect to the tensor order from exponential to linear. Secondly, a lower bound on the tensor rank, R, is introduced along with conditions under which it is attained, allowing for an increased efficiency of the CPD computation.

In the second part of the thesis a framework for TN summation is introduced through an analogy with feature fusion. A practical application on the ETH-80 dataset has shown that the proposed framework preserves feature locality. Through a process of deflation, it is also demonstrated how TN summation can be employed for eigenvalue extraction of large scale matrices. Next, a method to allow the use of kernels in the Support Tensor Machine (STM), the tensor extension of the well-known SVM, is developed. This has made it possible to successfully apply it to a problem of financial forecasting.

In the third and final part of this thesis, tensors are applied to deep learning. The introduction of the Tucker Tensor Layer (TTL) as an alternative to the fully connected matrices in Neural Networks (NNs), has achieved a more than 60-fold compression on NNs applied to the classification of the MNIST and Fashion-MNIST datasets, at a small sacrifice in accuracy. Moreover, the novel analytical derivation of the tensor-valued back-propagation algorithm allows to gain insights into the training process, thus mitigating the notorious "black-box" issue inherent to NNs. Practical benefits of this result have been shown on the CIFAR-10 dataset. Finally, Tensor-Train Recurrent Neural Networks (TT-RNNs) have been applied to financial forecasting, further highlighting the compression and interpretability properties of tensors.

Contents

Abstract 3

Contents 4

List of Figures 7

List of Tables 11

Nomenclature 12

Statement of Originality 13

Copyright Declaration 15

Acknowledgements 16

Publications 17

1 Introduction 19
- 1.1 Perspective . . . 19
- 1.2 Historical Background . . . 20
- 1.3 Problem Statement . . . 24
- 1.4 Thesis Contribution and Organization . . . 25

2 Tensor Notations and Decompositions 30
- 2.1 Nomenclature and Notation . . . 30
- 2.2 Tensor Decompositions . . . 33
- 2.3 Tensor Networks . . . 41
- 2.4 Chapter Summary . . . 42

I Insights into Tensor Decompositions 44

3 A Tale of Nodes and Edges 45
- 3.1 Introduction . . . 45
- 3.2 Tensor Decompositions as Tensor Networks . . . 46
- 3.3 Tensor Operations via Tensor Networks . . . 46
- 3.4 Chapter Summary . . . 55

4 Easing the Computational Burden on the Canonical Polyadic Decomposition 56
- 4.1 Introduction . . . 56
- 4.2 Tensor Train – Canonical Polyadic Decomposition Duality . . . 57
- 4.3 Conversion from the TT Format to the CPD Format . . . 58

4.4 Solving the Permutation Ambiguity . . . 61
4.5 The Need for a Lower-Bound on the Tensor Rank . . . 62
4.6 Lower Bound Derivation . . . 63
4.7 Rank Detectability . . . 67
4.8 Example Application: EEG Data . . . 68
4.9 Chapter Summary . . . 69

II Tensors for Shallow Learning **71**

5 Tensor Network Summation **72**
5.1 Introduction . . . 72
5.2 Summation Framework . . . 73
5.3 Feature Fusion via TN Summation . . . 79
5.4 Eigenvalue Extraction via TN Summation . . . 82
5.5 Chapter Summary . . . 89

6 The Support Tensor Machine and its Applications to Financial Forecasting **90**
6.1 Introduction . . . 90
6.2 From the Support Vector Machine to the Support Tensor Machine . . . 91
6.3 Kernel Usage in STM . . . 94
6.4 Implementation . . . 96
6.5 Simulations and Discussions . . . 98
6.6 Chapter Summary . . . 103

III Tensors for Deep Learning **104**

7 Compression and Interpretability of Neural Networks via the Tucker Tensor Layer **105**
7.1 Introduction . . . 105
7.2 Preliminaries . . . 107
7.3 Tensor Derivatives . . . 109
7.4 The Tucker Tensor Layer and Tensor-Valued Back-Propagation . . . 111
7.5 Algorithm Implementation . . . 116
7.6 Simulation Results . . . 118
7.7 Chapter Summary . . . 126

8 Tensor-Train Recurrent Neural Networks for Financial Forecasting **127**
8.1 Introduction . . . 127
8.2 Tensor-Train Neural Networks . . . 128
8.3 Application to Finance and Discussions . . . 130
8.4 Experimental Results . . . 131
8.5 Chapter Summary . . . 136

9 Conclusions & Future Work **137**
9.1 Conclusion 137
9.2 Future Work 140

A Graph Tensorization **142**
A.1 Perspective 142
A.2 Theoretical Background 142
A.3 Graph-Tensor Duality 144
A.4 Graph Separability 147

References **152**

List of Figures

1.1 The 4V's of Big Data. Figure adapted from [1]. 20
1.2 Signal Processing and Machine Learning methods for Big Data. Notice the 4V's on the left. Figure adapted from [1]. 21
1.3 Types of data: From scalars to tensors. Figure adapted from [1]. 22
1.4 Thesis Structure. 28

2.1 Illustration of the mode-1, mode-2, and mode-3 unfoldings of a 3-rd order tensor 31
2.2 Illustration of the CPD for a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. 35
2.3 Illustration of the TKD for a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. 38
2.4 Illustration of the TT-decomposition for an N-th order tensor. 40
2.5 Basic building blocks for TNs. From left to right, top to bottom: 0-th order (scalar), 1-st order (vector), 2-nd order (matrix) and 3-rd order tensors. 41
2.6 Basics of TN operations. 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 $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ along the common mode $I_3 = J_1$; contraction between tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ along all modes; unfolding of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ along its first mode I_1. . . 42

3.1 Example of two relatively involved tensor operations, but easily comprehensible through a graphical representation. Image adapted from [1]. 46
3.2 Comparison between conventional (left) and graphical (right) representations of tensors decompositions. 47
3.3 Graphical representation of $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \mathbf{A}^{(3)T}$. By starting from the TKD of $\boldsymbol{\mathcal{X}}$ and exploiting the fact that $\mathbf{A}^{(n)T}\mathbf{A} = \mathbf{I}_{R_n}$, the equality becomes evident. 48
3.4 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. 49
3.5 Graphical justification of (2.28). By performing the 1-st mode unfolding graphically, it becomes clear that $\mathbf{X}_{(1)} = \mathbf{A}^{(1)}\mathbf{G}_{(1)}\mathbf{K}$, where $\mathbf{K} = (\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T$. 50
3.6 Simple example to show that different expressions for the same topology are possible. In this case $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ can be written as in (3.7), on the left-hand side in the figure, or as in (3.8), on the right-hand side. 51

3.7 Graphical illustration of TTCP. A contraction is performed on a 4th-order tensor $\boldsymbol{\mathcal{X}}$ and a 3rd-order tensor $\boldsymbol{\mathcal{Y}}$ along common modes $I_3 = J_2$. This is done by exploiting the topology of TTs. The TT decompositions of the permuted versions of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ are obtained, allowing to reduce the contraction to a single matrix multiplication, highlighted by the orange ellipsis and node $\mathbf{K}$. The output $\boldsymbol{\mathcal{Z}}$ is then obtained by contracting the newly formed TT at the bottom of the figure. . . 53
3.8 Comparison of computational cost (in seconds, log-scale) between (2.3) and TTCP for the $\boldsymbol{\mathcal{X}} \times_1^1 \boldsymbol{\mathcal{Y}}$ contraction. The 3-rd, 4-th and 5-th order cases are considered. Stemming from a graphical intuition, TTCP is more computationally efficient than the direct definition of contraction. 54
3.9 Graphical illustration of an "all-but-n" contraction. $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle_{-3} = \mathbf{X}_{(3)} \mathbf{Y}_{(3)}^T$ on $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $I_1 = J_1, I_2 = J_2$. 55

4.1 Graphical representation of (4.15) for a 4-th order tensor. The big grey nodes represent the CPD of $\boldsymbol{\mathcal{G}}^{(2)}$ and $\boldsymbol{\mathcal{G}}^{(3)}$, while $\boldsymbol{\mathcal{H}}$ is obtained as a contraction of $\mathbf{D}_2$ and the two $\boldsymbol{\mathcal{I}}_3$. This is represented as the grey rectangle in the lower part of the figure, where the green node is $\mathbf{D}_2$. Unlabelled modes have dimensionality R. . . 60
4.2 Computational cost for the CPD computed through different methods. Top: direct CPD on N-th order tensor. Middle: CPD on a tensor first compressed through the TKD. Bottom: CPD computed from the conversion of a tensor in the TT format to the CPD format. 63
4.3 Maximum detectable rank for N-th order tensors, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with $I_n = I$ for $n = 1, \ldots, N$, where $N = 1, \ldots, 11$, and increasing values of I. 68
4.4 Rank estimation of a real-world data tensor (EEG) through (i) subsequent CPD decompositions (blue curve), (ii) Theorem 1 (red curve). The red curve was obtained almost 24 times faster. 69

5.1 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. 75
5.2 Topology of a sum of tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format. 77
5.3 Graphical representation of the sum of tensors $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$ expressed in the TT/MPS format. 79
5.4 Graphical representation of commutativity and inverse operator properties. Top: commutativity. Notice how the order of arrangement of corresponding cores is preserved. Bottom: inverse operator (i.e. TN subtraction). Negation of is achieved simply by negating all the elements of any of its cores. 80
5.5 Sample images from the ETH-80 dataset. 81

5.6 An example of TN summation for the TKD representations of the images in the training set, where $\{\mathbf{A}'_m, \mathbf{B}'_m, \mathbf{C}'_m\} = \eta_m^{1/3}\{\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m\}$ and $\mathcal{G}'_m = \mathcal{G}_m/\eta_m$ *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. 81
5.7 TT-MPO representation of an 8-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_8}$. Notice that, differently from the TT-MPS, the first and last cores of the TT-MPO are 3-rd order tensors instead of matrices, while the middle cores are 4-th order tensors. 83
5.8 A TN composed from a combination of two TT-MPS and one TT-MPO to compute the largest (or smallest) eigenvalue of a large scale matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, and the corresponding eigenvector $\mathbf{x} \in \mathbb{R}^I$. Top: Overall representation. Bottom: graphical representations of (5.32) and (5.33) for sequential optimization. 86
5.9 Graphical representation of how to perform the outer product $\mathbf{x}_i\mathbf{x}_i^T$, where $\mathbf{x}_i = \text{vec}(\boldsymbol{\mathcal{X}}_i)$, by utilizing only the cores obtained from the TT representation $\boldsymbol{\mathcal{X}}_i = \langle\!\langle \boldsymbol{\mathcal{X}}_i^{(1)}, \ldots, \boldsymbol{\mathcal{X}}_i^{(N)} \rangle\!\rangle$. 87
5.10 Hotelling's deflation via TN summation at iteration i, where $i = 1, \ldots, K$. λ_i is the i-th computed eigenvalue and $\boldsymbol{\mathcal{X}}_i = \langle\!\langle \boldsymbol{\mathcal{X}}_i^{(1)}, \ldots, \boldsymbol{\mathcal{X}}_i^{(N)} \rangle\!\rangle$ are the cores associated to the corresponding eigenvector. The bottom part of the figure shows the starting point for the computation of eigenvector $\mathbf{x}_{i+1}$ represented in TT-MPS format as $\boldsymbol{\mathcal{X}}_{i+1} = \langle\!\langle \boldsymbol{\mathcal{X}}_{i+1}^{(1)}, \ldots, \boldsymbol{\mathcal{X}}_{i+1}^{(N)} \rangle\!\rangle$. 88

6.1 Graphical representation of the inner product $\langle \boldsymbol{\mathcal{X}}_*, \boldsymbol{\mathcal{W}} \rangle$, where $\boldsymbol{\mathcal{X}}_* \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\boldsymbol{\mathcal{W}} = \mathbf{w}_1 \circ \mathbf{w}_2 \circ \mathbf{w}_3$, and $\mathbf{w}_1 \in \mathbb{R}^{I_1}$, $\mathbf{w}_2 \in \mathbb{R}^{I_2}$, $\mathbf{w}_3 \in \mathbb{R}^{I_3}$. 94
6.2 One batch of tensor data obtained through a sliding window of 250 time samples, i.e. one financial year. The first (and second) slice include (lagged) percentage change in prices, r_t, while the third slice comprises daily percentage change in trading volume, v_t. 98
6.3 Cumulative profits generated using SVM and LS-STM based strategies, for varying values of C. 100
6.4 The time evolution of the A and B parameters of the sigmoid in (6.29), obtained by applying Platt-scaling on LS-STM. 101
6.5 Cumulative profits generated using LS-STM$_T$ for $C = 0.001$ and $T = 0.07$. 102

7.1 Verification of the gradients by confirming that the remainder tends to 0 faster than the magnitude of the perturbation, as per (7.48) and (7.50). Left: Gradient verification for $\frac{\partial L}{\partial \mathbf{A}^{(2)}}$. Centre: Gradient verification for $\frac{\partial L}{\partial \mathbf{A}^{(4)}}$. Right: Gradient verification for $\frac{\partial L}{\partial \boldsymbol{\mathcal{G}}}$. Notice the inverted abscissa and the logarithmic scale. 116
7.2 Graphical representation of (7.67). 119
7.3 Sample images from two synthetic datasets. *Left*: Synthetic Dataset 1. *Right*: Synthetic Dataset 2. 120

7.4 Gradients across epochs – TTL applied to various datasets. *Left column. Top row:* SD1; *bottom row:* SD2. The disparity in modal structural information is reflected in the magnitude of the normalized gradients. Middle column. Top row: MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{10\times10\times30}$. Bottom row: MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{5\times5\times10}$. No difference in modal structural information is observed. Right column. Top row: Fashion-MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{10\times10\times30}$; tbottom row: Fashion-MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{5\times5\times10}$. Results suggest that the second mode carries more structural information. 121
7.5 Sample images from the MNIST dataset. 122
7.6 Sample images from the Fashion-MNIST dataset. 122
7.7 Sample images from the CIFAR-10 dataset. 123
7.8 Gradients across epochs – TTL applied to CIFAR-10. A TTL with a core tensor $\mathcal{G} \in \mathbb{R}^{10\times10\times3\times10}$ is employed, to unveil more structural information within mode-3 than within the other modes, thus indicating that the colour scheme of the images within CIFAR-10 plays a major role in classification tasks. 124
7.9 Modal-VGG architecture. Modal features are passed to the network in vector format as expressed in (7.69), while the simplified VGG-16 is the normal VGG-16 without the last two convolutional layers. 125

8.1 Example of bijection of the indices of a general vector $\mathbf{x} \in \mathbb{R}^M$ to a 3-dimensional vector index (J_1, J_2, J_3) of a 3-rd order tensor $\mathcal{X} \in \mathbb{R}^{J_1\times J_2\times J_3}$. 129
8.2 Modal connections between every mode of the input data tensor, $\mathcal{X}_t$, and the TT-cores, $\mathcal{G}_n$, of the tensorized weight matrix $\mathbf{W}^{xh}$ in TT-format. 133
8.3 Training curves of the simple RNN model compared to the TT-RNN. It is clear that the TT-RNN presents less overfitting, and can hence be employed as a regularization tool. 134
8.4 Cumulative profits generated from the long-only strategy, shown in black, the TT-RNN predictions, shown in red and the simple RNN predictions, shown in purple. 135
8.5 Normalized change of core weights of TT-RNN. 136

A.1 Cartesian product of two graphs. Figure adapted from [2]. 144
A.2 Tensorization of discrete samples in $\mathbb{R}^3$. 145
A.3 Cartesian product of 3 disjoint path graphs. 146
A.4 Tensorizing data, $\mathbf{x}_\mathcal{G} \in \mathbb{R}^P$, originally on graph $\mathcal{G}$, through the best path graph approximation. 147

List of Tables

2.1 Basic Tensor and Matrix Notation 31
2.2 Definition of Main Products 32

5.1 Summation 79
5.2 Scalar multiplication 79
5.3 Classification rates for the ETH-80 dataset, in %. 83

6.1 Results for pre- and post-Platt scaling for SVM and LS-STM, for varying values of C. The best results are highlighted in bold. The subscript $(\cdot)_P$ indicates post-Platt scaling. 99

7.1 Inputs and outputs of network layers. 117
7.2 TTL performance for various Compression Factors (CF). 123
7.3 Modal-VGG performance. 126

8.1 Multi-way financial data selected from Bloomberg. 132
8.2 Recurrent neural network architecture 133

Nomenclature

ACRONYMS

ALS	Alternating Least Squares
CANDECOMP	CANonical DECOMPposition
CPD	Canonical Polyadic Decomposition
HOSVD	Higher Order Singular Value Decomposition
MPO	Matrix Product Operator
MPS	Matrix Product State
NN	Neural Network
PARAFAC	PARAllel FACtor
STM	Support Tensor Machine
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TD	Tensor Decomposition
TKD	Tucker Decomposition
TN	Tensor Network
TT-RNN	Tensor Train Recurrent Neural Network
TTCP	Tensor Train Contraction Product
TTL	Tucker Tensor Layer
TT	Tensor Train

SYMBOLS

$\mathbf{1}_N$	$N \times 1$ column vector of ones
$\mathbb{R}$	Field of real numbers
$\mathbf{X}$	Matrix
$\mathbf{x}$	Vector
$\mathbf{X}_{(n)}$	Mode-n unfolding of tensor $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{X}}$	Tensor

$\underline{\mathbf{X}}$	Tensor network
x	Scalar
$\mathbf{I}_N$	$N \times N$ Identity matrix

OPERATORS

$(\cdot)^T$	Vector/matrix transposition
$\bar{\times}_n$	Mode-n tensor-vector contraction
$\circ$	Outer product
$\circledast$	Khatri-Rao Product
$\langle \cdot, \cdot \rangle$	Inner product
$[\![\cdot \; ; \; \cdot]\!]$	Full multilinear product
$(\!\| \cdot \; ; \; \cdot |\!)$	Full tensor train product
$\mathcal{P}(\cdot)$	Permutation operator
$\odot$	Hadamard product
$\oslash$	Hadamard division
$\otimes$	Kronecker product
$\mathcal{I}_N$	N-th order tensor with ones on its diagonal
Diag$(\cdot)$	Returns a vector of the diagonal elements of a square matrix
diag$(\cdot)$	Creates a diagonal matrix of its arguments
vec$(\cdot)$	Vectorization operator
$\times_n^m$	(m, n)-contraction
$\times_n$	Mode-n contraction

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.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Danilo Mandic who gave me the opportunity to undertake this PhD and for his support and guidance. During the last four years, through his encouragement and friendship, he has given me countless opportunities to develop as an academic and a person. There is still a lot left to learn from him but I hope that I was able to extract as much knowledge as possible while being his student.

I am also grateful to Vladimir Lucic, whose input and insights have laid the basis for some central results of this thesis.

Next, I am thankful to Ahmad Moniri, Bruno Scalzo, David Joseph, and Ilya Kisil, for the productive conversations, the lively and heated arguments, the daily banter, for winning competitions together, the coloured jokes, for working together, fixing my computer (Ilya), and for always, always having each other's backs even when times were not at their brightest. We sure tried to make them interesting.

I also would like to thank my friend and flatmate Andrea Cassinelli, for his constant positivity and healthy influence.

Nevertheless, I am grateful to former colleagues Sithan Kanna and Enrico Piovano for showing me the ropes in the early days. It is important to also thank my other fellow CSP friends, Tiffany, Metin, Chris, Mahmoud, for the shared experiences and memorable times.

Last but not least, my heartfelt gratitude goes to my mother, my father, my brother, and my sister. From Italy, to the UK, to the US, I was always provided with unconditional love, support, and guidance. I would not be here without you.

Pepe Calvi
December 2020

Publications

Published Peer-Reviewed Journals

[J1] G. G. Calvi, B. Scalzo Dees and D. P. Mandic "A lower bound on the tensor rank based on its maximally square matrix unfolding", *Elsevier Signal Processing*, vol. 180, Art. no. 107862, 2020.

[J2] A. H. Phan, A. Cichocki, I. Oseledets, G. G. Calvi, S. A. Asl and D. P. Mandic, "Tensor networks for latent variable analysis: Higher order canonical polyadic decomposition", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 2174-2188, 2019.

Submitted Journals

[J3] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao and D. P. Mandic "Compression and interpretability of neural networks via Tucker tensor layer: From first principles to tensor-valued back-propagation", *Submitted to Neural Computation*, 2020.

Journals in Preparation

[J4] G. G. Calvi, I. Kisil, Y. L. Xu, A. Cichocki, and D. P. Mandic, "A tale of nodes and edges: From basic tensor operations to advanced concepts", *In Preparation for IEEE Signal Processing Magazine*.

Published Peer-Reviewed Conference Proceedings

[C1] G. G. Calvi, V. Lucic, and D. P. Mandic, "Support tensor machine for financial forecasting", *In Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8152-8156, 2019.

[C2] 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.

[C3] 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.

Submitted to Conference Proceedings

[C4] Y. L. Xu, G. G. Calvi, and D. P. Mandic, "Tensor-train recurrent neural networks for financial forecasting", *Submitted to the Proceedings of the 46th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Published Book Chapters

[B1] I. Kisil, G. G. Calvi, B. Scalzo Dees, and D. P. Mandic, "Tensor decompositions and hands-on applications", *Studies in Computational Intelligence*, Springer, pp. TBA, 2020.

[B2] A. Konstantinides, B. Scalzo Dees, G. G Calvi and D. P. Mandic, "Text mining – A key lynchpin in the investment process: A survey", *Application of Intelligent Systems*, pp. 181–193, 2018.

Chapter 1

Introduction

> You can have data without information but you cannot have information without data.
>
> *Daniel Keys Moran*

1.1 Perspective

The widespread use of sensor technology and the emergence of ever increasing complex and voluminous datasets are becoming exceedingly common to the extent which renders standard analysis methods and algorithms inadequate [1, 3]. Such datasets fall under the umbrella of *Big Data*, that is, datasets consisting of multidimensional, multi-modal information characterized by one or more of the so-called "4V's" of Big Data outlined in Figure 1.1, each of which represents a research challenge in its own right [4, 5, 6]:

- **Volume**: The amount of information produced. Recent years have seen unrivalled proportions in a surge of generated data. For example, there are more than 2.2 billion active users on social media, each producing information by clicking on ads, writing comments, updating profiles and other activities. This calls for the need of scalable algorithms able to cope with the sheer size of datasets and their associated required storage memory;
- **Variety**: The multi-modality inherent to such datasets. This demands for reliable methods to fuse different data types, e.g. continuous, discrete, binary, time-series, text, video, multi-view, or probabilistic;
- **Velocity**: The rate of information flow. This creates the need for algorithms capable of processing large data-streams in near or in real-time;
- **Veracity**: The uncertainty in data. This calls for robust and predictive algorithms to process noisy, incomplete, and inconsistent data.

Owing to the increasingly affordable recording devices, extreme scale volumes and a large variety of data are appearing across science and engineering disciplines. Such massive datasets may have billions of entries and are typically represented in the form of very large block matrices, and tensors. This has spurred a renewed interest in the development of algorithms that are suitable for huge datasets, with a particular attention given to tensors and their ability to efficiently manipulate multidimensional data. An overview of potential applications of tensors is given in diagrammatical format in Figure 1.2.

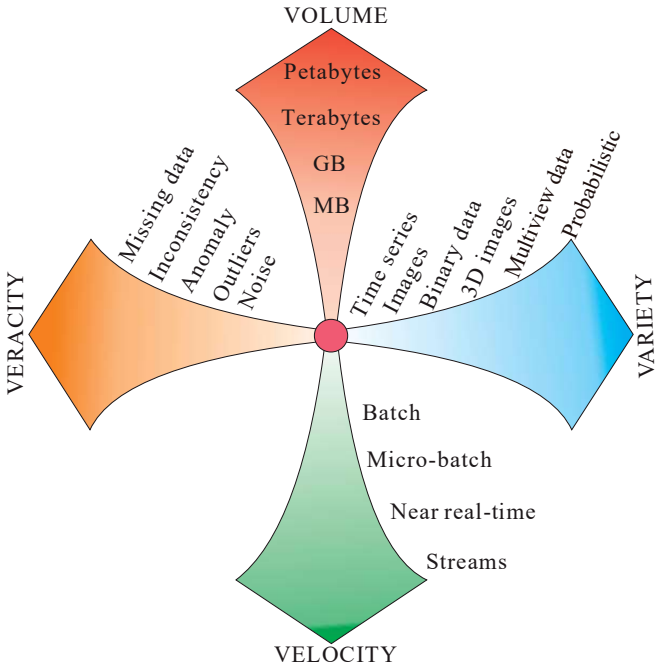


Figure 1.1: The 4V's of Big Data. Figure adapted from [1].

1.2 Historical Background

A tensor is a geometric object used in mathematics and physics to extend concepts such as scalars, vectors, and matrices to higher dimensions (an "N-way" or "N-th order" tensor indicates the number of dimensions present - as shown in Figure 1.3). The origin of the word "tensor" comes from the Latin term *tendere*, i.e. "to stretch", which first appeared in the 17th century in anatomy to denote muscle's stretch [7]. Tensor calculus, which is closer to the word's current meaning, was introduced in the early 20th century by Italian mathematicians Gregorio Ricci-Curbastro and Tullio Levi-Civita [8]. In 1915, the term "tensor" was used by Albert Einstein in general relativity for mathematical tools which were used to describe the geometry of space-time [9]. Moving towards signal processing and data analysis applications, the first principles of Tensor Decompositions (TDs) were introduced by American mathematician Frank Hitchcock in 1927 [10]. These faced a major surge in interest from the 1940s to the 1970s, thanks to psychologists such as Raymond Cattell [11, 12], Ledyard Tucker [13, 14, 15], and Richard Harshman [16] who were pioneers in extending tensor decompositions applications within their field. In particular, Tucker is known for the renowned Tucker Decomposition (TKD), and Harshman for the PARAllel FACtor model (PARAFAC), which was independently rediscovered by Carroll and Chang and put into an application within Psychometrics under the name of CANonical DECOMPosition (CANDECOMP) [17]. Nowadays, it takes the name of Canonical Polyadic Decomposition (CPD) [18].

Following their success within the Chemometrics community, thanks to the works of Appellof and Davidson in 1981 [19], tensors saw a considerable interest within the signal processing realm in the early 1990s, when it was realized that, for multivariate cases, Higher-Order

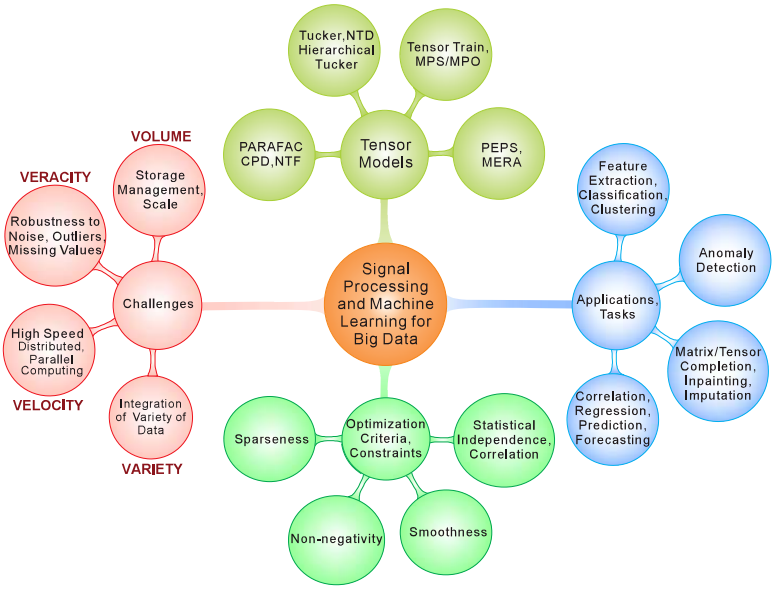


Figure 1.2: Signal Processing and Machine Learning methods for Big Data. Notice the 4V's on the left. Figure adapted from [1].

Statistics (HOS) are effectively higher-order tensors [20, 21, 22, 23]. Around 2000, De Lathauwer showed that the TKD can represent a higher-order generalization of the Singular Value Decomposition (SVD), now called Higher Order SVD (HOSVD) [24] . The HOSVD spurred interest in tensors in applied mathematics and computing in very high dimensions [18, 25, 26, 27]. Since the 1990s, tensors have been shown successful in a myriad of signal processing applications, some of which include:

- **Process control**: Tensor multi-way methods can be seen in the monitoring of batch processes, with the goal of understanding the major sources of batch-to-batch variations [28], real-time detection of faulty batches, and using them to improve operation policies. The multi-way data follows the pattern of *Batch*$\times$ *Measurement*$\times$ *Time*;

- **Environmental monitoring**: It is now feasible to analyze several ecological parameters through different locations and times. Due to spatiotemporal variations in such data, the need of tensor analysis naturally emerged within this domain. Tensors have been applied in water quality monitoring [29, 30, 31], control of air pollution [32, 33], and monitoring of soil quality [34, 35]. The related data generally follows the scheme of *Variables*$\times$*Sampling Site*$\times$*Sampling Time*;

- **Video surveillance**: Owing to their multi-way natures, tensors are natural data models for video data, and can, for example, provide a more accurate framework for abnormal activity discovery for security reasons [36, 37]. Video data can be represented as a 4D tensor spanning *RGB Colour*$\times$ *Image Row*$\times$ *Image Column*$\times$*Time*;

- **Neuroscience**: Brain data is inherently spatiotemporal, as every occurring activity within the brain is managed via different regions during a specific period of time. The two most known tools for capturing the brain's activities in a machine-readable format are the

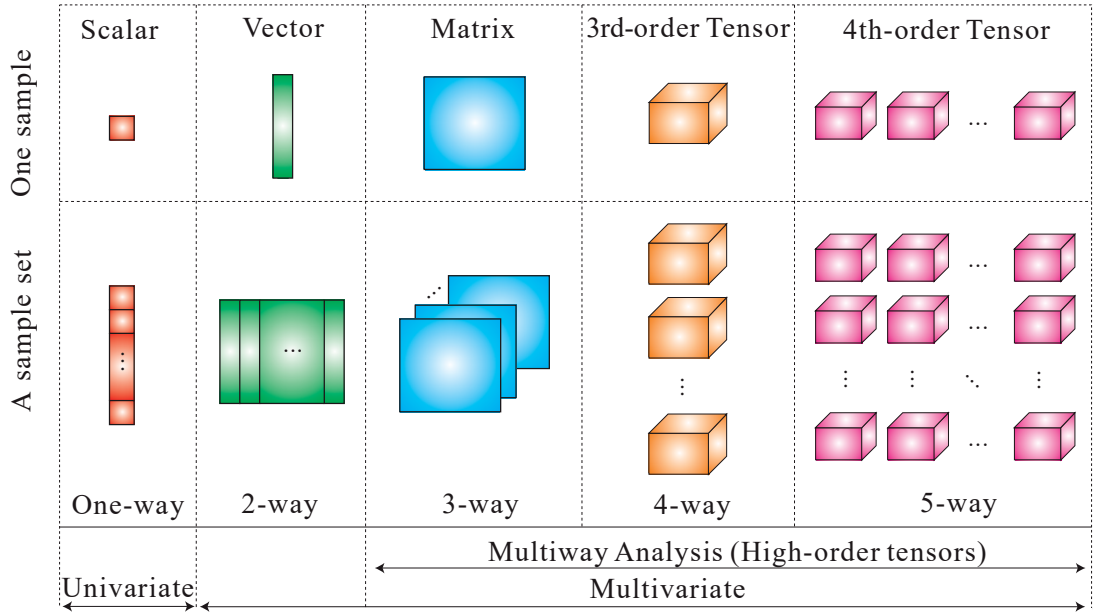


Figure 1.3: Types of data: From scalars to tensors. Figure adapted from [1].

Electroencephalography (EEG) and the Functional Magnetic Resonance Imaging (fMRI). The data extracted in such a way can be analyzed via tensor models to detect abnormal activities or specific patterns within the brain [38, 39]. The most common tensor model for EEG data is *Frequency*$\times$ *Channel* $\times$ *Time*;

- **Medical applications**: Tensors are also used for analysis of electronic medical records. For example, in [40] Tensor Decompositions were applied to Electronic Health Records (EHR) for prediction of heart failure, having the data arranged as *Medication* $\times$ *Patient* $\times$ *Diagnosis*.

Other applications of tensors include, but are not limited to, modelling spectral changes of substrates and products (*Wavenumber*$\times$ *Time* $\times$ *Activity*) [41], tracking the analysis of proteins (*Contact Matrix* $\times$ *Time*) [42], and seismology (*Space* $\times$ *Time* $\times$ *Frequency*) [43].

1.2.1 Curse of Dimensionality

Tensors are closely intertwined with the so-called *curse of dimensionality*, term which was first coined in [44], to indicate that the number of samples needed to estimate an arbitrary function grows exponentially with the number of parameters of the function itself. Within the realm of signal processing, the Curse of Dimensionality may refer to the exponentially increasing number of parameters required to describe a system. In particular, in the context of tensors, the Curse of Dimensionality is associated with the phenomenon whereby the number of elements I^N, of an N-th order tensor of equal dimensionality along all of its modes, that is a tensor of size $I \times I \times \cdots \times I$, grows exponentially with the tensor order, N. This implies that the tensor volume, i.e. the number of its elements, can become prohibitively big for multi-way arrays for which the number of dimensions is relatively high.

A tensor is said to be in *full format* when it is represented as an original, raw, multidimensional array [45]. The storage and processing of higher-order tensors in full format is often infeasible due to the Curse of Dimensionality. This, however, can be alleviated, or completely dealt with, through low-rank tensor approximations obtained via those known as *Tensor Decompositions* (TDs) (presented in detail in Section 2.2), which in turn lead to more complex and expressive *Tensor Networks* (TNs) formats. As in standard linear algebra matrix decompositions can be effectively used to compress matrices by obtaining low-rank representations, the concept of compression of multidimensional data through TDs can be intuitively explained as follows [46]. Consider an N-variate function, f, defined as,

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

Supposing f has to be represented by a product of individual functions, in the simplest case it can be approximated by

$$f(x_1, x_2, \ldots, x_N) \approx f^{(1)}(x_1) f^{(2)}(x_2) \cdots f^{(N)}(x_N) \tag{1.2}$$

Let the size of each mode $n = 1, \ldots, N$ be denoted by I_n, and define $I = \max_n\{I_n\}$. Then the memory requirement to store this approximation of the full tensor is $\sum_{n=1}^{N} I_n \ll I^N$. Unfortunately, the case in which all modes of a tensor are completely separable as in (1.2) is rare in practice. Instead, the concept of TDs is considerably more flexible, as it relies on the partial separability of the variables of a higher-order tensor [1].

It has been shown that TDs naturally cater for the excessive volume, veracity, and variety of data, and they are supported by algorithms involving relatively simple mathematical operations. Moreover, they "break down" a full tensor into smaller, lower-order, and more manageable *core tensors*. These are relatively easy to handle, enabling super-compression of the raw full tensor formats. The core tensors may also be distributed among different machines, further tackling storage issues.

Remark 1.1. It is important to note that, in general, a tensor cannot be compressed or decomposed without incurring compression errors, since any lower-rank representation is merely an approximation of the original full format tensor.

1.2.2 Blessing of Dimensionality

The higher dimensionality of tensors does not necessarily always come as a curse. Indeed, it can represent an opportunity to exploit the multi-way structure encoded in the data within. Hence, a curse turns into a blessing. While a significant amount of data is originally in vector and matrix format, as grey-scale images for example, tensors are often constructed solely for the mathematical benefits they provide in data analysis. For instance, it seemed natural to stack together excitation-emission spectroscopy matrices in chemometrics into a 3-rd order tensor [18, 47].

The procedure of formulating a data tensor from lower-order original data is referred to as *tensorization*, the techniques of which are summarized in the following:

1. **Rearrangement of lower-dimensional data structures**: Large scale matrices or vectors can be readily *tensorized*, i.e. reshaped into a tensor format, so that they can be compressed through tensor decompositions. This principle has been shown to facilitate data analysis [48, 49, 50]. More details are provided in Section 2.1.2.

2. **Mathematical construction**: Certain vector-valued and matrix-valued entities can be converted into tensors by exploiting the structure of the data, via mathematical operations [18]. For example, snapshots of covariance matrices of time-series, obtained via a sliding window, can be stacked to form a tensor. Also, a *Channel* $\times$ *Time* data matrix can be transformed into a *Channel* $\times$ *Time* $\times$ *Frequency* 3-rd order tensor via time frequency representations, which has shown to be a useful procedure in multi-channel Electroencephalogram (EEG) analysis [51, 52].

3. **Experiment design**: Multi-signal data can be naturally stacked into a tensor. For example, in wireless communications, the so-called signal diversity, that is, temporal, spatial, spectral, and so on, can form a tensor if stacked together [53]. Another example could be the arrangement of medical data: if an experiment involves collection of signals from different channels, for different trials, for different patients, then the data can be organized in a *Time* $\times$ *Channel* $\times$ *Trial* $\times$ *Patient* 4-th order tensor [54].

4. **Natural tensor data**: Some data sources are readily available in tensor format. For example, RGB images are inherently 3-rd order tensors [18], while sequences of images in a coloured video become a 4-th order tensor.

Remark 1.2. Although data in tensor format can be preferable due to the flexibility of multilinear algebra and the decompositions inherent to tensors, a tensor only "makes sense" if its structure can be physically justified. In other words, tensorization is useful only when there are grounds for it and is performed as a consequence of a prior hypothesis.

1.3 Problem Statement

Tensor operations and factorizations are not merely a form of matrix algebra with additional subscripts. For example, even basic notions such as the rank of a tensor have a more subtle meaning, and uniqueness conditions for tensor decompositions are generally more relaxed than those for their matrix counterparts [55, 56]. More generally, this implies that, although a large portion of tensor algorithms derive from matrix and linear algebra tools, tensors offer several avenues for both theoretical and practical research, that go beyond mitigating the curse of dimensionality.

An aspect that need not to be underestimated is how to visualize tensor operations, for better interpretability of results. The most common approach to tackle this issue is to employ those known as Tensor Networks (TNs), which diagrammatically represent tensor operations as a series of interconnected nodes and edges to elegantly overcome the need of cumbersome notation, at no expense in rigour [3]. These are applied extensively within the signal processing community, see, for example, [1, 27, 57, 58, 59, 60] to name but a few. Despite their extensive usage, however, their full potential is yet to be uncovered, given the vast scope of tensors and

their applications. Where standard tensor algebra stumbles upon a roadblock, novel insights into the mechanics of how tensor decompositions operate can be obtained through a TN approach, potentially unveiling new algorithms for original applications.

Indeed, it is well accepted that, although tensor decompositions first appeared in 1927 [10], they remain a thriving and active area of research [61, 62, 63, 64, 65, 66]. Some draw inspiration from their matrix counterparts, such as the Canonical Polyadic Decomposition (CPD) or the Higher Order Singular Value Decomposition (HOSVD) [24], while others are specific to tensors, such as the Tensor-Train (TT) [57] decomposition. In either case, each tensor decomposition presents properties yet requiring full explanation, including uniqueness and convergence conditions. Moreover, significant efforts are devoted towards improving their efficiency in terms of computational or storage requirements [67, 68, 69], and towards the development of novel decompositions to overcome potential shortcomings of previously existing ones [70].

Consequently, insights into tensor decompositions and operations open the path for novel and practically significant applications. In addition to the possible applications presented in Section 1.2, tensors may also be applied to image processing [71, 72, 73], time-series analysis [74, 75], and, more recently, to neural networks [57, 76, 77, 78, 79]. The general aim of these applications can always be summarized in three points: (i) use tensors to reveal and exploit the underlying data structure; (ii) leverage on the super-compression abilities of tensors and their decompositions to obtain more efficient models in terms of storage requirements or computational costs; (iii) employ the flexibility of multi-linear algebra to improve model expressiveness and increase interpretability.

The research undertaken for this thesis is therefore motivated by: (i) the need to provide further insights into tensor decompositions; (ii) the opportunities for novel tensor applications within the fields of image processing, time-series analysis, and deep learning; and (iii) the necessity to improve interpretability and mitigate the "black-box" nature of existing signal processing and machine learning models. The main research objective can therefore be summarised as *"to develop novel, interpretable tensor algorithms and to show their practical utility in signal processing and machine learning applications"*. This thesis aims to achieve the goal from various angles which include giving new perspectives and insights to some well-established tensor methods and decompositions, formulating new algorithms and uncovering their learning behaviour in the case of neural networks, and testing algorithms on real-world data.

1.4 Thesis Contribution and Organization

The contributions of this thesis can be divided into four areas, which roughly correspond to how the thesis is organized.

1. **A New Perspective to the Representation of Tensor Operations**
   The representation of TNs to effectively describe tensor operations in a synthetic and intuitive way is a widely applied concept within the research community [1, 27, 57, 58]. However, they are often applied to describe involved mathematical expressions, while those involving tensor basics are most of the times expressed purely algebraically. These still may not be trivial, and a new researcher in the field may feel that they require further

justification. **Chapter 3** tackles this issue by showing how any tensor decomposition can be interpreted as a TN, which demystifies tensor fundamentals by providing immediate intuition behind their mechanics. It is also shown, how a graphical representation of tensors, which is a common theme among most chapters of this thesis, is a powerful concept promoting novel insights and new findings. These, among others, include the possibility to combine various TNs by introducing a TN summation framework in **Chapter 5**, which is demonstrated to be applicable to feature fusion and eigenvalue extraction of large-scale matrices.

Relevant Publications:

[80] G. G. Calvi, I. Kisil, Y. L. Xu, A. Cichocki, and D. P. Mandic, "A tale of nodes and edges: From basic tensor operations to advanced concepts", *In Preparation for IEEE Signal Processing Magazine*.

[81] 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.

2. **Insights into the Canonical Polyadic Decomposition**
The Canonical Polyadic Decomposition (CPD) decomposes a tensor as a sum of *rank-1* tensors, and is one of the pillar decompositions together with the Tucker and the Tensor Train [63, 82]. The fundamental problems of the CPD are (i) its computation is a rather expensive procedure, which involves an Alternating Least Squares (ALS) approach and is hence prone to local minima; (ii) there exists no deterministic method for the choice of the rank-1 elements (i.e. the rank), R, that the tensor needs to be decomposed in, implying that an optimal value has to be searched iteratively. **Chapter 4** addresses these problems. First, a novel method to estimate the CPD of a higher order tensor is presented, which is based on approximating a tensor through a TT decomposition followed by individual CPDs of the TT factors. Secondly, a lower bound on the tensor rank, R, is proposed, with the aim of reducing the search space for R. Mild conditions under which the bound is attained are also provided.

Relevant Publications:

[83] A. H. Phan, A. Cichocki, I. Oseledets, G. G. Calvi, S. A. Asl and D. P. Mandic, "Tensor networks for latent variable analysis: Higher order canonical polyadic decomposition", *IEEE Transactions on NEural Networks and Learning Systems*, vol. 31, no. 6, pp. 2175–2188, 2019.

[84] G. G. Calvi, B. Scalzo Dees and D. P. Mandic, "A lower bound on the tensor rank based on its maximally square matrix unfolding", *Elsevier Signal Processing*, vol. 180, Art no. 107862, 2020.

3. **A Financial Application for the Support Tensor Machine**
   The Support Tensor Machine (STM) is a tensor extension of its matrix counterpart, that is, the Support Vector Machine (SVM). Because the STM operates on tensors rather than vectors, if the data are in tensor format and present underlying structure, the STM is expected to perform better [1]. Although the STM has found applications in fields such as image processing [85], it had yet to be applied to time-series analysis. **Chapter 6** addresses this by applying the STM to the problem of financial forecasting, by predicting the direction of movement of price of the S&P 500. Results show that the STM applied to an appropriately *tensorized* dataset outperforms the SVM.

   Relevant Publications:

   [86] G. G. Calvi, V. Lucic, and D. P. Mandic, "Support tensor machine for financial forecasting", *In Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8152-8156, 2019.

4. **Merging Tensors and Neural Networks**
   In recent years, there has been a surge in interest for bridging the gap between neural networks and tensors. Owing to the desirable compression properties of tensors, the merging of networks and tensors has the main aim of reducing the huge number of parameters required for training deep learning models [57, 76, 77, 78]. Despite these verified benefits, little has been done to derive the back-propagation algorithm [87] in terms of the tensor factors (most applications rely on specific software packages), or exploiting the expressiveness of tensors to improve model interpretability. These issues are addressed in **Chapter 7**, wherein the Tucker Tensor Layer (TTL) is introduced. In addition to achieving network compression, the tensor valued back-propagation algorithm is fully derived analytically. This deepens the mathematical understanding of the model training process, hence mitigating the black-box issue inherent to neural networks. Moreover it is shown how this knowledge may be used to train larger networks more efficiently. While **Chapter 7** is mostly theoretical, **Chapter 8** takes a more practical approach by adequately tensorizing a Recurrent Neural Network through the TT decomposition and applying it to finance, with the goal of forecasting the direction of price movement of the Japanese Yen. The compression and regularization properties of the so called TT-RNN are demonstrated.

   Relevant Publications:

   [88] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao and D. P. Mandic "Compression and interpretability of neural networks via tucker tensor layer: From first principles to tensor-valued back-propagation", *Submitted to Neural Computation*, 2020.

   [89] Y. L. Xu, G. G. Calvi, and D. P. Mandic, "Tensor-train recurrent neural networks for financial forecasting", *Submitted to the Proceedings of the 46th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

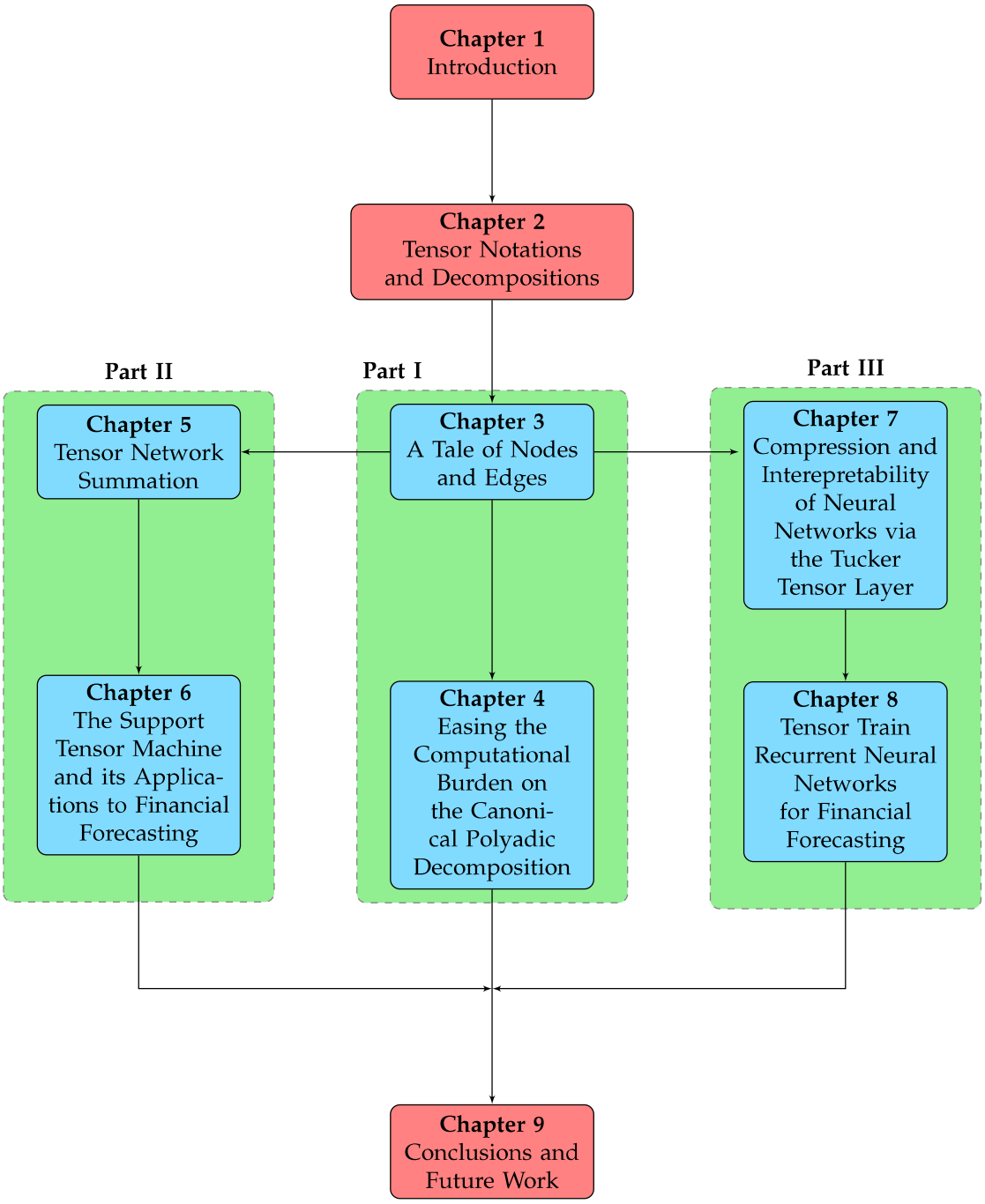


Figure 1.4: Thesis Structure.

Figure 1.4 shows an overview of the structure of the thesis which is organized as follows:

1. **Part I: Insights into Tensor Decompositions**
   This part focuses on providing novel insights pertinent to tensor decompositions and new techniques for their application. **Chapter 3** deals with TNs, and shows how any decomposition can be interpreted as a diagrammatic TN. This implies that tensor decompositions can be studied through intuitive graphical representations, paving the way for new perspectives even on the most fundamental concepts, including tensor contractions, inner products, and unfoldings. The importance of this concept is further demonstrated through the introduction of a new algorithm for tensor contractions, the Tensor Train Contraction Product (TTCP), solely based on a graphical approach. The ideas introduced in **Chapter 3** are a recurring theme throughout this thesis, as they are employed to elucidate important concepts. In **Chapter 4**, the focus is shifted towards the CPD, a pillar tensor decomposition, which is put under closer mathematical scrutiny. A TT-based approach to compute the CPD for higher order tensors is introduced, and a lower bound on the tensor rank is proposed, both with the aim of mitigating the computational burden associated to this decomposition.

2. **Part II: Tensors for Shallow Learning**
   This part of the thesis focuses on developing tensor methods from well established signal processing techniques. Drawing an analogy with feature fusion, **Chapter 5** introduces a framework for TN summation, which is shown to be beneficial through an application to an image classification problem. Moreover, it is also demonstrated how such a framework can be employed to perform eigenvalue extraction for very large scale matrices, which is extremely relevant for numerous signal processing and machine learning applications. **Chapter 6** extends the notion of the well-known Support Vector Machine (SVM) to the tensor domain, by presenting the Support Tensor Machine (STM). Emphasis is given on how to adequately preprocess data for the STM, on how to employ kernel methods, and, in particular, on how the STM can be applied to financial forecasting.

3. **Part III: Tensors for Deep Learning**
   This part focuses on the integration of tensors and neural networks. **Chapter 7** introduces the Tucker Tensor Layer (TTL) as an alternative to the fully connected layers often used in neural networks applications. Back-propagation is fully derived in terms of the tensor factors, and the TTL is applied to several synthetic and real-world datasets. Owing to the compressive properties of tensors, the TTL is shown to be able to significantly reduce the number of parameters of the network, at a small sacrifice in performance. Moreover, drawing from the concepts presented in **Chapter 3**, novel insights into the training process of the network are presented, thus mitigating the well-known "black-box" issue inherent to neural networks. While **Chapter 7** is highly theoretical, **Chapter 8**, is more practically oriented. A different type of tensorized neural network, namely the Tensor-Train Recurrent Neural Network (TTRNN) is considered and, similarly as in **Chapter 6**, is applied to a financial application. Its compression and regularization properties are demonstrated, and, analogously to the TTL, the TTRNN is able to gain insights into the network's training process thanks to the expressiveness of tensors.

Chapter 2

Tensor Notations and Decompositions

> You'll see, you'll come to understand. These big things, these terrible things, are not the important ones. If they were, how could one go on living? No, it is the small, little things that make up a day, that bring fullness and happiness to a life.
>
> *Benedict Freedman*

Chapter Overview

This chapter provides the required background in tensor algebra, also referred to as *multi-linear* algebra, for the ideas presented in this thesis. A list of necessary definitions are first presented, then the main nomenclature is introduced, and fundamental concepts, such as *tensorization*, are outlined. The main tensor decompositions are presented, these are the Canonical Polyadic, the Tucker, and the Tensor Train decompositions. For each, algorithms for the computation are outlined, together with notions on their uniqueness conditions. This chapter concludes with the introduction of the building blocks of Tensor Networks are also outlined, which represent tensors and their operations graphically through interactions of nodes and edges.

2.1 Nomenclature and Notation

This section introduces the general knowledge and background required for this thesis. Refer to Table 2.1 for the basic nomenclature used throughout.

2.1.1 Basics

The following definitions are required. These terms will be used extensively.

Definition 2.1. *A **tensor** is a multi-way array having N indices, and it is written as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$.*

Definition 2.2. *The n-th **mode** of a tensor is an the n-th index of the tensor.*

Definition 2.3. *The **order** of a tensor is the number of its modes.*

Definition 2.4. *A **fiber** of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the collection of entries obtained by fixing all but one index, e.g. $\mathbf{x}(:, i_2, i_3, \ldots, i_N)$ is obtained by fixing all i_n for $n = 2, 3, \ldots, N$.*

Definition 2.5. *A **slice** of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the collection of entries obtained by fixing all but two indices, e.g. $\mathbf{X}(:, :, i_3, \ldots, i_N)$ is obtained by fixing all i_n for $n = 3, 4, \ldots, N$.*

Table 2.1: Basic Tensor and Matrix Notation

$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$	Tensor of order N of size $I_1 \times I_2 \times \cdots \times I_N$
$\underline{\mathbf{X}}$	Tensor Network representation of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
$x_{i_1 i_2 \cdots i_N} = \boldsymbol{\mathcal{X}}(i_1, i_2, \cdots, i_N)$	$(i_1, i_2, \cdots, i_N)$ entry of $\boldsymbol{\mathcal{X}}$
x, $\mathbf{x}$, $\mathbf{X}$	Scalar, vector, matrix
$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$	Mode-n unfolding of tensor $\boldsymbol{\mathcal{X}}$
$\mathbf{A}^{(n)}$	Factor matrices, in tensor decompositions
$(\cdot)^T$, $(\cdot)^{-1}$, $(\cdot)^\dagger$	Transpose, inverse, and Moore-Penrose pseudoinverse operators for matrices
$\circ, \otimes, \odot, \circledast$	Outer, Kronecker, Hadamard and Khatri-Rao products
$\text{vec}(\boldsymbol{\mathcal{X}}) = \mathbf{x} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$	Vectorization of tensor $\boldsymbol{\mathcal{X}}$
$\|\| \cdot \|\|_F$	Frobenius norm
$\mathbf{I}_M$	Identity matrix of size $M \times M$
$\mathbf{1}_M$	Vector of ones of size M
$\langle \cdot, \cdot \rangle$	Inner product

Definition 2.6. *The **mode-**n unfolding of tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *is denoted by* $\mathbf{X}_{(n)}$*, and is a matrix with entries*

$$(\mathbf{X}_{(n)})_{i_n, \overline{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N}} = x_{i_1, i_2, \ldots, i_N} \tag{2.1}$$

where, according to the Little-Endian convention [46]:

$$\overline{i_1 i_2 \ldots i_N} = i_1 + (i_2 - 1) I_1 + (i_3 - 1) I_1 I_2 + \cdots + (i_n - 1) I_1 \ldots I_{N-1} \tag{2.2}$$

This is sometimes referred to in the literature as "classical" or "Kolda-unfolding" [63, 70]. Unless stated otherwise, when a tensor is said to be unfolded the Kolda-unfolding is implied.

The unfolding of a 3-rd order tensor, along each of its modes, is shown in Figure 2.1.

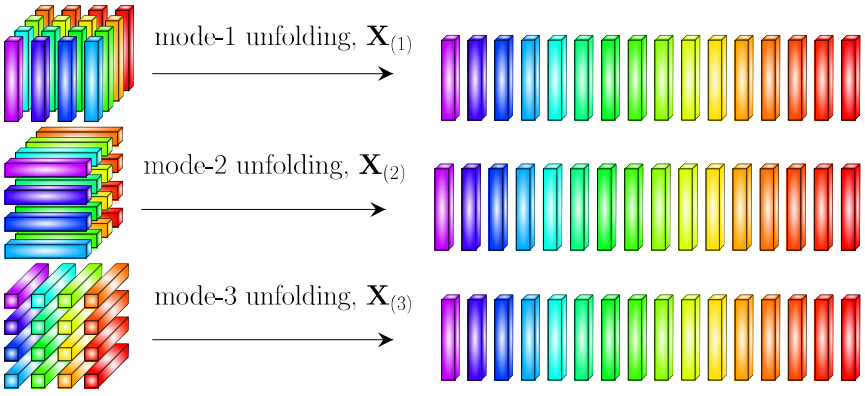


Figure 2.1: Illustration of the mode-1, mode-2, and mode-3 unfoldings of a 3-rd order tensor

Given an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ and another M-th order tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_m \times \cdots \times J_M}$, with sizes of $I_n = J_m$, their (m,n)-contraction , $\times_n^m$, yields a third tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$ of order $(N+M-2)$, with entries

$$c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \sum_{i_n}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} b_{j_1,\ldots,j_{m-1},i_n,j_{m+1},\ldots,j_M} \tag{2.3}$$

The operator $\times_n$ is equal by convention to $\times_n^2$, and referred to as "mode-n" contraction or product. It is used for contraction between tensors and matrices. A few properties of the mode-n product are in order and summarized below:

- $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \mathbf{U} \iff \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)}$
- $\boldsymbol{\mathcal{X}} \times_m \mathbf{A} \times_n \mathbf{B} = \boldsymbol{\mathcal{X}} \times_n \mathbf{B} \times_m \mathbf{A}$, if $m \neq n$
- $\boldsymbol{\mathcal{X}} \times_n \mathbf{A} \times_n \mathbf{B} = \boldsymbol{\mathcal{X}} \times_n (\mathbf{B}\mathbf{A})$

The Kronecker product of matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and matrix $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$ yields a matrix $\mathbf{C} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}$ with entries

$$c_{(i_1-1)J_1+j_1,(i_2-1)J_2+j_2} = a_{i_1 i_2} b_{j_1 j_2} \tag{2.4}$$

whereas the Khatri-Rao product of $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R] \in \mathbb{R}^{I \times R}$ and $\mathbf{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_R] \in \mathbb{R}^{J \times R}$ yields a matrix $\mathbf{C} \in \mathbb{R}^{IJ \times R}$, with columns

$$\boldsymbol{c}_r = \boldsymbol{a}_r \otimes \boldsymbol{b}_r \tag{2.5}$$

The above is summarized in Table 2.2.

Table 2.2: Definition of Main Products

$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_n^m \boldsymbol{\mathcal{B}}$	(m,n)-contraction product of tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ and tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_m \times \cdots \times J_M}$ yields $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$
$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_n \mathbf{B}$	Mode-n products of tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and matrix $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$ yields $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$
$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \bar{\times}_n \mathbf{b}$	Mode-n tensor-vector contraction of tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and vector $\mathbf{b} \in \mathbb{R}^{I_n}$ yields the $(N-1)$-th order tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}$
$\boldsymbol{\mathcal{C}} = [\![\boldsymbol{\mathcal{A}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]\!]$	Full multilinear product $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \cdots \times_N \mathbf{B}^{(N)}$
$\boldsymbol{\mathcal{X}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)}$	Outer product of vectors $\boldsymbol{a}^{(n)} \in \mathbb{R}^{(I_n)}, n = 1, \ldots, N$, yields a rank-1 tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$	Kronecker product of $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$ yields $\mathbf{C} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}$
$\mathbf{C} = \mathbf{A} \circledast \mathbf{B}$	Khatri-Rao product of $\mathbf{A} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} \in \mathbb{R}^{J \times R}$ yields $\mathbf{C} \in \mathbb{R}^{IJ \times R}$

2.1.2 Tensorization via Rearrangement of Lower-Dimensional Data

2.1.2.1 Reshaping or Folding

One of the simplest ways to perform tensorization is by reshaping or "folding" the original data [90]. This method preserves the original sequential ordering and the original number of elements, not requiring any additional storage space.

Definition 2.7. *An N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be a **folding** of vector $\mathbf{x} \in \mathbb{R}^{I_1 I_2 \ldots I_N}$ if*

$$\boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_N) = \mathbf{x}(i) \tag{2.6}$$

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 example, consider a vector $\mathbf{x} \in \mathbb{R}^L$. This is tensorized into a matrix $\mathbf{X} \in \mathbb{R}^{L \times L/I}$ by

$$\mathbf{X} = \begin{bmatrix} x(1) & x(I+1) & \ldots & x(L-I+1) \\ x(2) & x(I+2) & \ldots & x(L-I+2) \\ \vdots & \vdots & \ddots & \vdots \\ x(I) & x(2I) & \ldots & x(L) \end{bmatrix} \tag{2.7}$$

2.1.2.2 Hankel Folding

A Hankel matrix is a structured matrix in which each ascending skew diagonal from left to right is constant, and has found numerous applications in signal processing and control theory [91].

Definition 2.8. *A **Hankel matrix** $\mathbf{X} \in \mathbb{R}^{I \times J}$ is defined as*

$$\mathbf{X} = \begin{bmatrix} x(1) & x(2) & \cdots & x(J) \\ x(2) & x(3) & \cdots & x(J+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(I) & x(I+1) & \cdots & x(L) \end{bmatrix} \tag{2.8}$$

It can be convenient to tensorize a vector or a matrix via a process that injects such a structure, known as "Hankel Folding" [60]. Then, the following definition follows

Definition 2.9. *An N-th order **Hankel tensor** $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is constructed from a generating vector $\mathbf{x} \in \mathbb{R}^L$, where $L = \sum_{n=1}^{N} I_n - N + 1$, with entries*

$$\boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_N) = \mathbf{x}(i_1 + i_2 + \cdots + i_N - N + 1) \tag{2.9}$$

2.2 Tensor Decompositions

Following their original application in chemometrics and psychometrics, Tensor Decompositions (TDs) have expanded to several other fields over the last two decades [63]. Examples include signal processing [64, 82, 92], linear algebra [24, 61] , computer vision [85, 93], graph

analysis [94], and more. In the following the main TDs are introduced, that is, the Canonical Polyadic, the Tucker, and the Tensor Train Decompositions (CPD, TKD, and TT, respectively).

2.2.1 Canonical Polyadic Decomposition

The first to propose the idea of a polyadic form of a tensor was Hitchcock in 1927 [10], that is, expressing a tensor as the sum of a finite number of rank-1 tensors. In 1944, Cattell proposed ideas for having multiple axes for analysis in psychology [11], but it was not until the 1970s that it became popular thanks to Carroll and Chang [17], after they applied it within the psychometrics community under the name of CANonical DECOMPosition (CANDECOMP), and to Harshman who named it PARAllel FACtor model (PARAFAC) [16]. Today it is known as Canonical Polyadic Decomposition (CPD).

Definition 2.10. *An N-th order tensor is **rank-1** if it can be written as an outer product of N vectors, that is*

$$\boldsymbol{\mathcal{X}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)} \tag{2.10}$$

In other words, each element of the tensor is the product of the corresponding vector elements

$$x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}, \text{ for all } 1 \leq i_n \leq I_n \tag{2.11}$$

The CPD decomposes a tensor into a sum of rank-1 tensors. Consider, for example, a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. The CPD expresses it as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \lambda_r \mathbf{a}^{(1)}_r \circ \mathbf{a}^{(2)}_r \circ \mathbf{a}^{(3)}_r \tag{2.12}$$

where $\mathbf{a}^{(1)} \in \mathbb{R}^{I_1}$, $\mathbf{a}^{(2)} \in \mathbb{R}^{I_2}$, $\mathbf{a}^{(3)} \in \mathbb{R}^{I_3}$. If the approximation in (2.12) is exact, then R is said to be the rank of the tensor.

Definition 2.11. *The **rank** of a tensor $\boldsymbol{\mathcal{X}}$, rank($\boldsymbol{\mathcal{X}}$) $= R$, is the smallest number of rank-1 tensors the sum of which generates $\boldsymbol{\mathcal{X}}$ exactly.*

Elementwise, (2.12) takes the form of

$$\boldsymbol{\mathcal{X}}(i_1, i_2, i_3) = \sum_{r=1}^{R} \lambda_r a^{(1)}_{i_1 r} a^{(2)}_{i_2 r} a^{(3)}_{i_3 r} \tag{2.13}$$

and if the rank-1 component vectors are collected into matrices, e.g. $\mathbf{A}^{(1)} = [\mathbf{a}^{(1)}_1, \mathbf{a}^{(1)}_2, \cdots, \mathbf{a}^{(1)}_R]$, the CPD can be expressed with multi-linear products as

$$\begin{aligned} \boldsymbol{\mathcal{X}} &\approx \boldsymbol{\mathcal{D}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \\ &= [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!] \end{aligned} \tag{2.14}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times R}$, $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times R}$, and $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{R \times R \times R}$ is a tensor having only

the λ_r's on its diagonal, that is

$$\mathcal{D}(r_1, r_2, r_3) = \begin{cases} \lambda_r, \text{ if } r_1 = r_2 = r_3 = r \\ 0, \text{ otherwise} \end{cases} \tag{2.15}$$

The CPD for the 3-rd order case is shown in Figure 2.2.

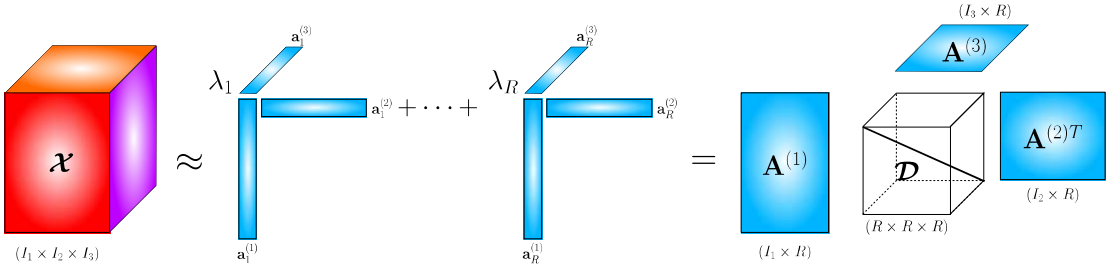


Figure 2.2: Illustration of the CPD for a 3-rd order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$.

The CPD can also be written in matrix form, which consists in unfolding the decomposed tensor $\mathcal{X}$ in terms of its CP factors. For the 3-rd order case, this becomes

$$\begin{aligned} \mathbf{X}_{(1)} \approx & \mathbf{A}^{(1)} (\mathbf{A}^{(3)} \circledast \mathbf{A}^{(2)})^T \\ \mathbf{X}_{(2)} \approx & \mathbf{A}^{(2)} (\mathbf{A}^{(3)} \circledast \mathbf{A}^{(1)})^T \\ \mathbf{X}_{(3)} \approx & \mathbf{A}^{(3)} (\mathbf{A}^{(2)} \circledast \mathbf{A}^{(1)})^T \end{aligned} \tag{2.16}$$

For an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the CPD takes the general form of

$$\begin{aligned} \mathcal{X} \approx & \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} \\ = & \mathcal{D} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times \cdots \times_N \mathbf{A}^{(N)} \\ = & [\![\mathcal{D}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \end{aligned} \tag{2.17}$$

whereby its mode-n unfolding is expressed as

$$\mathbf{X}_{(n)} \approx \mathbf{A}^{(n)} \mathbf{D} (\mathbf{A}^{(N)} \circledast \cdots \circledast \mathbf{A}^{(n+1)} \mathbf{A}^{(n-1)} \circledast \cdots \circledast \mathbf{A}^{(1)})^T \tag{2.18}$$

where $\mathbf{D} = \text{diag}(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_R]$. If $\boldsymbol{\lambda} = \mathbf{1}_R$, then a tensor in the CPD format can simply be written as $\mathcal{X} \approx [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$.

2.2.1.1 Uniqueness

It is in order to note that rank decompositions of higher-order tensors are often unique, whereas the analogous matrix decompositions are not. For example, consider a matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, and assume it is of rank R. Then, a rank decomposition of this matrix could be

$$\mathbf{X} = \mathbf{A}\mathbf{B}^T = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \tag{2.19}$$

However, if the SVD is performed, $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, one could choose (i) $\mathbf{A} = \mathbf{A}\mathbf{\Sigma}$ and $\mathbf{B} = \mathbf{V}$, or (ii) $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}$ and $\mathbf{B} = \mathbf{V}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{R\times R}$ is any orthogonal matrix.

The CPD decomposition, on the other hand, is unique under much milder conditions, up to a permutation and scaling ambiguity [63]. For example, if a 3-rd order tensor admits a CPD decomposition $\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)} = [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!]$, permutation ambiguity means that

$$\boldsymbol{\mathcal{X}} = [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!] = [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}\mathbf{P}, \mathbf{A}^{(2)}\mathbf{P}, \mathbf{A}^{(3)}\mathbf{P}]\!] \tag{2.20}$$

for any $\mathbf{P} \in \mathbb{R}^{R\times R}$ permutation matrix. Instead, scaling ambiguity refers to

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r (\alpha_r \mathbf{a}_r^{(1)}) \circ (\beta_r \mathbf{a}_r^{(2)}) \circ (\gamma_r \mathbf{a}_r^{(3)}) \tag{2.21}$$

as long as $\alpha_r\beta_r\gamma_r = 1$ for all $r = 1, \ldots, R$. On the other hand, the mild conditions required for CPD uniqueness are due to Kruskal [55, 95] and depend on the so called *Kruskal rank* or *k-rank*.

Definition 2.12. *The **k-rank** of a matrix $\mathbf{A}$, denoted by $k_{\mathbf{A}}$, is the maximum possible k such that any k columns of $\mathbf{A}$ are linearly independent.*

Note that the above is not the same as the classical definition of matrix rank, which is the maximum number of independent column (or row) vectors of a matrix (both definitions are equivalent). The subtle but significant difference in *k-rank* lies in the word "any". The result by Kruskal states that a sufficient condition for uniqueness for the 3-rd order case is

$$k_{\mathbf{A}^{(1)}} + k_{\mathbf{A}^{(2)}} + k_{\mathbf{A}^{(3)}} \geq 2R + 2 \tag{2.22}$$

This was extended to the general case for an N-th order tensor by Sidiropolous and Bro [96], whereby sufficient conditions for CPD uniqueness are given by

$$\sum_{n=1}^{N} k_{\mathbf{A}^{(n)}} \geq 2R + (N - 1) \tag{2.23}$$

2.2.1.2 Computation

There exists no finite algorithm for determining the rank of a tensor, as it is an NP-hard problem [95]. Therefore, the first issue arising when computing a CPD is the choice of the rank, R. If the tensor to be decomposed is noise-free, then the computation can be done for increasing values of R until the model fits the underlying tensor 100%. However, most tensors are noisy, so either specific "rank consistency diagnostics" need to be performed (such as CORCONDIA by Bro and Kiers [97]), or a satisfactory value for R is arbitrarily chosen [63].

For a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, CPD solves the following optimization problem

$$\begin{aligned} &\arg\min_{\hat{\boldsymbol{\mathcal{X}}}} ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}|| \\ &\text{s.t. } \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)} = [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!] \end{aligned} \tag{2.24}$$

The most common approach to solve the above is to use an Alternating Least Squares (ALS) procedure. That is, any matrix $\mathbf{A}^{(n)}$, $n = 1, \ldots, N$, is estimated one at a time while keeping the others fixed. For example, $\mathbf{A}^{(1)}$ is estimated via

$$\hat{\mathbf{A}} = \mathbf{X}_{(1)}(\mathbf{A}^{(3)} \circledast \mathbf{A}^{(2)})(\mathbf{A}^{(3)T}\mathbf{A}^{(3)} \odot \mathbf{A}^{(2)T}\mathbf{A}^{(2)})^{\dagger} \tag{2.25}$$

where $\hat{\mathbf{A}} = \mathbf{AD}$, $\mathbf{D} = \text{Diag}(\boldsymbol{\lambda})$, and $\mathbf{A}^{(2)}$, $\mathbf{A}^{(3)}$ are kept fixed. These are then computed in an analogous way, until convergence [63]. The procedure for an N-th order tensor is summarized in Algorithm 1.

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

Input: Data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and rank R

Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$, for $n = 1, \ldots, N$ and vector $\boldsymbol{\lambda}$

1: Initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$
2: **while** not converged or *max iterations* reached **do**
3: **for** $n = 1$ to N **do**
4: $\mathbf{U} \leftarrow \mathbf{A}^{(1)T}\mathbf{A}^{(1)} \odot \cdots \odot \mathbf{A}^{(n-1)T}\mathbf{A}^{(n-1)} \odot \mathbf{A}^{(n+1)T}\mathbf{A}^{(n+1)} \odot \cdots \odot \mathbf{A}^{(N)T}\mathbf{A}^{(N)}$
5: $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \circledast \cdots \circledast \mathbf{A}^{(n+1)} \circledast \mathbf{A}^{(n-1)} \circledast \cdots \circledast \mathbf{A}^{(1)})\mathbf{U}^{\dagger}$
6: **end for**
7: Normalize columns of $\mathbf{A}^{(n)}$ and store them in vector $\boldsymbol{\lambda}$
8: **end while**
9: **Return** $\mathbf{A}^{(n)}$, for $n = 1, \ldots, N$, and $\boldsymbol{\lambda}$.

2.2.2 Tucker Decomposition

The Tucker Decomposition (TKD) was first introduced by Tucker in 1963 [13], and is a higher-order form of the well known Principal Component Analysis (PCA). It decomposes a tensor into a core tensor which is transformed by a factor matrix along each mode. Elementwise, for a 3-rd order tensor, it takes the form of

$$\boldsymbol{\mathcal{X}}(i_1, i_2, i_3) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} a^{(3)}_{i_3 r_3}, \tag{2.26}$$

Similarly as with the CPD, given a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the TKD can be compactly written as

$$\begin{aligned} \boldsymbol{\mathcal{X}} &\approx \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3}^{I_3} g_{r_1 r_2 r_3} \mathbf{a}^{(1)}_{r_1} \circ \mathbf{a}^{(2)}_{r_2} \circ \mathbf{a}^{(3)}_{r_3} \\ &= \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \\ &= [\![\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}]\!] \end{aligned} \tag{2.27}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times R_2}$, $\mathbf{A}^{(3)} \in \mathbb{R}^{I_3 \times R_3}$ are called *factor matrices*, and can be thought of as the principal components in each mode, whereas the tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is referred to as the *core tensor* and its entries determine the level of interaction between the different components. An illustration of the TKD for a 3-rd order tensor is shown in Figure 2.3.

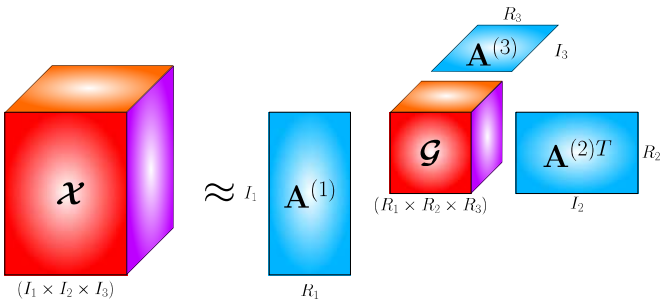


Figure 2.3: Illustration of the TKD for a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$.

The original tensor $\boldsymbol{\mathcal{X}}$ can also be unfolded in terms of its TKD factors, as

$$\begin{aligned} \mathbf{X}_{(1)} &= \mathbf{A}^{(1)}\mathbf{G}_{(1)}(\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T \\ \mathbf{X}_{(2)} &= \mathbf{A}^{(2)}\mathbf{G}_{(2)}(\mathbf{A}^{(3)} \otimes \mathbf{A}^{(1)})^T \\ \mathbf{X}_{(3)} &= \mathbf{A}^{(3)}\mathbf{G}_{(3)}(\mathbf{A}^{(2)} \otimes \mathbf{A}^{(1)})^T \end{aligned} \tag{2.28}$$

For a general N-th order tensor, the TKD takes the form of

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)} \\ &= [\![\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \end{aligned} \tag{2.29}$$

whereby its mode-n unfolding is expressed as

$$\mathbf{X}_{(n)} = \mathbf{A}^{(n)}\mathbf{G}_{(n)}(\mathbf{A}_{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)})^T \tag{2.30}$$

where $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ are the factor matrices. The tuple $(R_1, \ldots, R_N)$ is referred to as the *multilinear rank*.

2.2.2.1 Computation

Two principal computation methods for the TKD are considered: the Higher-Order Singular Value Decomposition (HOSVD), and the Higher-Order Orthogonal Iteration (HOOI), both proposed by De Lathauwer in [24], and in [64], respectively. The HOSVD is a a direct generalization of standard matrix SVD, in the sense that the factor matrices are composed of the singular vectors, for each mode, which interact with each other according to the singular values in the core tensor. Moreover, the factor matrices are orthogonal, and the core tensor is *all-orthogonal*, meaning that slices in each modes are mutually orthogonal (see [3] for further detail). The procedure for the HOSVD is provided in Algorithm 2.

Algorithm 2. HOSVD for an N-th order tensor

Input: Data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and multilinear rank $(R_1, \ldots, R_N)$

Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$, for $n = 1, \ldots, N$ and core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

1: **for** $n = 1$ to N **do**
2: $\quad \mathbf{A}^{(n)} \leftarrow R_n$ principal singular vectors of $\mathcal{X}_{(n)}$
3: **end for**
4: $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \cdots \times_N \mathbf{A}^{(N)T}$
5: **Return** $\boldsymbol{\mathcal{G}}$ and $\mathbf{A}^{(n)}$, for $n = 1, \ldots, N$.

The HOSVD may also be truncated for compression, however, this approximation is not optimal in terms of Frobenius norm. It can be, nonetheless, a good starting point for iterative procedures, sush as the HOOI. Assuming a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, HOOI solves the following optimization

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{G}}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}} \left|\left| \boldsymbol{\mathcal{X}} - [\![\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \right|\right|_F \\ &\text{s.t. } \boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N} \text{ and } \mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n} \\ &\text{where } \mathbf{A}^{(n)} \text{ are columnwise orthogonal, for } n = 1, \ldots, N \end{aligned} \tag{2.31}$$

The procedure computing HOOI is summarized in Algorithm 3.

Algorithm 3. HOOI for an N-th order tensor

Input: Data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and multilinear rank $(R_1, \ldots, R_N)$

Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$, for $n = 1, \ldots, N$ and core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

1: Initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for $n = 1, \ldots, N$, using HOSVD
2: **while** not converged or *max iterations* reached **do**
3: $\quad$ **for** $n = 1$ to N **do**
4: $\quad\quad \boldsymbol{\mathcal{Y}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \cdots \times_{n-1} \mathbf{A}^{(n-1)T} \times_{n+1} \mathbf{A}^{(n+1)T} \times_{n+2} \cdots \times_N \mathbf{A}^{(N)T}$
5: $\quad\quad \mathbf{A}^{(n)} \leftarrow R_n$ principal singular vectors of $\mathbf{Y}_{(n)}$
6: $\quad$ **end for**
7: **end while**
8: $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \cdots \times_N \mathbf{A}^{(N)T}$
9: **Return** $\boldsymbol{\mathcal{G}}$ and $\mathbf{A}^{(n)}$, for $n = 1, \ldots, N$.

2.2.3 Tensor Train Decomposition

The Tensor Train (TT) decomposition was first introduced by Oseledets in [98] and [27], with the purpose of overcoming the curse of dimensionality, as algorithms which operate on tensors can become intractable if their order, N, is high. Perhaps the most common (and simplest) form of TT is the one referred to as the Matrix Product state (MPS or TT-MPS), within the quantum physics community [3]. This consists in decomposing an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, into N, 2-nd or 3-rd order, tensors

$$\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}, \quad n = 1, \ldots, N \tag{2.32}$$

where $R_0 = R_N = 1$. By defining $\mathbf{G}^{(n)}(i_k)$, $i_k = 1, \ldots, I_k$ as the i_k-th vertical slice of tensor $\mathcal{G}^{(n)}$, the MPS assumes the elementwise form of

$$\begin{aligned}\mathcal{X}(i_1, i_2, \ldots, i_N) &= \mathbf{G}^{(1)}(i_1)\mathbf{G}^{(2)}(i_2)\cdots\mathbf{G}^{(N)}(i_N) \\ &= \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\cdots\sum_{r_{(N-1)}=1}^{R_{(N-1)}} g^{(1)}_{1 i_1 r_1} g^{(2)}_{r_1 i_2 r_2}\cdots g^{(N)}_{r_{(N-1)} i_N 1}\end{aligned} \tag{2.33}$$

Here the tuple $(R_1, \ldots, R_{N-1})$ takes the name of *TT-rank*, and the tensors $\mathcal{G}^{(n)}$, as is the case for the TKD, are referred to as *core tensors*. The conditions on R_0 and R_N ensure that the output of (2.33) is a scalar. An illustrative example of the TT-decomposition is provided in Figure 2.4.

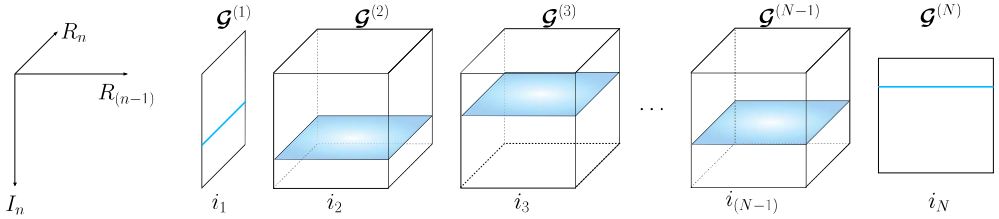


Figure 2.4: Illustration of the TT-decomposition for an N-th order tensor.

The TT decomposition in (2.33) may also be expressed as a series of (m, n)-contractions (refer to Table 2.2) as

$$\begin{aligned}\mathcal{X} &= \mathcal{G}^{(1)} \times_2^1 \mathcal{G}^{(2)} \times_3^1 \mathcal{G}^{(3)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(N)} \\ &= (\!|\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}|\!)\end{aligned} \tag{2.34}$$

Remark 2.1. Because $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(N)}$ are matrices, often they are denoted by $\mathcal{G}^{(1)} = \mathbf{A} \in \mathbb{R}^{I_1 \times R_1}$, $\mathcal{G}^{(N)} = \mathbf{B} \in \mathbb{R}^{R_{(N-1)} \times I_N}$

2.2.3.1 Computation

The algorithm computing the TT-decomposition for a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in the MPS consists in iteratively applying standard SVD to different unfoldings of the underlying tensor, hence its name: TT-SVD [27]. More precisely, the TT-SVD approximates an original tensor via a TT, with a prescribed accuracy ϵ. In other words, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is approximated by the TT-representation of $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, such that

$$||\mathcal{X} - \mathcal{Y}||_F \leq \epsilon ||\mathcal{X}||_F \tag{2.35}$$

For completeness, the procedure is reported in Algorithm 4.

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

Input: Data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and prescribed accuracy ϵ

Output: Core tensors $\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}$ approximating $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$

Initialize cores $\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}$
Compute truncation parameter $\delta = \frac{\epsilon}{\sqrt{N-1}} ||\boldsymbol{\mathcal{X}}||_F$
$R_0 = 1$
$\boldsymbol{\mathcal{Z}} \leftarrow \boldsymbol{\mathcal{X}}$, and $\mathbf{Z} \leftarrow \mathbf{Z}_{(1)}$
for $n = 1$ to $N - 1$ **do**
    Compute δ-truncated SVD: $\mathbf{Z} = \mathbf{USV} + \mathbf{E}$, s.t. $||\mathbf{E}||_F \leq \delta$; $\mathbf{U} \in \mathbb{R}^{R_{(n-1)} I_n \times R_n}$
    $\boldsymbol{\mathcal{G}}^{(n)} \leftarrow \text{reshape}(\mathbf{U}, [R_{(n-1)}, I_n, R_n])$
    $\mathbf{Z} \leftarrow \text{reshape}(\mathbf{SV}^T, [R_n I_{(n+1)}, I_{(n+2)} I_{(n+3)} \ldots I_N])$
end for
$\boldsymbol{\mathcal{G}}^{(N)} \leftarrow \mathbf{Z}$
Return $\boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}$

2.3 Tensor Networks

A precise description of tensor operations via standard conventions and mathematical terms can easily become cumbersome and lose intuition. To tackle this issue, researchers have drawn inspiration from quantum physics and quantum chemistry methodologies wherein mathematical operations are represented diagrammatically [1]. Tensors, and hence tensor decompositions, can be represented as a connection of nodes and edges, yielding structures resembling mathematical graphs [3, 80]. Such structures are referred to as Tensor Networks (TNs) [1, 3, 60, 80].

Within a TN, any tensor of any order (including scalars, vectors, and matrices) is represented as nodes, while the number of edges connect to it is an indication of its order, as shown in Figure 2.5. The dimensionality of each mode is labelled on the corresponding edge in an anti-clockwise fashion.

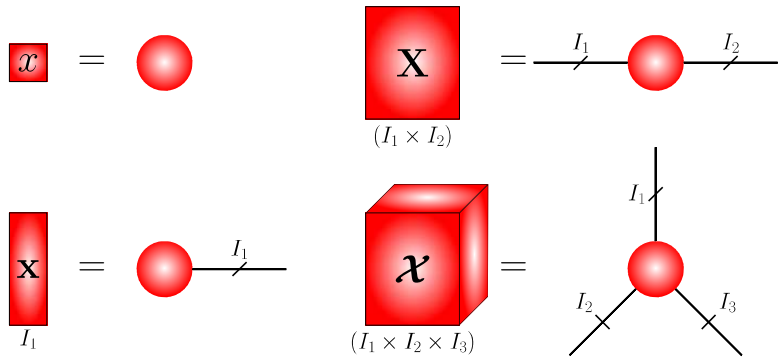


Figure 2.5: Basic building blocks for TNs. From left to right, top to bottom: 0-th order (scalar), 1-st order (vector), 2-nd order (matrix) and 3-rd order tensors.

The basics of TN operations are summarized in Figure 2.6. Notice how this representation has the potential to intuitively represent complex operations, often involving several tensors at

once. Moreover, it offers avenues for insights even in the most simple tensor decompositions, not only benefiting new researchers in the field, but also allowing familiar researchers to attain a deeper grasp of this fascinating field. Indeed, the way tensors interact within the context of TNs requires special attention, which provides the focus of the next chapter.

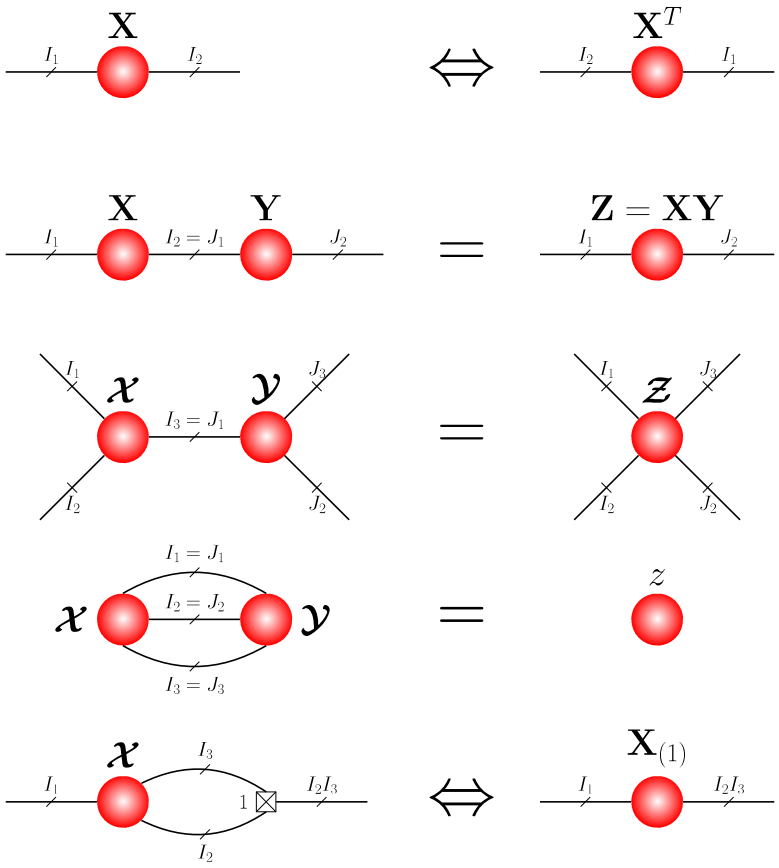


Figure 2.6: Basics of TN operations. From top to bottom: matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ and its transpose; contraction between matrices $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{Y} \in \mathbb{R}^{J_1 \times J_2}$ along the common mode $I_2 = J_1$; contraction between tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ along the common mode $I_3 = J_1$; contraction between tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ along all modes; unfolding of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ along its first mode I_1.

2.4 Chapter Summary

The main tensor notation and operations have been reviewed in Section 2.1, wherein Tables 2.1 and 2.2 will often be referred to in later chapters. Implications of both the *blessing* as well as the *curse of dimensionality* have also been provided, while Section 2.2 has presented the main tensor decompositions, namely the Canonical Polyadic (Section 2.2.1), the Tucker (Section 2.2.2), and the Tensor Train (Section 2.2.3) decompositions, which underpin most of the ideas presented in this thesis. Finally, Section 2.3 has introduced the main building blocks of Tensor Networks (TNs), and their fundamental operations are portrayed in Figure 2.6. The next chapter builds on the concept of TNs by showing how they and tensor decompositions are in reality two sides

of the same coin, notion which is exploited to gain new insights and deeper understanding of tensor operations.

Part I

Insights into Tensor Decompositions

Chapter 3

A Tale of Nodes and Edges

> Nothing sticks in your head better than a story. Stories can express the most complicated ideas in the most digestible ways.
>
> *Sam Balter*

Chapter Overview

This chapter aims to offer new understandings of the underlying mathematical operations involving tensors and tensor decompositions, via the graphical representation introduced in **Chapter 2**, Section 2.3, which is commonly adopted within Tensor Networks (TNs). In the following it is shown how a tensor decomposition implies a TN, in fact, the simplest TNs merely are a consequence of direct decompositions, implying a duality between the two entities. In other words, tensor decompositions can be studied via intuitive graphical representations, reducing the interaction of its components to connections between nodes and edges. Moreover, this chapter reveals how the power of intuition offered by a graphical depiction of mathematical formulae can be crucial in attaining new grasps even on the most fundamental concepts, benefiting not only new researchers in the field, but also those familiar with the topic, by offering efficient tools promoting new findings. A practical example of this is shown in later sections of this chapter, wherein an efficient algorithm for performing (m, n)-contractions between any two tensors of any order is devised through a purely graphical approach.

3.1 Introduction

By inspecting Tables 2.1 and 2.2, together with the main operations and tensor decompositions introduced in **Chapter 2**, it becomes immediately clear that precise description of tensor operations via conventional mathematical terms and equations may become cumbersome and confusing, especially for higher order tensors. Hence, methodologies have been developed to interpret such operations diagrammatically, as outlined in later sections of **Chapter 2**. Any N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, can be represented as a node with edges, where the number of edges represents its order, as illustrated in Figures 2.5 and 2.6. This, in turn, allows for intuitive ways to correctly portray even the most complex operations. For instance, Figure 3.1 portrays two interactions among tensors which would translate in an extremely tedious and hard to follow formula, as several matrices, 3-rd order, and 4-th order tensors are present. Even the relatively concise multilinear operations, such as (2.29) for the Tucker (TKD) and (2.34) for the Tensor Train (TT) decompositions, would be difficult to outline.

In fact, even single decompositions may lose their immediate intuition for higher order

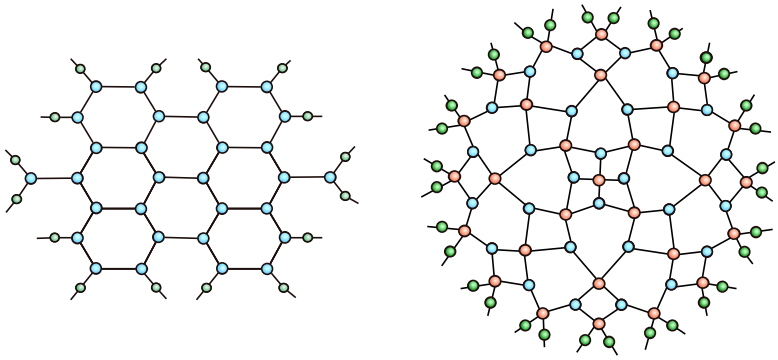


Figure 3.1: Example of two relatively involved tensor operations, but easily comprehensible through a graphical representation. Image adapted from [1].

tensors. Already for 4-th order tensors, illustrations such as those in Figures 2.12, 2.3, 2.4, become infeasible. To tackle this issue, researchers indeed do represent single decompositions as Tensor Networks in graphical format, an approach which is summarized in the following. Moreover, in later sections of this chapter it is shown how such representations may be exploited to deepen understandings of the fundamental concepts, as well as they promote new findings and novel applications.

3.2 Tensor Decompositions as Tensor Networks

In this section, the tensor decompositions presented in Section 2.2 are revisited under the lens of the diagrammatical approach introduced in Section 2.3. Observe Figure 3.2. The conventional way of representing the CPD, TKD, and TT in an illustrative fashion is portrayed in the left column, while their graphical equivalent, using the building blocks shown in Figure 2.5 and the operations in Figure 2.6, are portrayed in the right column. For ease of representation, 3-rd order tensors are shown as red nodes, while matrices as blue nodes. In Figure 3.2b, the diagonal core of the CPD is represented as a white node with a diagonal stripe. In all cases, a connection between two nodes mathematically represents a contraction.

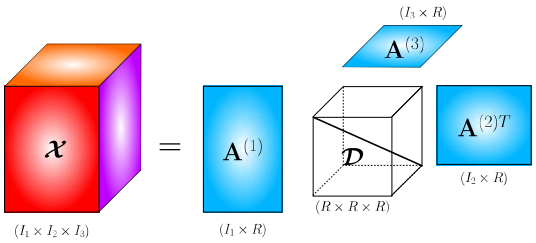
Indeed, it is now possible to fully appreciate the meaning of the (m, n)-contraction product expressed as the operator $\times_n^m$, as opposed to a summation over common indices defined in (2.3). Contraction products are an intuitive and efficient way of simplifying tensor operations, and their expressiveness fully becomes evident in light of graphical tensor representations.

3.3 Tensor Operations via Tensor Networks

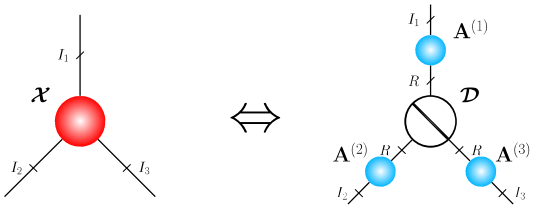
3.3.1 Insights in the Tucker Decomposition

Consider, for example, the TKD of a 3-rd order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. If orthogonality constraints are imposed [24], then the following holds:

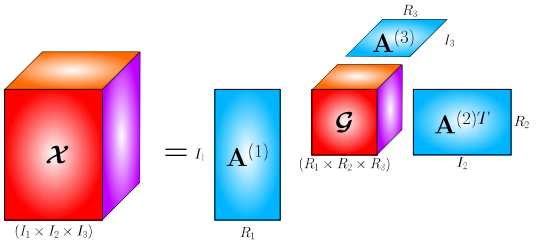
$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \iff \boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \mathbf{A}^{(3)T} \tag{3.1}$$



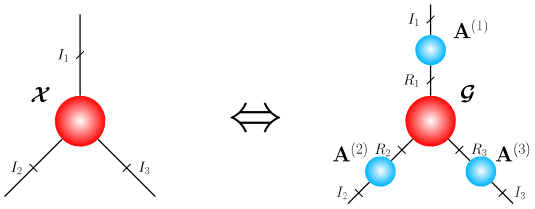
(a) Conventional illustration of CPD.



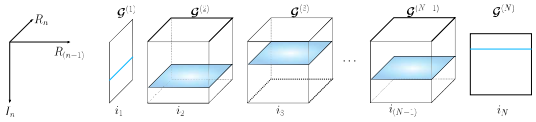
(b) Graphical illustration of CPD.



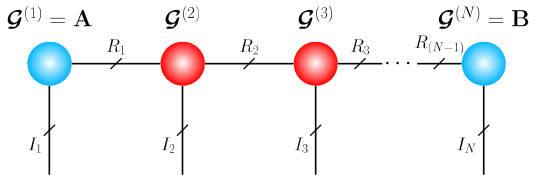
(c) Conventional illustration of TKD.



(d) Graphical illustration of TKD.



(e) Conventional illustration of TT.



(f) Graphical illustration of TT.

Figure 3.2: Comparison between conventional (left) and graphical (right) representations of tensors decompositions.

where $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, 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 an immediate full intuitive understanding of (3.1) can be difficult to grasp. However, if the problem is analyzed graphically as shown in Figure 3.3, it becomes immediately clear that the right-hand side of (3.1) reduces to $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ being multiplied by an identity matrix $\mathbf{I}_{R_n}$ along each mode, which causes no effect.

Graphical representations may be employed as a tool to tackle even more complicated mathematical problems. Consider the optimization problem for the Higher-Order Orthogonal Iteration (HOOI), shown in (2.31). This is equivalent to minimizing the square of the objective function, that is

$$\begin{aligned}
\left\| \mathcal{X} - [\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \right\|_F^2 &= ||\mathcal{X}||_F^2 - 2\langle \mathcal{X}, [\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \rangle + ||[\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]||_F^2 \\
&= ||\mathcal{X}||_F^2 - 2\langle [\![\mathcal{X}; \mathbf{A}^{(1)T}, \ldots, \mathbf{A}^{(N)T}]\!], \mathcal{G} \rangle + ||\mathcal{G}||_F^2 \\
&= ||\mathcal{X}||_F^2 - 2\langle \mathcal{G}, \mathcal{G} \rangle + ||\mathcal{G}||_F^2 \\
&= ||\mathcal{X}||_F^2 - ||\mathcal{G}||_F^2 \\
&= ||\mathcal{X}||_F^2 - ||\mathcal{X} \times_1 \mathbf{A}^{(1)T} \times_2 \cdots \times_N \mathbf{A}^{(N)T}||_F^2
\end{aligned} \tag{3.2}$$

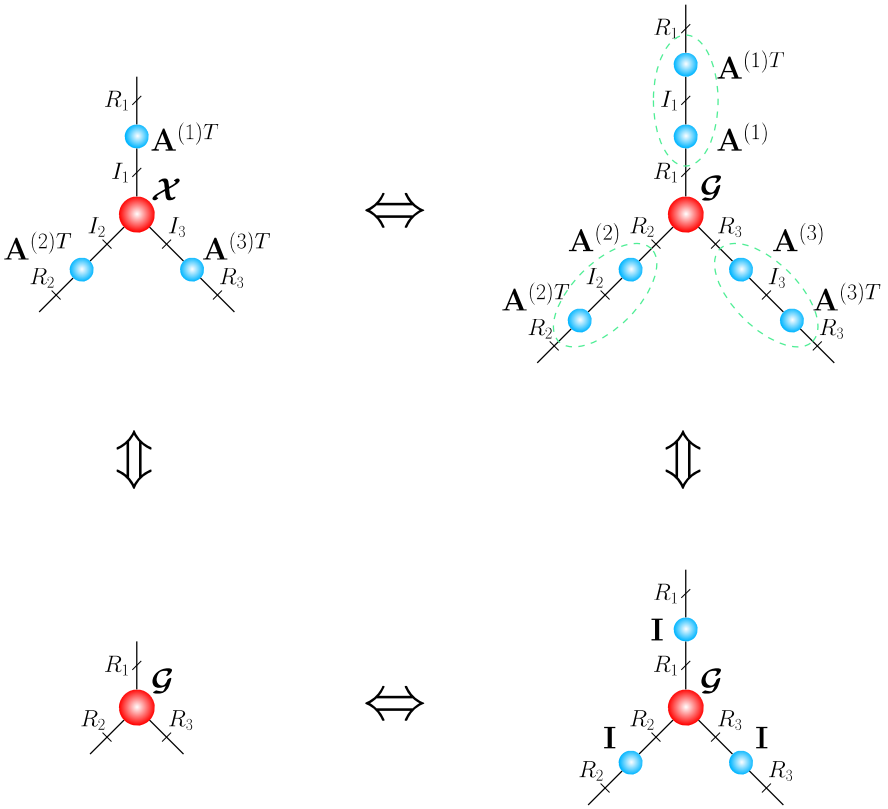


Figure 3.3: Graphical representation of $\mathcal{G} = \mathcal{X} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \mathbf{A}^{(3)T}$. By starting from the TKD of $\mathcal{X}$ and exploiting the fact that $\mathbf{A}^{(n)T}\mathbf{A} = \mathbf{I}_{R_n}$, the equality becomes evident.

The above can be reformulated as a series of subproblems and solved via an ALS approach [63]. What is shown next is a justification of the steps in (3.2), by virtue of the introduced graphical approach.

First, consider the equivalence

$$\langle \mathcal{X}, [\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \rangle = \langle [\![\mathcal{X}; \mathbf{A}^{(1)T}, \ldots, \mathbf{A}^{(N)T}]\!], \mathcal{G} \rangle \tag{3.3}$$

The reason why this holds becomes obvious by observing Fig. 3.4, in which, for simplicity, a 3-rd order tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, is considered. An inner product between any two tensors equals contraction along all modes [63], represented in the top-right corner of Fig. 3.4. It is possible to manipulate these graphical representations: instead of considering $\mathcal{G}$ and factor matrices $\mathbf{A}^{(n)}$ together, we consider $\mathcal{X}$ and $\mathbf{A}^{(n)T}$ (lower portion of Fig. 3.4). Notice that in the case of $\mathcal{X}$ the contracting modes are I_n, while for $\mathcal{G}$ contractions are carried out on modes R_n. As a result the lower-left section of Fig. 3.4, and hence (3.3), follow [80].

The same approach can be used to justify

$$||[\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]||_F^2 = ||\mathcal{G}||_F^2 \tag{3.4}$$

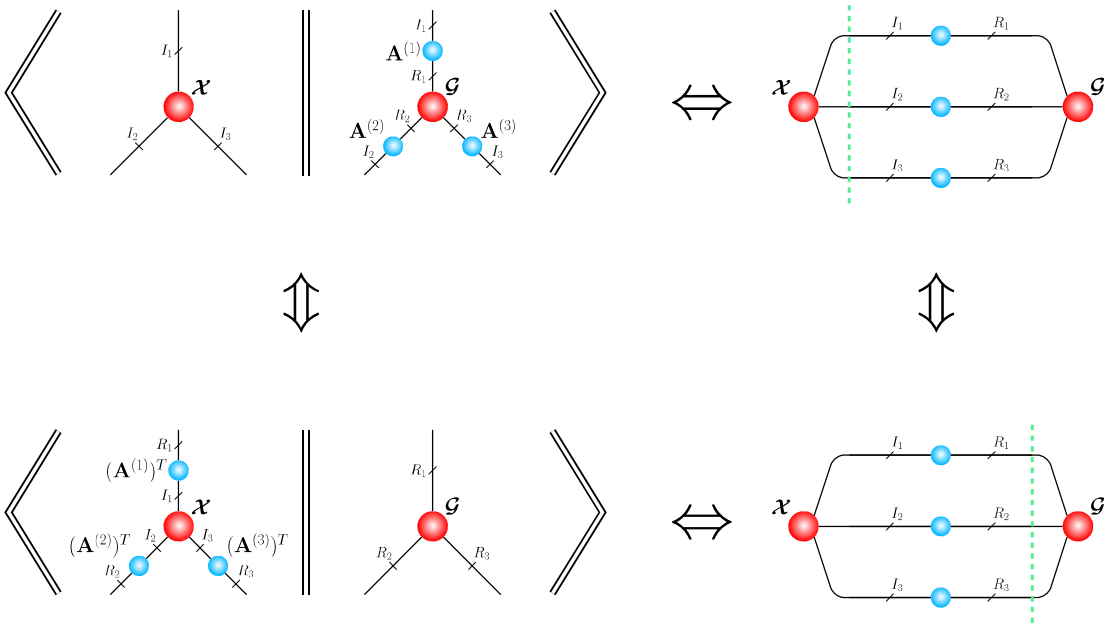


Figure 3.4: A tensor inner product, from a graphical perspective. Treating the matrices as individual blocks and changing the order of contraction, it is possible to manipulate the original inner product into one expressed differently, but ultimately equivalent.

Because $\langle \mathcal{X}, \mathcal{X} \rangle = ||\mathcal{X}||^2$ [1], the procedure portrayed in Fig. 3.4 is repeated with

$$\left\langle [\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!], [\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \right\rangle \tag{3.5}$$

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

$$\left\langle [\![\mathcal{G}; \mathbf{I}^{(1)}, \ldots, \mathbf{I}^{(N)}]\!], \mathcal{G} \right\rangle = \langle \mathcal{G}, \mathcal{G} \rangle = ||\mathcal{G}||_F^2 \tag{3.6}$$

justifying (3.4).

Remark 3.1. The above properties about inner products are valid for any tensor, but were presented in the light of the TKD for convenience, as a way to show some of the benefits of graphical representations of TNs.

3.3.2 Tensor Unfoldings

It is often convenient to unfold a tensor into a matrix, while in the TKD format. This allows 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 (2.28) and (2.30). Although not particularly complicated, these equations might be difficult to accept if no justification is provided. To this end, observe Figure 3.5. The question is how do matrices $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ take part in the operation? The second and third dimension of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, I_2 and I_3, 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 maintain dimensionality matches and allow for

contraction it also becomes natural to unfold $\mathcal{G}$ 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 $\mathbf{K} = \mathbf{K}_*^T = (\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T \in \mathbb{R}^{R_2 R_3 \times I_2 I_3}$.

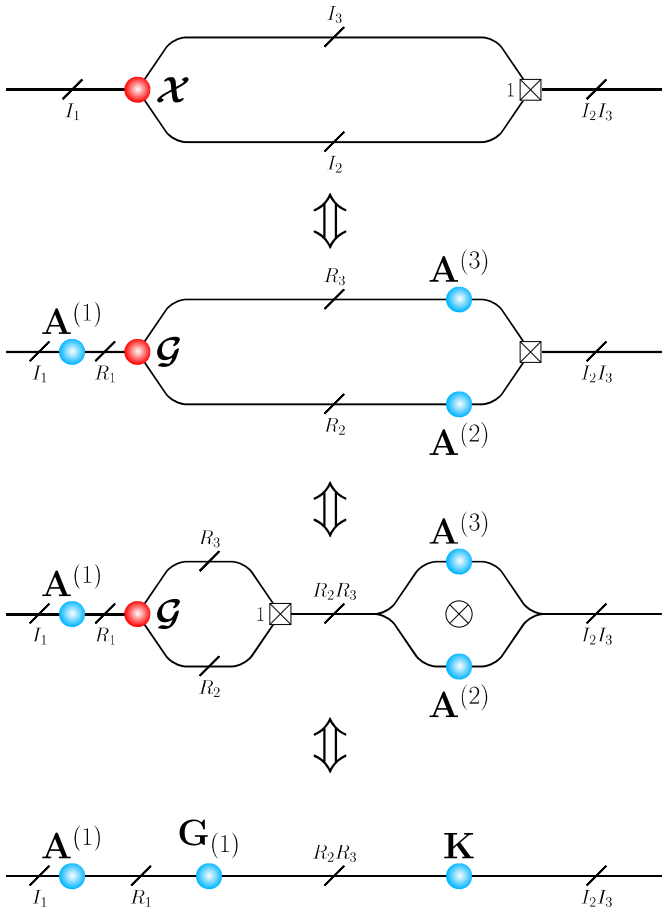


Figure 3.5: Graphical justification of (2.28). By performing the 1-st mode unfolding graphically, it becomes clear that $\mathbf{X}_{(1)} = \mathbf{A}^{(1)}\mathbf{G}_{(1)}\mathbf{K}$, where $\mathbf{K} = (\mathbf{A}^{(3)} \otimes \mathbf{A}^{(2)})^T$.

Remark 3.2. 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.

3.3.3 A Note on Topology

Together with the elements composing it, the topology of a TN defines the TN itself. This means that any two TNs, even if they are represented differently but have the same topology and elements, are equal. This result is intuitive and a foundation of Graph Theory [99]. However it is possible to express the same graphical representation of a tensor with different equations. In other words, any TN has multiple mathematical representations, which are all equivalent.

To illustrate this, consider a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format, that is

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \tag{3.7}$$

The above is equivalent to

$$\boldsymbol{\mathcal{X}} = \mathbf{A}^{(1)} \times_2^1 \left(\boldsymbol{\mathcal{G}} \times_2 \mathbf{A}^{(2)}\right) \times_3^1 \mathbf{A}^{(3)T} \tag{3.8}$$

The equality of (3.7) and (3.8) is evident upon observation of Figure 3.6, which shows a TKD expressed with the same notation which is normally used for a TT, hence the requirement of the transpose on $\mathbf{A}^{(3)}$.

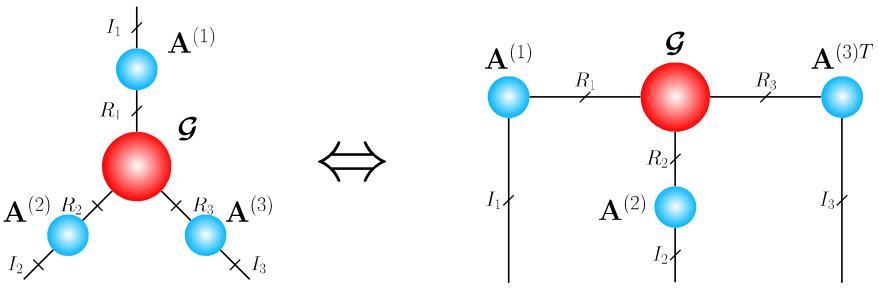


Figure 3.6: Simple example to show that different expressions for the same topology are possible. In this case $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ can be written as in (3.7), on the left-hand side in the figure, or as in (3.8), on the right-hand side.

3.3.4 Contraction of Tensors in the TT-Format

As long as mathematical rigour is maintained, the sole topology of a TN is sufficient to obtain insights which otherwise would be difficult to access. To illustrate this, a novel method to perform tensor contractions is devised, stemming solely from a graphical approach. Recalling the elementwise definition in (2.3), it is clear that a direct implementation would be computationally expensive as it would incur a significative number of iterations, especially for higher-order tensors. However, the graphical representation of the TT in Figure 3.2f suggests that the same problem could be tackled using TTs, as each core has information regarding the corresponding original tensor mode [80]. In particular, the first and last cores are always matrices. In the following it is shown how this can be exploited to reduce an $\times_n^m$ contraction product to a simple matrix multiplication.

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

$$\boldsymbol{\mathcal{X}} \times_3^2 \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times I_2 \times I_4 \times J_1 \times J_3} \tag{3.9}$$

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

$$\begin{aligned} \boldsymbol{\mathcal{G}}_x^{(n)} &\in \mathbb{R}^{R_{(n-1)} \times I_n \times R_n}, \quad n = 1, 2, 3, 4 \\ \boldsymbol{\mathcal{G}}_y^{(m)} &\in \mathbb{R}^{P_{(m-1)} \times J_m \times P_m}, \quad m = 1, 2, 3 \end{aligned} \tag{3.10}$$

This implies that, since $I_3 = J_2$, the desired contraction in (3.9) can be obtained as a simple

matrix product $\mathcal{G}_x^{(1)} = \mathbf{A}_x \in \mathbb{R}^{I_1 \times R_1}$ and $\mathcal{G}_y^{(1)} = \mathbf{A}_y \in \mathbb{R}^{J_2 \times P_1}$ as

$$\mathbf{A}_x^T \mathbf{A}_y = \mathbf{K} \in \mathbb{R}^{R_1 \times P_1} \tag{3.11}$$

This is the centre of the proposed Tensor-Train Contraction Product (TTCP) [80], a method to perform contractions between tensors in the TT-format. Observe Figure 3.7. First, the cores $\mathbf{A}_x$ and $\mathbf{A}_y$ of tensors $\boldsymbol{\mathcal{X}}^\rho$ and $\boldsymbol{\mathcal{Y}}^\rho$ are joined by a normal matrix multiplication $\mathbf{A}_x^T \mathbf{A}_y$, connecting the two TTs of $\boldsymbol{\mathcal{X}}^\rho$ and $\boldsymbol{\mathcal{Y}}^\rho$. This yields matrix $\mathbf{K} \in \mathbb{R}^{R_1 \times P_1}$, which in the newly formed TT is the link introduced by the contraction product. Reconstructing the tensor represented by the union of the two TTs yields $\boldsymbol{\mathcal{Z}}^\rho \in \mathbb{R}^{I_4 \times I_2 \times I_1 \times J_1 \times J_3}$, which is a permuted version of tensor $\boldsymbol{\mathcal{Z}}$ in (3.9). In order to obtain the latter it is sufficient to permute $\boldsymbol{\mathcal{Z}}^\rho$ such that the resulting dimensions follow the natural order obtained by the original contraction. The procedure is summarized in Algorithm 5, where the operator $(\cdot)^{T_{[1;3]}}$ permutes any given tensor such that the first and third modes are exchanged.

Algorithm 5. Tensor-Train Contraction Product (TTCP) on $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ along modes n and m respectively.

1: **Input:** $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, n and m.
2: **Output:** $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \times_n^m \boldsymbol{\mathcal{Y}}$ s.t. $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$
3:
4: **Procedure:**
5: Define $\boldsymbol{\mathcal{X}}^\rho \in \mathbb{R}^{I_n \times I_1 \times \cdots \times I_{n-1} \times I_{n+1} \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}}^\rho \in \mathbb{R}^{J_m \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \cdots \times J_M}$ by permuting $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ accordingly.
6: Obtain TTs of $\boldsymbol{\mathcal{X}}^\rho$ and $\boldsymbol{\mathcal{Y}}^\rho$:

$$\begin{aligned} \boldsymbol{\mathcal{X}}^\rho &= \mathbf{A}_x \times_2^1 \mathcal{G}_x^{(2)} \times_3^1 \times \cdots \times_3^1 \mathcal{G}_x^{(N-1)} \times_3^1 \mathbf{B}_x \\ \boldsymbol{\mathcal{Y}}^\rho &= \mathbf{A}_y \times_2^1 \mathcal{G}_y^{(2)} \times_3^1 \times \cdots \times_3^1 \mathcal{G}_y^{(M-1)} \times_3^1 \mathbf{B}_y \end{aligned} \tag{3.12}$$

7: Perform $\times_n^m$ contraction by:

$$\mathbf{K} = \mathbf{A}_x^T \mathbf{A}_y \tag{3.13}$$

8: Obtain

$$\boldsymbol{\mathcal{Z}}^\rho \in \mathbb{R}^{I_N \times \cdots \times I_{n+1} \times I_{n-1} \times \cdots \times I_1 \times J_1 \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M} \tag{3.14}$$

by

$$\boldsymbol{\mathcal{Z}}_p \leftarrow \mathbf{B}_x \times_2^1 \mathcal{G}_x^{(N-1)T_{[1;3]}} \times_3^1 \mathcal{G}_x^{(N-2)T_{[1;3]}} \times_3^1 \mathcal{G}_x^{(2)T_{[1;3]}} \times_3^1 \mathbf{K} \times_2^1 \mathcal{G}_y^{(2)} \times_3^1 \cdots \times_3^1 \mathcal{G}_y^{(M-1)} \times_3^1 \mathbf{B}_y \tag{3.15}$$

9: Permute $\boldsymbol{\mathcal{Z}}_p$ to obtain $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$

Remark 3.3. Benefits of the TTCP are two-fold:

1. It provides a more efficient alternative to the actual contraction, in comparison to (2.3);
2. The resulting tensor $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \times_1^1 \boldsymbol{\mathcal{Y}}$ is automatically yielded in the TT-format, whereby its cores still contain modal information with respect to the individual tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$.

Remark 3.4. An (m, n)-contraction between $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_m}$ can also be performed by $\mathbf{Z}_* = \mathbf{X}_{(n)}^T \mathbf{Y}_{(m)} \in \mathbb{R}^{I_1 I_2 \ldots I_{(n-1)} I_{(n+1)} \ldots I_N \times J_1 J_2 \ldots J_{(m-1)} J_{(m+1)} \ldots J_M}$, followed

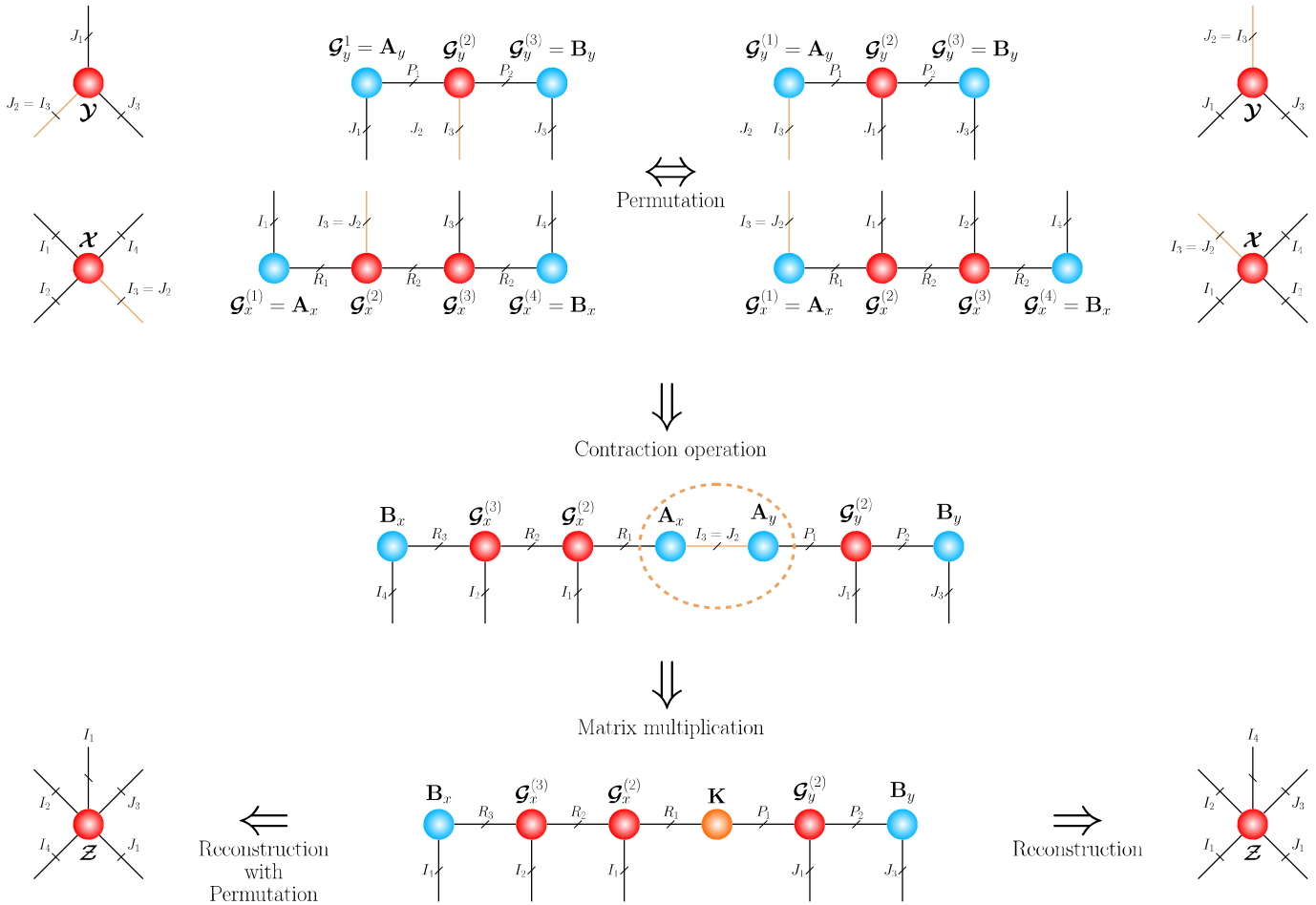


Figure 3.7: Graphical illustration of TTCP. A contraction is performed on a 4th-order tensor $\mathcal{X}$ and a 3rd-order tensor $\mathcal{Y}$ along common modes $I_3 = J_2$. This is done by exploiting the topology of TTs. The TT decompositions of the permuted versions of $\mathcal{X}$ and $\mathcal{Y}$ are obtained, allowing to reduce the contraction to a single matrix multiplication, highlighted by the orange ellipsis and node $\mathbf{K}$. The output $\mathcal{Z}$ is then obtained by contracting the newly formed TT at the bottom of the figure.

by an appropriate reshaping of $\mathbf{Z}_*$. However, doing so would lose modal information of the individual original tensors, as pointed out in Remark 3.3.

Computational benefits of the TTCP can be observed in Figure 3.8. Two tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ are drawn from a zero-mean, unit-variance Gaussian distribution, and three different cases are considered, that is (i) $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{20\times 20\times 20}$, (ii) $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{20\times 20\times 20\times 5}$, and (iii) $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{20\times 20\times 20\times 5\times 4}$. For each case, the contraction $\boldsymbol{\mathcal{X}} \times_1^1 \boldsymbol{\mathcal{Y}}$ is computed via (2.3) and via the TTCP algorithm, and computational times are recorded for both cases. TTCP shows 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 (2.3) tends to become intractable, and the advantages of the TTCP become apparent.

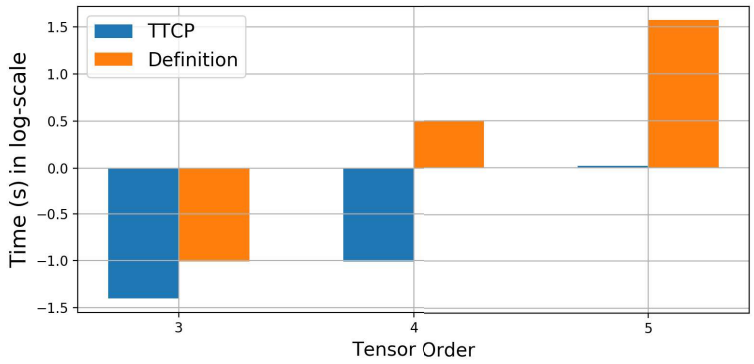


Figure 3.8: Comparison of computational cost (in seconds, log-scale) between (2.3) and TTCP for the $\boldsymbol{\mathcal{X}} \times_1^1 \boldsymbol{\mathcal{Y}}$ contraction. The 3-rd, 4-th and 5-th order cases are considered. Stemming from a graphical intuition, TTCP is more computationally efficient than the direct definition of contraction.

Remark 3.5. The TTCP stems purely from graphical foundations, by making full use of the topology of the TT representations of tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ [80].

Remark 3.6. In the same fashion 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 $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1\times\cdots\times J_N}$. 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 as [51]

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle_{-n} = \mathbf{X}_{(n)} \mathbf{Y}_{(n)}^T \in \mathbb{R}^{I_n \times J_n} \tag{3.16}$$

To graphically justify this, consider the 3-rd order case for two general tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times I_3}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1\times J_2\times J_3}$, with $I_1 = J_1$, $I_2 = J_2$, shown in Figure 3.9. Unfolding is performed on the 3-rd mode, i.e. the mode not taking part in the contraction, for both $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$, and the transpose on $\mathbf{Y}_{(3)}$ is taken in order to avoid a dimension mismatch. It is not difficult to extend and accept this result for any pair of higher order tensors.

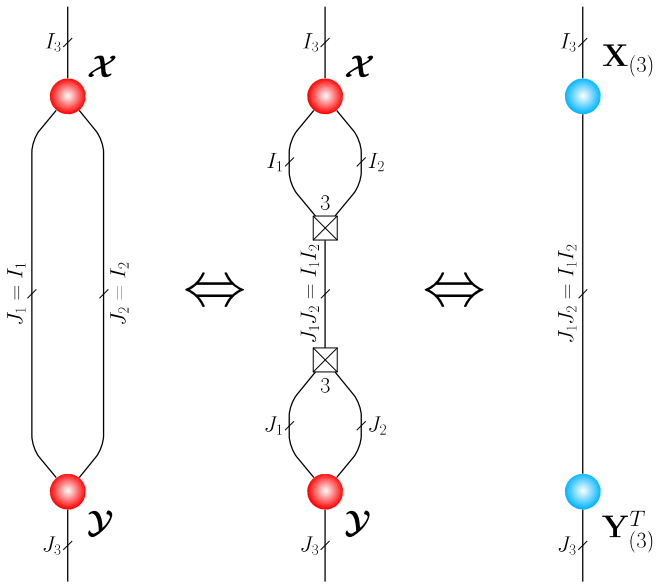


Figure 3.9: Graphical illustration of an "all-but-n" contraction. $\langle \mathcal{X}, \mathcal{Y} \rangle_{-3} = \mathbf{X}_{(3)} \mathbf{Y}^T_{(3)}$ on $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $I_1 = J_1, I_2 = J_2$.

3.4 Chapter Summary

This chapter has presented avenues leading to deeper understanding of tensor operations. This has been achieved by appealing to the graphical representations of tensors introduced in **Chapter 2**, referred to as Tensor Networks (TNs). Implications of such representation have first been explored in light of the TKD (Section 3.3.1) and tensor unfoldings (Section 3.3.2), offering novel interpretations of their corresponding mathematical operations. The Tensor-Train Contraction Product (TTCP) has then been introduced in Section 3.3.4, further emphasizing how the graphical approach promotes not only intuition, but, as a consequence, also the discovery of new techniques. Benefits of the TTCP over the standard definition of contraction have been portrayed in Figure 3.8. This chapter can be interpreted as the main "theme" of this thesis (refer to Figure 1.4). Indeed, the notions here presented will be recalled throughout to elucidate important concepts.

Chapter 4

Easing the Computational Burden on the Canonical Polyadic Decomposition

> Reduce the complexity of life by eliminating the needless wants of life, and the labours of life reduce themselves.
>
> *Edwin Way Teale*

Chapter Overview

While **Chapter 2** provided an overview of the main tensor decompositions, namely the Canonical Polyadic (CPD), Tucker Decomposition (TKD) and Tensor-Train (TT) decompositions, and **Chapter 3** presented intuitive ways to represent them graphically, this chapter takes a closer scrutiny at the CPD. Differently from the TKD and TT decompositions, which can be computed in a non-iterative approach (that is, just one iteration across all modes is required), the CPD is generally computed through an Alternating Least Squares (ALS) procedure, which makes it prone to local minima. This, combined with the fact that the tensor rank, R, may be greater than any of the tensor dimensions, and that the best low-rank tensor approximation may not always exist [63, 100], makes a practical usage of the CPD rather complicated.

To this end, two approaches to ease the computational burden on the CPD are introduced. Firstly, a novel method to estimate the CPD of higher order tensors is presented [83], which relies on approximating a tensor through a TT-MPS decomposition followed by individual CPDs of the corresponding 3-rd order cores. It is shown that the factor matrices of the original higher order tensor are reliably estimated from the compressed TT model. Secondly, a lower bound on the tensor rank, R, based on matrix unfoldings of the associated tensor is proposed [84]. Not only knowledge of lower bounds on R lowers the computational costs by reducing the search spaces for R, but mild conditions under which the proposed bound is attained are provided.

4.1 Introduction

The Canonical Polyadic Decomposition (CPD) has maintained its role as one of the workhorses of tensor decompositions. Since the 1990s the CPD has attracted attention from signal processing researchers [92, 101, 102], including applications in audio and speech processing, biomedical engineering, and chemometrics [38, 47], and also wireless communications [20]. Recent developments in machine learning have instead opened up new applications of the CPD in feature extraction, data reconstruction, and image completion [70, 103, 104]. The CPD is also

well-known for its ability to provide great advantages in dimensionality reduction [105, 106] and also for determining the complexity of matrix multiplication, as this problem corresponds to finding the rank of certain tensors [107, 108].

Over its history, the CPD has been the subject of study of many researchers. Efficient algorithms for its computation have also been developed, and the model has been extended with various additional constraints in order to promote interpretability [109]. Despite its successes, there are still many challenging problems related to the computation of the CPD and its representation. The main stumbling block in computing the CPD for big data is that computation for higher order tensors (i. e. order $N \geq 10$) is quite complicated and involves non-trivial technical difficulties, related to the fact that, due to the Curse of Dimensionality, computational costs of most existing algorithms for the CPD increase exponentially with the tensor order. Assuming equal dimensions for an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, that is, $I_1 = I_2 = \cdots = I_N = I$, the CPD has a computational cost of $\mathcal{O}(NRI^N)$ [110, 111], which explains why most CPD algorithms are efficient only for 3-rd order tensors. Possible solutions to this have been proposed, such as a prior compression of the tensor using a TKD, which can reduce the computational cost of the CPD to $\mathcal{O}(NR^{N+1})$. However, this is applicable when the rank, R, is smaller than the tensor dimensions, but it is a well-known fact that this may not always be the case. Moreover, the computational costs for such an approach still remain high for $N \geq 4$ [83], and, in some cases, to obtain a good tensor approximation the choice of R may be higher than the tensor dimensions.

To address these issues, this chapter introduces two approaches, both having the aim to ease the computational burden associated with the CPD. Firstly, it is shown how the CPD can be reliably estimated from a compressed TT-MPS model of the original tensor. A bidirectional mapping between tensors in the CPD and TT formats is demonstrated, which makes the proposed approach quite general and able to serve a wide variety of purposes [83]. Secondly, a lower bound on R, based on the so-called *maximally square tensor unfolding* (i.e. unfolding with aspect ratio closest to unity) is introduced [84], with the aim to reduce the search space for the choice of R. Implications of this result are illustrated through numerical examples.

4.2 Tensor Train – Canonical Polyadic Decomposition Duality

The key idea behind the computation of the CPD from a TT-MPS formulation finds its physical meaning in the properties of the TT-format, whereby any rank-R tensor has an equivalent TT representation with TT-rank $(R, R, \ldots, R)$ and any tensor in the TT-format with TT-rank $(R_1, R_2, \ldots, R_{N-1})$ has an equivalent CPD representation of rank-$R_1 R_2 \ldots R_{N-1}$ [83]. This is referred to as *TT-CPD duality* and is summarized in the following lemmas.

Lemma 4.1. ([27]) A tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, which is expressed in the CPD format as $\boldsymbol{\mathcal{X}} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$, with $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and scaling factors absorbed within (see Section 2.2.1), can be equivalently expressed as a TT, $\boldsymbol{\mathcal{X}} = (\!|\boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}|\!)$, with $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R \times I_n \times R}$ for $n = 2, \ldots, N-1$, and $\boldsymbol{\mathcal{G}}^{(1)} = \mathbf{A}^{(1)}$, $\boldsymbol{\mathcal{G}}^{(N)} = \mathbf{A}^{(N)}$. The i-th vertical slices of the core tensors $\boldsymbol{\mathcal{G}}^{(n)}$ are diagonal matrices of the i-th rows $\mathbf{A}^{(n)}(i,:)$, for $i = 1, 2, \ldots, I_n$. That is, $\boldsymbol{\mathcal{G}}^{(n)}(:, i, :) = \text{diag}(\mathbf{A}^{(n)}(i,:))$.

Lemma 4.1 indicates that if a TT of rank-$(R, R, \ldots, R)$ is fit to a tensor $\boldsymbol{\mathcal{X}}$, then the approximation error of the tensor in the TT-format will not be worse than that of the best rank-R approximation of $\boldsymbol{\mathcal{X}}$, that is

$$\left|\left|\boldsymbol{\mathcal{X}} - \langle\!\langle \boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)} \rangle\!\rangle\right|\right|_F^2 \leq \left|\left|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]\right|\right|_F^2 \tag{4.1}$$

As part of the TT-CPD duality, a tensor in the CPD format can be equivalently expressed in the TT-format, as shown in the following.

Lemma 4.2. A tensor in the TT-format $\boldsymbol{\mathcal{X}} = \langle\!\langle \boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)} \rangle\!\rangle$ with rank-$(R_1, R_1, \ldots, R_{N-1})$ has an equivalent CPD representation $\boldsymbol{\mathcal{X}} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$ of rank-$R_1 R_2 \ldots R_{N-1}$, where the factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_1 R_2 \ldots R_{N-1}}$ are given by

$$\mathbf{A}^{(n)} = \mathbf{G}^{(n)}_{(2)} \left(\mathbf{1}^T_{R>n} \otimes \mathbf{I}_{R_{n-1} R_n} \otimes \mathbf{1}^T_{R<n-1} \right) \tag{4.2}$$

with $R_{<n-1} = R_0 R_1 \ldots R_{n-2}$ and $R_{>n} = R_{n+1} \ldots R_{N-1} R_N$.

The TT-CPD duality represents the physical meaning serving as the motivation behind the proposed approach, that is the conversion of a tensor in the TT-format with rank-$(R, R, \ldots, R)$ to a CPD representation of rank-R.

Remark 4.1. Since the factor matrices are of dimensionality $I_n \times R_1 R_2 \ldots R_{N-1}$, the CPD representation of a tensor in the TT-format according to Lemma 4.2 implies a rank $R_1 R_2 \ldots R_{N-1}$, which usually exceeds the true rank of the underlying tensor. This is not convenient from a storage perspective as redundancies are being introduced. Hence, a better technique is proposed in the next section.

4.3 Conversion from the TT Format to the CPD Format

Differently from the results of Lemma 4.1, the proposed approach assumes a tensor in the TT-format $\boldsymbol{\mathcal{X}} = \langle\!\langle \boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)} \rangle\!\rangle$ with *dense* core tensors $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R \times I_n \times R}$. In other words, to efficiently compute the CPD and avoid the computational cost $\mathcal{O}(NRI^N)$, an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is first expressed as a TT, which is then converted to a CPD. This is done by retrieving the factor matrices $\mathbf{A}^{(n)}$, $n = 2, \ldots, N-1$ from CPDs of the 3-rd order tensors $\boldsymbol{\mathcal{G}}^{(n)}$, and separately computing $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(N)}$.

Lemma 4.3. Assume that a rank-R, N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ has a unique CPD and a TT representation of rank-$(R, R, \ldots, R)$, that is

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] = \langle\!\langle \mathbf{G}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \mathbf{G}^{(N)} \rangle\!\rangle \tag{4.3}$$

where $\mathbf{G}^{(1)} \equiv \boldsymbol{\mathcal{G}}^{(1)}$ and $\mathbf{G}^{(N)} \equiv \boldsymbol{\mathcal{G}}^{(N)}$. Then $\boldsymbol{\mathcal{G}}^{(n)}$ can be expressed by tensors in the CPD format of rank-R, the second factor matrices of which are $\mathbf{A}^{(n)}$, up to scaling and permutation ambiguity, as

$$\boldsymbol{\mathcal{G}}^{(n)} = [\![\mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!], \quad n = 2, \ldots, N-1 \tag{4.4}$$

where matrices $\mathbf{Q}^{(n)}, \mathbf{S}^{(n)} \in \mathbb{R}^{R\times R}$ satisfy

$$\mathbf{S}_n\mathbf{Q}_{n+1} = \mathbf{D}_n \in \mathbb{R}^{R\times R}, \quad n = 2, \ldots, N-2 \tag{4.5}$$

and matrices $\mathbf{D}_n$ are diagonal. Matrices $\mathbf{A}^{(1)} \in \mathbb{R}^{R\times R}$ and $\mathbf{A}^{(N)} \in \mathbb{R}^{R\times R}$ are then given by

$$\begin{aligned} \mathbf{A}^{(1)} &= \mathbf{G}_1\mathbf{Q}_2 \\ \mathbf{A}^{(N)} &= \mathbf{G}_N^T\mathbf{S}_{N-1}\mathbf{D} \end{aligned} \tag{4.6}$$

where $\mathbf{D} = \text{Diag}(\boldsymbol{\lambda}) \in \mathbb{R}^{R\times R}$, and

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_2 \odot \boldsymbol{\lambda}_3 \odot \cdots \odot \boldsymbol{\lambda}_{N-1} \tag{4.7}$$

with each $\boldsymbol{\lambda}_n \in \mathbb{R}^R$, $n = 2, \ldots, N-1$, being found as the scaling factors of the CPDs for $\boldsymbol{\mathcal{G}}^{(n)}$ in (4.4), but absorbed in the last factor matrix $\mathbf{A}^{(N)}$.

An outline of the proof is provided next. Define

$$\begin{aligned} \mathbf{G}_{<n} &= \left(\mathbf{G}^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}^{(2)} \times_3^1 \cdots \times_3^1 \boldsymbol{\mathcal{G}}^{(n-1)}\right)_{(n)} \in \mathbb{R}^{R\times I_1\cdots I_{n-1}} \\ \mathbf{G}_{>n} &= \left(\mathbf{G}^{(n+1)} \times_3^1 \boldsymbol{\mathcal{G}}^{(n+2)} \times_3^1 \cdots \times_3^1 \boldsymbol{\mathcal{G}}^{(N)}\right)_{(1)} \in \mathbb{R}^{R\times I_{n+1}\cdots I_N} \end{aligned} \tag{4.8}$$

and

$$\begin{aligned} \mathbf{A}_{<n} &= \mathbf{A}^{(n-1)} \circledast \mathbf{A}^{(n-1)} \circledast \cdots \circledast \mathbf{A}^{(1)} \in \mathbb{R}^{I_1\cdots I_{n-1}\times R} \\ \mathbf{A}_{>n} &= \mathbf{A}^{(N)} \circledast \mathbf{A}^{(N-1)} \circledast \cdots \circledast \mathbf{A}^{(n+1)} \in \mathbb{R}^{I_{n+1}\cdots I_N\times R} \end{aligned} \tag{4.9}$$

Next, reshape the tensor from (4.3) to a tensor of size $I_1I_2\cdots I_{n-1} \times I_n \times I_{n+1}I_{n+2}\cdots I_N$, to yield

$$[\![\mathbf{A}_{<n}, \mathbf{A}^{(n)}, \mathbf{A}_{>n}]\!] = \boldsymbol{\mathcal{G}}^{(n)} \times_1 \mathbf{G}_{<n}^T \times_3 \mathbf{G}_{>n}^T \tag{4.10}$$

Because $\boldsymbol{\mathcal{G}}^{(n)}$ is contracted with matrices along its first and third mode, both of dimensionality R, then it can also be expressed as a CPD of rank-R, of which, due to the identity in (4.10), the mode-2 factor matrix is $\mathbf{A}^{(n)}$, that is

$$\boldsymbol{\mathcal{G}}^{(n)} = [\![\mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!], \quad n = 2, \ldots, N-1 \tag{4.11}$$

The CPDs for $\boldsymbol{\mathcal{G}}^{(n)}$ can be found in closed form through Direct Three-Linear Decomposition (DTLD) for which the solution can be found through the generalized eigenvalue decomposition [112]. Replacing (4.11) in the TT formulation of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ yields

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \mathbf{G}^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}^{(2)} \times_3^1 \cdots \times_3^1 \boldsymbol{\mathcal{G}}^{(N-1)} \times_3^1 \mathbf{G}^{(N)} \\ &= \mathbf{G}^{(1)} \times_2^1 [\![\mathbf{Q}_2, \mathbf{A}^{(2)}, \mathbf{S}_2]\!] \times_3^1 [\![\mathbf{Q}_3, \mathbf{A}^{(3)}, \mathbf{S}_3]\!] \times_3^1 \cdots \times_3^1 [\![\mathbf{Q}_{N-1}, \mathbf{A}^{(N-1)}, \mathbf{S}_{N-1}]\!] \times_3^1 \mathbf{G}^{(N)} \end{aligned} \tag{4.12}$$

Defining an N-th order tensor, $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{R\times R\times\cdots\times R}$, as

$$\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{I}}_3 \times_3^1 \mathbf{D}_2 \times_2^1 \boldsymbol{\mathcal{I}}_3 \times_3^1 \cdots \times_3^1 \mathbf{D}_{N-2} \times_2^1 \boldsymbol{\mathcal{I}}_3 \tag{4.13}$$

where $\mathbf{D}_n = \mathbf{S}_n\mathbf{Q}_{n+1} \in \mathbb{R}^{R\times R}$, and $\boldsymbol{\mathcal{I}}_3 \in \mathbb{R}^{R\times R\times R}$ is a 3-rd order tensor with ones on its

diagonal, that is

$$\mathcal{I}_3(r_1, r_2, r_3) = \begin{cases} 1, \text{ if } r_1 = r_2 = r_3, \quad r_1, r_2, r_3 = 1, \dots, R \\ 0, \text{ otherwise} \end{cases} \tag{4.14}$$

then (4.12) can be rewritten as

$$\mathcal{X} = [\![\mathcal{H}; \mathbf{G}^{(1)}\mathbf{Q}_2, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}, \dots, \mathbf{A}^{(N-1)}, \mathbf{G}^{(N)T}\mathbf{S}_{N-1}]\!] \tag{4.15}$$

This is shown in Figure 4.1 for the 4-th order case. Because (4.15) must be identical to the CPD representation in (4.3), $\mathcal{H}$ must admit a CPD of rank-R, and diagonal matrices $\mathbf{D}_1, \mathbf{D}_N \in \mathbb{R}^{R\times R}$ must exist, such that

$$\begin{aligned} \mathcal{I}_N &= \mathbf{D}_1 \times_2^1 \mathcal{H} \times_{(N-1)}^1 \mathbf{D}_{N-1} \\ \mathbf{A}^{(1)} &= \mathbf{G}^{(1)}\mathbf{Q}_2\mathbf{D}_1^{-1} \\ \mathbf{A}^{(N)} &= \mathbf{G}^{(N)T}\mathbf{S}_{N-1}\mathbf{D}_{N-1}^{-1} \end{aligned} \tag{4.16}$$

where $\mathcal{I}_N \in \mathbb{R}^{R\times R\times\cdots\times R}$ is an N-th order tensor, analogously defined as in (4.14).

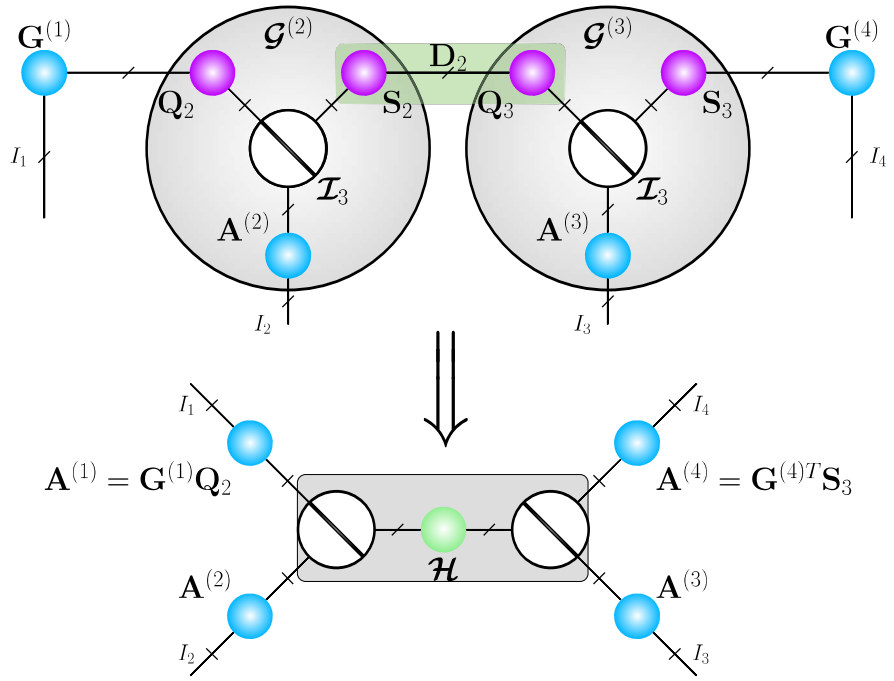


Figure 4.1: Graphical representation of (4.15) for a 4-th order tensor. The big grey nodes represent the CPD of $\mathcal{G}^{(2)}$ and $\mathcal{G}^{(3)}$, while $\mathcal{H}$ is obtained as a contraction of $\mathbf{D}_2$ and the two $\mathcal{I}_3$. This is represented as the grey rectangle in the lower part of the figure, where the green node is $\mathbf{D}_2$. Unlabelled modes have dimensionality R.

Remark 4.2. Note the slight abuse of notation in (4.16), which, at first glance, seems to be in

contrast with Lemma 4.2. In reality, this makes it such that the full contractions are

$$\begin{aligned}\mathbf{A}^{(1)} &= \mathbf{G}^{(1)}\mathbf{Q}_2\mathbf{D}_1^{-1}\mathbf{D}_1 = \mathbf{G}^{(1)}\mathbf{Q}_2 \\ \mathbf{A}^{(N)} &= \mathbf{G}^{(N)T}\mathbf{S}_{N-1}\mathbf{D}_{N-1}^{-1}\mathbf{D}_{N-1} = \mathbf{G}^{(N)T}\mathbf{S}_{N-1}\end{aligned} \tag{4.17}$$

hence conforming with what the lemma states. Matrix $\mathbf{A}^{(N)}$ then has the scaling factors of the CPDs for $\boldsymbol{\mathcal{G}}^{(n)}$, $n = 2, \ldots, N-1$, absorbed within, as in (4.6)-(4.7).

In other words, the presence of $\mathbf{D}_1$ and $\mathbf{D}_{N-1}$ is required merely to ensure diagonality of the CPD core. Moreover, it can be shown [83] that the diagonal matrices $\mathbf{D}_n = \text{Diag}(\boldsymbol{\gamma}_n) \in \mathbb{R}^{R\times R}$, $n = 1, \ldots, N-1$ satisfy

$$\boldsymbol{\gamma}_1 \odot \boldsymbol{\gamma}_2 \odot \cdots \odot \boldsymbol{\gamma}_{N-1} = \mathbf{1}_R \tag{4.18}$$

In turn, this allows $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ to be expressed as

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{G}^{(1)}\mathbf{Q}_2, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{G}^{(N)T}\mathbf{S}_{N-1}]\!] \tag{4.19}$$

4.4 Solving the Permutation Ambiguity

The permutation ambiguity refers to a possible arbitrary ordering of the columns of the factor matrices $\mathbf{A}^{(n)}$ in the CPDs of $\boldsymbol{\mathcal{G}}^{(n)}$. These permutations can be determined through the products $\mathbf{S}_n^T\mathbf{Q}_{n+1}$ for $n = 2, \ldots, N-2$. In practice, these products are not always diagonal matrices, but exhibit some form of permutation, that is

$$\mathbf{S}_n^T\mathbf{Q}_{n+1} = \mathbf{D}_n\mathbf{P}_n = \text{Diag}(\boldsymbol{\gamma}_n)\mathbf{P}_n \tag{4.20}$$

where $\mathbf{P}_n \in \mathbb{R}^{R\times R}$ are the permutation matrices. By identifying the dominant entries in the rows of $\mathbf{S}_n^T\mathbf{Q}_{n+1}$, it is possible to determine $\mathbf{P}_n$ and $\boldsymbol{\gamma}_n$. Consequently, the components of $\boldsymbol{\mathcal{G}}^{(n+1)}$ expressed in the CPD format,

$$\boldsymbol{\mathcal{G}}^{(n+1)} = [\![\boldsymbol{\lambda}_{n+1};\ \mathbf{Q}_{n+1},\ \mathbf{A}^{(n+1)},\ \mathbf{S}_{n+1}]\!] \tag{4.21}$$

can be permuted and normalized to give

$$\boldsymbol{\mathcal{G}}^{(n+1)} = [\![\boldsymbol{\lambda}_{n+1} \odot \mathbf{P}_n\boldsymbol{\gamma}_n;\ \mathbf{Q}_{n+1}\mathbf{P}_n^T\text{Diag}(\mathbf{1}_R \oslash \boldsymbol{\gamma}_n),\ \mathbf{A}^{(n+1)}\mathbf{P}_n,\ \mathbf{S}_{n+1}\mathbf{P}_n]\!] \tag{4.22}$$

where $\oslash$ indicates the elementwise (Hadamard) division. This correction is applied sequentially for factor matrices $\mathbf{A}^{(n)}$, $n = 2, \ldots, N-1$, while $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(N)}$ are given as in Lemma 4.2. The full procedure to convert a tensor from the TT format with TT-rank $(R, R, \ldots, R)$ to a rank-R CPD format is summarized in Algorithm 6.

The computational complexity of Algorithm 6 is $\mathcal{O}(RR^2I_2 + RR^2I_3 + \cdots + RR^2I_{N-1}) = \mathcal{O}\big(R^3\sum_{n=2}^{N-1} I_n\big)$. If $I_1 = I_2 = \cdots = I_N$, the computational cost becomes $\mathcal{O}(NIR^3)$, which, compared to that of the CPD applied directly on an N-th order tensor, $\mathcal{O}(NRI^N)$, or that of the CPD applied on a TKD compressed tensor, $\mathcal{O}(NR^{N+1})$, is substantially cheaper, as shown in Figure 4.2. Clearly, this is of high practical importance.

Algorithm 6. Exact Conversion from the TT-Format to the CPD Format

Input: N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in a rank-$(R, R, \ldots, R)$ TT-format $\boldsymbol{\mathcal{X}} = \langle\!\langle \mathbf{G}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N-1)}, \mathbf{G}^{(N)} \rangle\!\rangle$
for $n = 2, \ldots, N-1$ **do**
  Compute CPD $\boldsymbol{\mathcal{G}}^{(n)} = [\![\boldsymbol{\lambda}_n; \mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!]$
end for
for $n = 2, \ldots, N-2$ **do**
  Find permutation matrix $\mathbf{P}_n$
  $\mathbf{S}_n^T \mathbf{Q}_{n+1} = \text{Diag}(\boldsymbol{\gamma}_n) \mathbf{P}_n$
  $\mathbf{A}^{(n+1)} \leftarrow \mathbf{A}^{(n+1)} \mathbf{P}_n$
  $\mathbf{S}^{(n+1)} \leftarrow \mathbf{S}^{(n+1)} \mathbf{P}_n$
  $\boldsymbol{\lambda}_n \leftarrow \boldsymbol{\lambda}_n \odot \mathbf{P}_n \boldsymbol{\gamma}_n$
end for
$\boldsymbol{\lambda} = \boldsymbol{\lambda}_2 \odot \boldsymbol{\lambda}_3 \odot \cdots \odot \boldsymbol{\lambda}_{N-1}$
$\mathbf{A}^{(1)} = \mathbf{G}^{(1)} \mathbf{Q}_2$
$\mathbf{A}^{(N)} = \mathbf{G}^{(N)T} \mathbf{S}_{N-1} \text{Diag}(\boldsymbol{\lambda})$
Return: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in the CPD format $\boldsymbol{\mathcal{X}} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$

4.5 The Need for a Lower-Bound on the Tensor Rank

The previous section presented a method to efficiently compute the CPD for higher order tensors. This section also deals with easing the CPD computational burden, but, differently from above, this is done by adopting a more theoretical approach to determine techniques to more efficiently estimate the rank of the tensor, R. Indeed, the first issue which arises when computing a CPD is the choice of R, however, determining the rank of a tensor is an NP-hard problem [100]. Although the computation can be performed successively for increasing values of R, until the model fits the underlying tensor arbitrarily well, it is a well-known fact that a perfect fit may not always exist, which implies that a value for R has to be arbitrarily chosen [63]. Because the CPD is intrinsically multilinear, the most common approach for the computation of the CPD is through an Alternating Least Squares (ALS) procedure, whereby the CP factors are estimated one at a time, while the others are kept fixed. The presence of local minima in this approach, combined with the fact that an acceptable value for R has to be sought iteratively, makes CPD a rather computationally expensive procedure.

This calls for the need to relax the computational burden associated with the CPD, by studying the properties of lower bounds on the tensor rank through matrix unfoldings of the associated tensor. Knowledge of lower bounds on R would, in turn, lower the computational cost by reducing the search space for R. In the following it is shown that the maximum of such a bound is equal to the maximum matrix rank attainable by any unfolding of the tensor in hand [84]. Starting from the definition of a "flattened CPD" to allow for a manipulation of the CP factors using standard linear algebra, the theory behind the proposed lower bound is developed, and it is demonstrated that said lower bound relies on a specific permutation of the original tensor. From Sylvester's rank inequality, is subsequently shown how, for generic tensors, the bound is attained under very mild conditions. This, in turn, allows for the tensor rank to be detected by simply inspecting the matrix rank of associated tensor unfoldings. The practical implications of this result are shown through numerical examples.

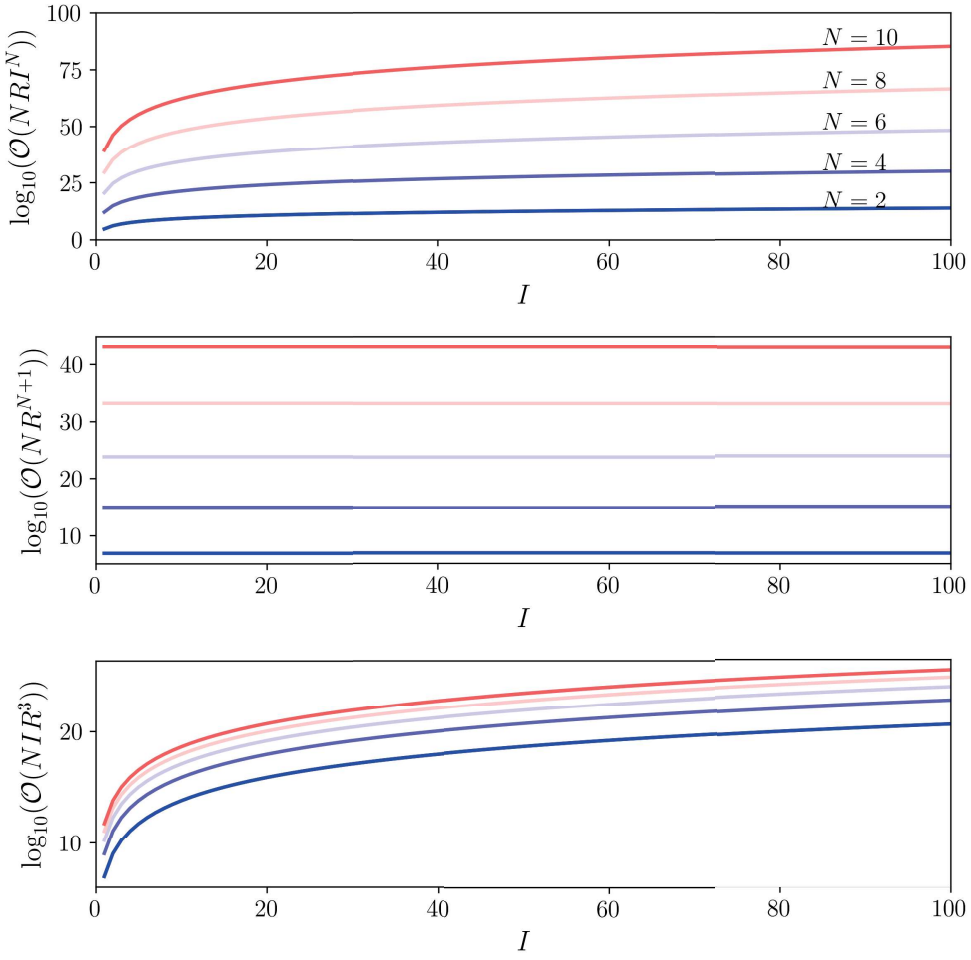


Figure 4.2: Computational cost for the CPD computed through different methods. Top: direct CPD on N-th order tensor. Middle: CPD on a tensor first compressed through the TKD. Bottom: CPD computed from the conversion of a tensor in the TT format to the CPD format.

4.6 Lower Bound Derivation

The following definitions serve as a starting point to arrive at a lower bound on tensor rank, R.

Definition 4.1. *The n-**unfolding** of a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is denoted by $\mathbf{X}_{<n>}$, and represents a matrix with entries*

$$(\mathbf{X}_{<n>})_{\overline{i_1 \ldots i_n}, \overline{i_{n+1} \ldots i_N}} = x_{i_1 i_2 \ldots i_N} \tag{4.23}$$

Definition 4.2. *The n-**rank** of a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is* $\operatorname{rank}(\mathbf{X}_{<n>})$*, i.e. the matrix rank of its n-unfolding.*

Remark 4.3. Recall that the CPD of an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is expressed as

$$\begin{aligned} \boldsymbol{\mathcal{X}} &\approx \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} \\ &= \boldsymbol{\mathcal{D}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times \cdots \times_N \mathbf{A}^{(N)} \\ &\equiv [\![\boldsymbol{\mathcal{D}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \end{aligned} \tag{4.24}$$

If the above approximation is exact, in the following, the underlying tensor is said to *admit a CPD*.

Note that if an N-th order tensor admits a CPD, then its n-mode unfolding is given by

$$\begin{aligned} \mathbf{X}_{<n>} &= (\mathbf{A}^{(n)} \circledast \cdots \circledast \mathbf{A}^{(1)}) \mathbf{D} (\mathbf{A}^{(N)} \circledast \cdots \circledast \mathbf{A}^{(n+1)})^T \\ &= \underbrace{\Big(\overset{1}{\underset{i=n}{\circledast}} \mathbf{A}^{(i)} \Big)}_{\in \mathbb{R}^{I_1 \cdots I_n \times R}} \mathbf{D} \underbrace{\Big(\overset{n+1}{\underset{i=N}{\circledast}} \mathbf{A}^{(i)T} \Big)}_{\in \mathbb{R}^{R \times I_{n+1} \cdots I_N}} \end{aligned} \tag{4.25}$$

Using the fact that, for matrices,

$$\operatorname{rank}(\mathbf{ABC}) \leq \min\{ \operatorname{rank}(\mathbf{A}), \operatorname{rank}(\mathbf{B}), \operatorname{rank}(\mathbf{C}) \} \tag{4.26}$$

and that $\mathbf{D} \in \mathbb{R}^{R \times R}$ is diagonal, the following well-known result is determined,

$$\begin{aligned} \operatorname{rank}(\mathbf{X}_{<n>}) &\leq \min \Big\{ \operatorname{rank}\Big(\overset{1}{\underset{i=n}{\circledast}} \mathbf{A}^{(i)T} \Big), \operatorname{rank}\Big(\overset{n+1}{\underset{i=N}{\circledast}} \mathbf{A}^{(i)T} \Big), \operatorname{rank}(\mathbf{D}) \Big\} \\ &= \min \Big\{ \operatorname{rank}\Big(\overset{1}{\underset{i=n}{\circledast}} \mathbf{A}^{(i)T} \Big), \operatorname{rank}\Big(\overset{n+1}{\underset{i=N}{\circledast}} \mathbf{A}^{(i)T} \Big), R \Big\} \\ &\leq R \end{aligned} \tag{4.27}$$

Because $\operatorname{rank}(\mathbf{X}_{<n>}) \leq R$ for all $n = 1, \ldots, N$, this implies that the tensor rank, R, is bounded from below by its maximum n-rank, that is

$$\max_{n=1,\ldots,N} \{ \operatorname{rank}(\mathbf{X}_{<n>}) \} \leq R \tag{4.28}$$

Let k denote the argument index of the n-unfolding of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with maximum n-rank, that is

$$k = \arg\max_{n=1,\ldots,N} \{ \operatorname{rank}(\mathbf{X}_{<n>}) \} \tag{4.29}$$

and consider now the subsequent k-unfolding of $\boldsymbol{\mathcal{X}}$,

$$\mathbf{X}_{<k>} = \underbrace{(\mathbf{A}^{(k)} \circledast \cdots \circledast \mathbf{A}^{(1)})}_{\underline{\mathbf{A}}_1 \in \mathbb{R}^{I_1 \cdots I_k \times R}} \mathbf{D} \underbrace{(\mathbf{A}^{(N)} \circledast \cdots \circledast \mathbf{A}^{(k+1)})^T}_{\underline{\mathbf{A}}_2 \in \mathbb{R}^{I_{k+1} \cdots I_N \times R}} \tag{4.30}$$

From Sylvester's rank inequality [113], for any matrix triplet $\mathbf{A} \in \mathbb{R}^{N \times M}$, $\mathbf{B} \in \mathbb{R}^{M \times P}$, $\mathbf{C} \in$

$\mathbb{R}^{P\times K}$, the following holds

$$\text{rank}(\mathbf{ABC}) \geq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) + \text{rank}(\mathbf{C}) - M - P \tag{4.31}$$

Hence, from (4.30), the next result is obtained

$$\text{rank}(\mathbf{X}_{<k>}) \geq R_1 + R + R_2 - R - R \tag{4.32}$$

where $R_1 = \text{rank}(\underline{\mathbf{A}}_1)$, $R_2 = \text{rank}(\underline{\mathbf{A}}_2)$. Combining the above result with that in (4.28), yields

$$R_1 + R_2 - R \leq \text{rank}(\mathbf{X}_{<k>}) \leq R \tag{4.33}$$

whereby, for the equality $R_1 + R_2 - R = R$ to hold, then $\text{rank}(\mathbf{X}_{<k>}) = R$. Consequently, $R_1 = R_2 = R$ is required. Generically, for matrices $\mathbf{A} \in \mathbb{R}^{I\times R}$, and $\mathbf{B} \in \mathbb{R}^{J\times R}$, the following identity holds [65, 114]

$$\text{rank}(\mathbf{A} \circledast \mathbf{B}) = \min\{IJ, R\} \tag{4.34}$$

Remark 4.4. The same terminology as in [65, 114] is adopted. Writing that, generically, $\text{rank}(\mathbf{A}\circledast \mathbf{B}) = \min\{IJ, R\}$, for $\mathbf{A} \in \mathbb{R}^{I\times R}, \mathbf{B} \in \mathbb{R}^{J\times R}$, indicates that the property should hold with probability one for matrices $\mathbf{A}$, $\mathbf{B}$, drawn from a continuous probability density function over their respective subspaces.

As a result, for generic $\underline{\mathbf{A}}_1$ and $\underline{\mathbf{A}}_2$,

$$\begin{aligned} \text{rank}(\underline{\mathbf{A}}_1) &= \min\left\{\prod_{i=1}^{k} I_i, R\right\} \\ \text{rank}(\underline{\mathbf{A}}_2) &= \min\left\{\prod_{i=k+1}^{N} I_i, R\right\} \end{aligned} \tag{4.35}$$

This implies, in turn, that $R_1 = R_2 = R$ is equivalent to satisfying $R \leq \prod_{i=1}^{k} I_i$ and $R \leq \prod_{i=k+1}^{N} I_i$ simultaneously, that is

$$R \leq \min\left\{\prod_{i=1}^{k} I_i, \prod_{i=k+1}^{N} I_i\right\} \tag{4.36}$$

In other words, if a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, admits a CPD and if (4.36) holds, then $R = \text{rank}(\mathbf{X}_{<k>})$, where $k = \arg\max_n\{\text{rank}(\mathbf{X}_{<n>})\}$. Moreover, notice that the strict inequality

$$\text{rank}(\mathbf{X}_{<k>}) < \left\{\prod_{i=1}^{k} I_i, \prod_{i=k+1}^{N} I_i\right\} \tag{4.37}$$

implies

$$R < \min\left\{\prod_{i=1}^{k} I_i, \prod_{i=k+1}^{N} I_i\right\} \tag{4.38}$$

and, as a result, (4.36) is satisfied. Thus, if $\mathbf{X}_{<k>}$ is rank-deficient (4.38), generically it follows

that $\text{rank}(\mathbf{X}_{<k>}) = R$.

Next, it is shown that it is possible to find a value r such that $\text{rank}(\mathbf{X}_{<k>}) \leq r = R$. This is achieved by considering permutations of the indices of the original tensor $\boldsymbol{\mathcal{X}}$.

Definition 4.3. *For a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its N indices admit $N!$ permutations, denoted by $\rho = 1, \ldots, N!$, with the corresponding permuted tensor, $\boldsymbol{\mathcal{X}}^{\rho} \in \mathbb{R}^{[I_1 \times I_2 \times \cdots \times I_N]^{\rho}}$. The permuted tensor has the same elements as $\boldsymbol{\mathcal{X}}$, but with their indices permuted according to a permutation scheme ρ. Therefore, $\boldsymbol{\mathcal{X}}^{\rho}$ is referred to as a **permutation** of $\boldsymbol{\mathcal{X}}$.*

For example, consider a 3-rd order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Based on its indices $\{I_1, I_2, I_3\}$, we can define up to $\rho = 1, \ldots, 3! = 6$ permutations, which include $\boldsymbol{\mathcal{X}}^1 \in \mathbb{R}^{[I_1 \times I_2 \times I_3]^1} \equiv \boldsymbol{\mathcal{X}}^1 \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, or $\boldsymbol{\mathcal{X}}^2 \in \mathbb{R}^{[I_1 \times I_2 \times I_3]^2} \equiv \boldsymbol{\mathcal{X}}^2 \in \mathbb{R}^{I_1 \times I_3 \times I_2}$ all the way to $\boldsymbol{\mathcal{X}}^6 \in \mathbb{R}^{[I_1 \times I_2 \times I_3]^6} \equiv \boldsymbol{\mathcal{X}}^6 \in \mathbb{R}^{I_3 \times I_2 \times I_1}$.

Definition 4.4. *The argument index of the n-unfolding for a permutation ρ which exhibits a maximum n-rank is denoted by k_ρ, that is*

$$k_\rho = \underset{n=1,\ldots,N}{\arg\max} \{ \text{rank}(\mathbf{X}^{\rho}_{<n>}) \}, \ \rho = 1, \ldots, N! \tag{4.39}$$

*By continuity, a **maximum n-rank permutation** (or **maximal permutation**, for brevity) of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\boldsymbol{\mathcal{X}}^{\rho *}$, where ρ^* satisfies*

$$\rho^* = \underset{\rho=1,\ldots,N!}{\arg\max} \, k_\rho \tag{4.40}$$

Remark 4.5. A tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, may admit multiple maximal permutations.

Remark 4.6. If a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, admits a CPD of rank R, then any permutation of $\boldsymbol{\mathcal{X}}$, given by $\boldsymbol{\mathcal{X}}^{\rho} \in \mathbb{R}^{[I_1 \times I_2 \times \cdots \times I_N]^{\rho}}$, admits a CPD of rank R which shares the same factors with the CPD of $\boldsymbol{\mathcal{X}}$, but rearranged following the permutation scheme of the indices. In other words, if

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{D}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!] \tag{4.41}$$

then,

$$\boldsymbol{\mathcal{X}}^{\rho} = [\![\mathbf{D}; [\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]^{\rho}]\!] \tag{4.42}$$

and $\text{rank}(\boldsymbol{\mathcal{X}}) = \text{rank}(\boldsymbol{\mathcal{X}}^{\rho}) = R$. Note that, upon defining $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ and $\mathcal{I}^{\rho *} = \mathcal{J} = \{J_1, J_2, \ldots, J_N\}$, the maximal permutation, ρ^*, can be interpreted as the bijection $\rho^* : \mathcal{I} \mapsto \mathcal{J}$.

The above findings can be summarized as follows.

Theorem 4.4. If a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, admits a CPD of rank R, its maximal permutation, $\boldsymbol{\mathcal{X}}^{\rho *} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, also admits a CPD of rank R. Hence, R is bounded from below by the rank of the k-unfolding, $\mathbf{X}^{\rho *}_{<k>}$, with $k = \arg\max_n \text{rank}(\mathbf{X}^{\rho *}_{<n>})$, that is

$$\text{rank}(\mathbf{X}^{\rho *}_{<k>}) \leq R \tag{4.43}$$

Moreover, if $\mathbf{X}_{<k>}$ is rank-deficient, i.e. if

$$\text{rank}(\mathbf{X}^{\rho*}_{<k>}) < \min\left\{\prod_{i=1}^{k} J_i, \prod_{i=k+1}^{N} J_i\right\} \tag{4.44}$$

then, generically, $\text{rank}(\mathbf{X}^{\rho*}_{<k>}) = \text{rank}(\boldsymbol{\mathcal{X}}^{\rho*}) = R$.

Proof. By (4.28) and Remark 4.6, $\text{rank}(\mathbf{X}^{\rho*}_{<k>}) \leq R$, while the rest of Theorem 4.4 follows from (4.33)–(4.37). □

4.7 Rank Detectability

Theorem 4.4 implies that it is generically possible to *detect* (compute) the rank of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ simply by computing the maximum n-rank of its maximal permutation, provided that (4.44) holds, in which case, the rank of the tensor is automatically given as the rank of the unfolding. The problem then boils down to finding the maximal permutation ρ^*, which is generically given as the solution to,

$$\underset{\mathcal{S}_1, \mathcal{S}_2}{\arg\max} \min\left\{\prod_{i \in \mathcal{S}_1} I_i, \prod_{i \in \mathcal{S}_2} I_i\right\} \tag{4.45}$$

where $\mathcal{S}_1 \cup \mathcal{S}_2 = \{I_1, \ldots, I_N\}$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \varnothing$, $|\mathcal{S}_1| = N_1$, $|\mathcal{S}_2| = N_2$, and $N_1 + N_2 = N$. The problem in (4.45) is then equivalent to finding the matrix unfolding of the original tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ which is "as square as possible", referred to as **maximally square unfolding**.

The problem in (4.45) can hence be expressed as determining its maximally square unfolding (aspect ratio ≈ 1), that is

$$\underset{\mathcal{S}_1, \mathcal{S}_2}{\arg\min} \left| \frac{\prod_{i \in \mathcal{S}_1} I_i}{\prod_{i \in \mathcal{S}_2} I_i} - 1 \right| \tag{4.46}$$

which is equivalent to

$$\begin{aligned} &\underset{\mathcal{S}_1, \mathcal{S}_2}{\arg\min} \left| \log\left(\frac{\prod_{i \in \mathcal{S}_1} I_i}{\prod_{i \in \mathcal{S}_2} I_i}\right) \right| \\ =&\underset{\mathcal{S}_1, \mathcal{S}_2}{\arg\min} \left| \log\Big(\prod_{i \in \mathcal{S}_1} I_i\Big) - \log\Big(\prod_{i \in \mathcal{S}_1} I_i\Big) \right| \\ =&\underset{\mathcal{S}_1, \mathcal{S}_2}{\arg\min} \left| \sum_{i \in \mathcal{S}_1} \log(I_i) - \sum_{i \in \mathcal{S}_2} \log(I_i) \right| \end{aligned} \tag{4.47}$$

The above is a variant of the well-known *number partitioning problem*, and has been extensively studied in the literature [115, 116, 117]. Although the partitioning problem is NP-complete, it can be solved efficiently in practice via dynamic programming. The optimal sets, $\mathcal{S}_1$ and $\mathcal{S}_2$, then immediately give the maximal unfolding, $\mathbf{X}^{\rho*}_{<k>}$.

The practical implications of Theorem 4.4 are two-fold:

i. Given an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, of rank R, if $R > \text{rank}(\mathbf{X}^{\rho*}_{<k>})$, i.e. if the maximal unfolding $\mathbf{X}^{\rho*}_{<k>}$ is full rank, a search for R may commence from $\text{rank}(\mathbf{X}^{\rho*}_{<k>})$;

ii. Perhaps more importantly, if $\mathbf{X}^{\rho*}_{<k>}$ is rank-deficient, then, generically, $R = \text{rank}(\mathbf{X}^{\rho*}_{<k>})$,

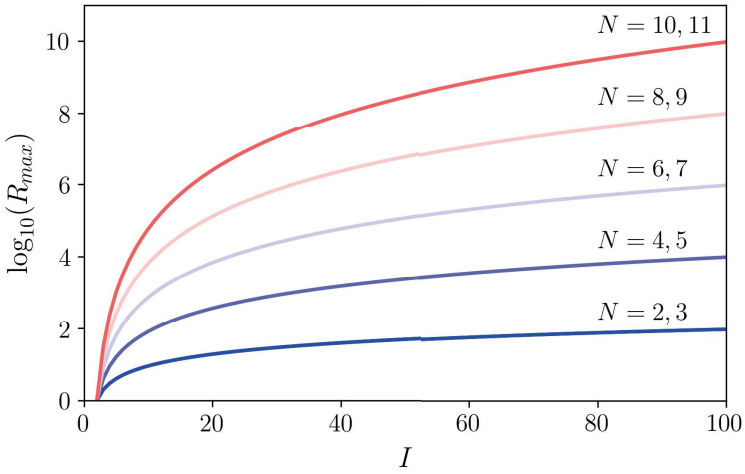


Figure 4.3: Maximum detectable rank for N-th order tensors, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with $I_n = I$ for $n = 1, \ldots, N$, where $N = 1, \ldots, 11$, and increasing values of I.

i.e. the rank is *detected*, and no search as in (i) is required.

The maximum detectable rank is therefore equal to the highest attainable rank by a rank-deficient maximally square unfolding, that is

$$R_{max} = \min \left\{ \prod_{i \in \mathcal{S}_1} I_i - 1, \prod_{i \in \mathcal{S}_2} I_i - 1 \right\} \tag{4.48}$$

Figure 4.3 illustrates the range of R in (4.48) for N-th order tensors, whereby equal dimensionality across modes has been assumed for simplicity, such that $R_{max} = I^{\lfloor \frac{N}{2} \rfloor} - 1$ and $I_i = I$ for all $i = 1, \ldots, N$. For such configuration, the maximum detectable rank is exponential in $\lfloor \frac{N}{2} \rfloor$. Consider for example an N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{20 \times 20 \times \cdots \times 20}$. If $N = 2, 3$, then $R_{max} = 19$, while if $N = 4, 5$, then $R_{max} = 399$ and $R_{max} = 7999$ if $N = 6, 7$. This means that, generically, the tensor rank, R, can immediately be found if $\text{rank}(\mathbf{X}^{\rho*}_{<k>}) \leq R_{max}$, i.e. if the maximal unfolding is rank-deficient, which removes the need for iterative procedures. The simplicity of such test makes it very useful in practice.

4.8 Example Application: EEG Data

A common biomedical application of tensors is the study of multi-channel electroencephalogram (EEG) in brain science [52], whereby data can be arranged in a multi-way array as time $\times$ frequency $\times$ channels or as time $\times$ channel $\times$ subject. This allows for the use of tensor decompositions, which yield a significantly more interpretable representation of latent features over their flat-view matrix counterparts [18]. The CPD is naturally suited for such problems and it is shown in the next section how the proposed lower bound may be used to make the CPD more efficient, through direct application of Theorem 4.4.

We consider data from 10 different subjects and 8 EEG channels, arranged as a $T\times$ channel $\times$ subject 3-rd order tensor. ($32107 \times 8 \times 10$). Each participant had to focus on balancing on

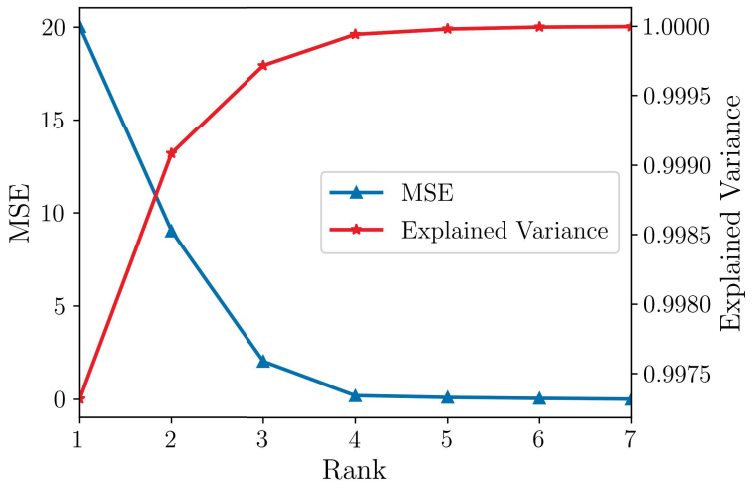


Figure 4.4: Rank estimation of a real-world data tensor (EEG) through (i) subsequent CPD decompositions (blue curve), (ii) Theorem 1 (red curve). The red curve was obtained almost 24 times faster.

a hard surface for one minute, with their eyes closed. The 8 channels of EEG covered frontal, central, parietal and occipital areas of the brain. As is common practice, a bandpass filter 1-95Hz and a notch filter at 50Hz were first applied. CPDs were subsequently performed, and the tensor rank can be estimated to be $R = 4$ from Figure 4.4 (blue curve).

The same result can be obtained via Theorem 4.4. The 3-rd order tensor was first unfolded into a 32107×80 matrix, $\mathbf{M}$. Due to presence of noise, $\mathbf{M}$ is obviously full rank, however, a low rank representation can still be obtained. Upon computing the eigenvalues of $\mathbf{M}^T\mathbf{M} \in \mathbb{R}^{80\times 80}$ we examined the position of the elbow in their cumulative sum (explained variance). If such an elbow occurs at $r \leq 80$ eigenvalues, then r can be taken as the rank of the unfolding matrix, and hence as the rank of the original tensor (red curve in Figure 4.4). This was observed again at $r = 4$, indicating that the rank of $\mathbf{M}$ is 4 (in the absence of noise), which matches the rank of the tensor, as stated in Theorem 4.4. The estimation of the tensor rank for the blue curve (standard way) took 3.35 seconds and 0.14s for the red curve (proposed method), corresponding to almost a 25-fold improvement in computation speed, demonstrating a clear benefit of the proposed lower bound. (CPDs were performed using the freely available software `HOTTBOX` [118]).

Remark 4.7. Such an approach assumes that the considered real-world data is not overly corrupted by noise. This would spread the overall variance among subsequent eigenvectors and eigenvalues, making rank detectability more difficult. However, this would also affect the efficiency of a standard iterative search for R, and that of any estimation method, in general.

4.9 Chapter Summary

Two approaches to ease the computational burden associated to the Canonical Polyadic Decomposition (CPD) have been introduced. Firstly, it has been shown that a CPD of an N-th order tensor can be reliably estimated from a compressed TT-MPS model of the original tensor.

This has been achieved by exploiting the TT-CPD duality, outlined in Lemma 4.1 and Lemma 4.2, which was employed to provide a method that allows converting a tensor in the TT format to the CPD format, as explained in Section 4.3. This approach has been demonstrated to reduce the computational cost of the CPD from $\mathcal{O}(NRI^N)$ to $\mathcal{O}(NIR^3)$, which is particularly suited for higher order tensors.

Secondly, a lower bound on the tensor rank, R, has been proposed with the aim to reduce the search space for R, hence further easing computational costs on the CPD. The desired bound of R has been shown to be equal to the maximum possible matrix rank attainable from any unfolding of the tensor. Determining this so-called maximal unfolding which has been shown to be equivalent to solving the number partitioning problem on the indices of the original tensor, which can be done efficiently through dynamic programming. For a generic tensor, the bound has been demonstrated to be attainable when the maximal unfolding is rank-deficient, thereby providing a simple test for rank detectability, while numerical examples in Figure 4.3 and a practical application in Figure 4.4 illustrate the practical benefits of this result.

Conclusion to Part I

In **Part I**, important insights in major tensor decompositions have been provided. **Chapter 3** drew the analogy between Tensor Decompositions (TDs) and Tensor Networks (TNs), emphasizing that any TD can be interpreted as a TN and can be represented via nodes and edges. This has been used to elucidate fundamental tensor operations, such as unfolding, tensorization, and inner products, as well as to introduce a novel algorithm for tensor contraction based upon the Tensor Train (TT) decomposition, called the Tensor-Train Contraction Product (TTCP). **Chapter 4** specifically focused on the Canonical Polyadic Decomposition (CPD), and proposed means to alleviate its computational burden by employing TTs to compute the CPD of higher order tensors and by introducing a lower bound on the tensor rank. **Chapter 4** utilized the insights from **Chapter 3** to elucidate important concepts, as for example in Figure 4.1. Indeed, the notions from **Chapter 3** are crucial for the understanding of central ideas presented in later sections of this thesis, that deal with the development of tensor-based machine learning algorithms.

Part II

Tensors for Shallow Learning

Chapter 5

Tensor Network Summation

> Beauty is a summation of the parts working together in such a way that nothing needed to be added, taken away or altered.
>
> *Elio Carletti*

Chapter Overview

The following two chapters will focus on machine learning applications for tensors and tensor networks, drawn from important signal processing concepts, involving feature fusion, eigenvalue extraction, and the Support Vector Machine. This chapter expands on the graphical representations of tensors and tensor networks (TNs) presented in **Chapter 3**, by introducing a framework to combine TNs via summation. Despite the obvious benefits that come with their ability to tackle the Curse of Dimensionality through the representation of large-scale tensors via smaller-scale interconnections of their intrinsic features, the current treatment of TNs as stand-alone entities does not take full advantage of their underlying structure and the associated feature localization. To this end, the analogy with feature fusion is exploited to propose a rigorous framework for the combination of TNs. The proposed framework is shown to allow for feature combination of any number of tensors, as long as their TN representation topologies are isomorphic. The properties of TN summation ar formalized by introducing the inverse (subtraction) operator and the identity element. The benefits of the proposed framework are demonstrated through simulations involving multi-class classification of the ETH-80 image dataset, and by introducing a novel approach for eigenvalue extraction of large-scale matrices.

5.1 Introduction

Any Tensor Decomposition (TD) can be considered 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. 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. Minimization of the number of the parameters necessary for a TN representation;
3. A rigorous framework to combine TNs.

This chapter addresses issue (iii), and introduces 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.

The framework for the summation of TNs is achieved by intelligent block arrangement of the corresponding original cores [81]. The fact that the interconnections among the cores in a TN describe the coupling of data structures in the original tensor serves as a basis to explore ways to combine the corresponding individual cores of two TNs with the same topology in order to obtain a new TN, the cores of which carry information jointly present in the individual original cores. This is related to the recently introduced concept of common feature extraction in [109, 119], however, unlike matrices, the proposed framework enables this operation on tensors of any order, with the only condition that the original tensors are represented as isomorphic TNs. General properties of TN summation are defined, clarified through illustrative examples, and found to be analogous to those of standard scalar and matrix addition involving associativiy, commutativity, the identity element, and the inverse operator.

Practical advantages are firstly demonstrated through an image classification application based on the ETH-80 dataset [120], whereby every dataset entry is represented as a TN, and their 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 in terms of classification rates when used in conjunction with standard machine learning classifiers.

Secondly, the TN summation framework is applied to the problem of eigenvalue and eigenvector estimation for very large scale matrices. This is achieved through the Rayleigh Quotient (RQ) by applying relevant TDs and by representing the related cost function in tensor format [60]. A deflation scheme based on Hotelling's deflation [121] is implemented via the proposed framework of TN summation through the inverse operator (subtraction). This allows to estimate the largest K eigenvalues and associated eigenvectors of large-scale matrices in an iterative fashion. Results show that eigenvalues and eigenvectors can be reliably computed in such way.

5.2 Summation Framework

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

1. The physical modes of the summands are of equal dimensions;
2. The same topology.

In the following, relevant definitions, the proposed framework, and related mathematical formalism are provided. Also, in reference to Figure 2.6 in **Chapter 2**, when representing tensors and TNs as nodes and edges, "dangling" edges will be referred to as *physical modes*, as these represented the actual dimensions of the underlying tensor, while the rest will be referred to as *contraction modes*.

Definition 5.1. *A **block tensor** is a tensor that is arranged into sub-tensors called **blocks**, that is, its entries are tensors of the same order but not necessarily of the same dimensionality.*

Definition 5.2. *The **superdiagonal** of a block tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the collection of entries $\boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_n)$, where $i_1 = i_2 = \cdots = i_n$.*

With the above, the method of TN summation can be summarized in the following. The TN representation of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by underline boldface uppercase letters, $\underline{\mathbf{X}}$.

Sum of TNs Framework. Consider two tensors $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_1 \times \cdots \times I_N}$ 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 $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}}$ 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.

5.2.1 Analysis and Examples

Proposition 5.1. The proposed framework for a sum of TNs is valid for chains of matrices.

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

$$
\begin{aligned}
\mathbf{Z} &= \mathbf{C}_1\mathbf{C}_2 \cdots \mathbf{C}_N \\
&= \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2 \end{bmatrix} \cdots \begin{bmatrix} \mathbf{A}_{N-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}_N \\ \mathbf{B}_N \end{bmatrix} \\
&= \mathbf{A}_1\mathbf{A}_2 \cdots \mathbf{A}_n + \mathbf{B}_1\mathbf{B}_2 \cdots \mathbf{B}_n \\
&= \mathbf{X} + \mathbf{Y}
\end{aligned}
\tag{5.1}
$$

□

Figure 5.1 shows a graphical illustration of Proposition 5.1.

Remark 5.1. The above result for matrices is well-known, and serves here as a specific and straightforward intuition behind the general concept of TN summation.

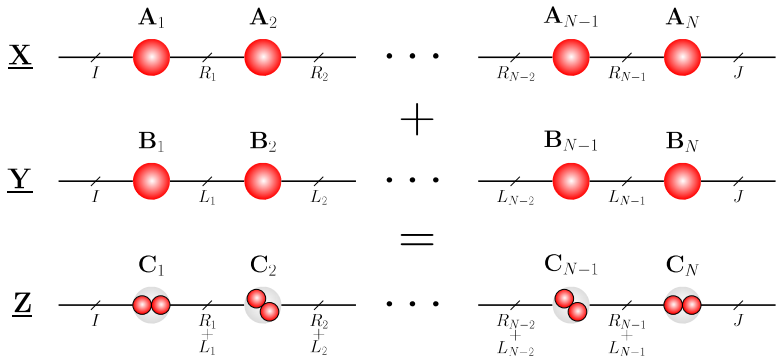


Figure 5.1: Topology preservation for the TN summation operator, illustrated for a sum of matrices $\mathbf{X}$ and $\mathbf{Y}$, expressed as a matrix chain. The matrices (nodes) $\mathbf{C}_n$ are composed of the matrices $\mathbf{A}_n$ and $\mathbf{B}_n$, which are arranged according to the sum of TNs framework, either through (i) concatenation or (ii) block-diagonal arrangement.

Proposition 5.2. The TN summation operator is valid for any tensor expressed in the TKD format.

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. 5.2 shows the tensors $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format, expressed as

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \boldsymbol{\mathcal{G}}_x \times_1 \mathbf{A}_x \times_2 \mathbf{B}_x \times_3 \mathbf{C}_x \\ \boldsymbol{\mathcal{Y}} &= \boldsymbol{\mathcal{G}}_y \times_1 \mathbf{A}_y \times_2 \mathbf{B}_y \times_3 \mathbf{C}_y \end{aligned} \tag{5.2}$$

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 (see Figure 5.2), and the so generated tensor, $\boldsymbol{\mathcal{Z}}$, can be described by

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{G}}_z \times_1 \mathbf{A}_z \times_2 \mathbf{B}_z \times_3 \mathbf{C}_z \tag{5.3}$$

We next show that $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}}$, based on the mode-1 unfolding, however, note that the same procedure can be applied to any mode. Define $\mathbf{A}_z = \begin{bmatrix} \mathbf{A}_x & \mathbf{A}_y \end{bmatrix}$, and $\boldsymbol{\mathcal{G}}_z$ as an arrangement of $\boldsymbol{\mathcal{G}}_x$ and $\boldsymbol{\mathcal{G}}_y$ along the superdiagonal of $\boldsymbol{\mathcal{G}}_z$, and consider

$$\begin{aligned} \mathbf{K}_1 &= (\mathbf{C}_x \otimes \mathbf{B}_x)^T \\ \mathbf{K}_2 &= (\mathbf{C}_y \otimes \mathbf{B}_y)^T \end{aligned} \tag{5.4}$$

Upon performing the mode-1 unfolding of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$, and combining the matrices $\mathbf{X}_{(1)}, \mathbf{Y}_{(1)}$ in their TN formats, then from Proposition 5.1 the resulting matrix, $\mathbf{Z}_*$, can be expressed as

$$\mathbf{Z}_* = \begin{bmatrix} \mathbf{A}_x & \mathbf{A}_y \end{bmatrix} \begin{bmatrix} \mathbf{G}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{y(1)} \end{bmatrix} \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix}^T \tag{5.5}$$

Therefore, in order to prove Proposition 5.2 it is sufficient to show that $\mathbf{Z}_{(1)} = \mathbf{Z}_*$, where $\mathbf{Z}_{(1)}$ is the mode-1 unfolding of $\mathcal{Z}$, represented as in (5.3). To this end, consider

$$\begin{aligned}\mathbf{Z}_{(1)} &= \mathbf{A}_z \mathbf{G}_{z(1)} (\mathbf{C}_z \otimes \mathbf{B}_z)^T \\ &= \begin{bmatrix}\mathbf{A}_x & \mathbf{A}_y\end{bmatrix} \mathbf{G}_{z(1)} \left(\begin{bmatrix}\mathbf{C}_x & \mathbf{C}_y\end{bmatrix} \otimes \begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} \right)^T\end{aligned} \tag{5.6}$$

For convenience, denote $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{G}}_x, \boldsymbol{\mathcal{G}}_y \in \mathbb{R}^{2\times 2\times 2}$, and define $\boldsymbol{\mathcal{G}}_\alpha(:,:,j) = \hat{\mathbf{G}}_\alpha(j)$, where $\alpha \in \{x, y\}$. Hence,

$$\mathbf{G}_{z(1)} = \begin{bmatrix} \hat{\mathbf{G}}_x(1) & \mathbf{0} & \hat{\mathbf{G}}_x(2) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y(1) & \mathbf{0} & \hat{\mathbf{G}}_y(2) \end{bmatrix} \tag{5.7}$$

Without loss of generality, assume the concrete values

$$\mathbf{C}_x = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \qquad \mathbf{C}_y = \begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix} \tag{5.8}$$

to give

$$\begin{aligned}&\begin{bmatrix}\mathbf{C}_x & \mathbf{C}_y\end{bmatrix} \otimes \begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} = \\ &= \begin{bmatrix} 1\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 2\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 3\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 4\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} \\ 5\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 6\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 7\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} & 8\begin{bmatrix}\mathbf{B}_x & \mathbf{B}_y\end{bmatrix} \end{bmatrix} \\ &= \mathbf{U}\end{aligned} \tag{5.9}$$

Upon substituting $\mathbf{G}_{z(1)}$ and $\mathbf{U}$ into (5.6) and making use of the sparse nature of (5.7), we arrive at

$$\begin{aligned}\mathbf{Z}_{(1)} &= \mathbf{A}_z \mathbf{G}_{z(1)} \mathbf{U}^T \\ &= \begin{bmatrix}\mathbf{A}_x & \mathbf{A}_y\end{bmatrix} \begin{bmatrix} \hat{\mathbf{G}}_x(1) & \hat{\mathbf{G}}_x(2) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y(1) & \hat{\mathbf{G}}_y(2) \end{bmatrix} \begin{bmatrix} 1\mathbf{B}_x^T & 5\mathbf{B}_x^T \\ 2\mathbf{B}_x^T & 6\mathbf{B}_x^T \\ 3\mathbf{B}_y^T & 7\mathbf{B}_y^T \\ 4\mathbf{B}_y^T & 8\mathbf{B}_y^T \end{bmatrix} \\ &= \begin{bmatrix}\mathbf{A}_x & \mathbf{A}_y\end{bmatrix} \begin{bmatrix} \mathbf{G}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{y(1)} \end{bmatrix} \begin{bmatrix} \mathbf{K}_1^T \\ \mathbf{K}_2^T \end{bmatrix} \\ &= \begin{bmatrix}\mathbf{A}_x & \mathbf{A}_y\end{bmatrix} \begin{bmatrix} \mathbf{G}_{x(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{y(1)} \end{bmatrix} \begin{bmatrix}\mathbf{K}_1 & \mathbf{K}_2\end{bmatrix}^T = \mathbf{Z}_*\end{aligned} \tag{5.10}$$

□

Proposition 5.3. The TN summation framework is valid for tensor trains.

Proof. Without loss of generality we consider two 3-rd order tensors of low dimensionality (to

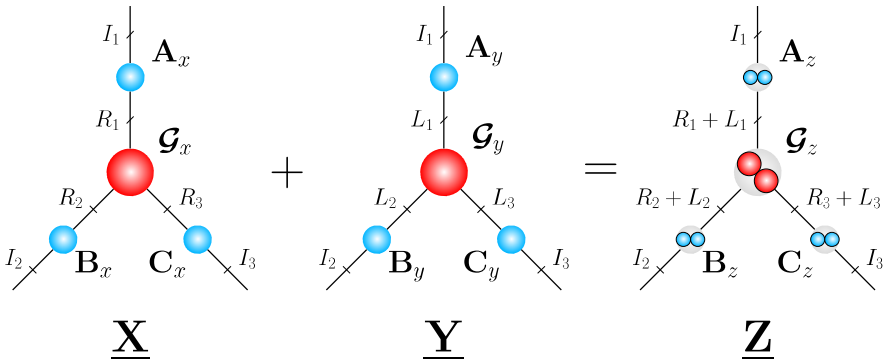


Figure 5.2: Topology of a sum of tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ in the TKD format.

make the equations more tractable) $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{2\times 2\times 2}$

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \left[\begin{array}{cc|cc} x_1 & x_2 & x_5 & x_6 \\ x_3 & x_4 & x_7 & x_8 \end{array}\right] \\ \boldsymbol{\mathcal{Y}} &= \left[\begin{array}{cc|cc} y_1 & y_2 & y_5 & y_6 \\ y_3 & y_4 & y_7 & y_8 \end{array}\right] \end{aligned} \tag{5.11}$$

where the vertical bars represent separations among the tensors' frontal slices. Then suppose we decompose them as TTs

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \mathbf{A}_x \times_2^1 \boldsymbol{\mathcal{G}}_x \times_3^1 \mathbf{B}_x^T \\ \boldsymbol{\mathcal{Y}} &= \mathbf{A}_y \times_2^1 \boldsymbol{\mathcal{G}}_y \times_3^1 \mathbf{B}_y^T \end{aligned} \tag{5.12}$$

The transpose operator on matrices $\mathbf{B}_{\{x,y\}}$ is preferred for representation purposes, as we can manipulate $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ in

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \mathbf{A}_x \times_2^1 (\boldsymbol{\mathcal{G}}_x \times_2 \mathbf{I}) \times_3^1 \mathbf{B}_x^T \\ &= \boldsymbol{\mathcal{G}}_x \times_1 \mathbf{A}_x \times_2 \mathbf{I} \times_3 \mathbf{B}_x \\ \\ \boldsymbol{\mathcal{Y}} &= \mathbf{A}_y \times_2^1 (\boldsymbol{\mathcal{G}}_y \times_2 \mathbf{I}) \times_3^1 \mathbf{B}_y^T \\ &= \boldsymbol{\mathcal{G}}_y \times_1 \mathbf{A}_y \times_2 \mathbf{I} \times_3 \mathbf{B}_y \end{aligned} \tag{5.13}$$

where $\mathbf{I}$ is the identity matrix. We have $\boldsymbol{\mathcal{G}}_{\{x,y\}} \in \mathbb{R}^{2\times 2\times 2}$, $\mathbf{A}_{\{x,y\}}, \mathbf{B}_{\{x,y\}}, \mathbf{I} \in \mathbb{R}^{2\times 2}$. From Proposition 5.2 we know that

$$\begin{aligned} \boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}} &= \boldsymbol{\mathcal{G}}_* \times_1 \begin{bmatrix} \mathbf{A}_x & \mathbf{A}_y \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{I} & \mathbf{I} \end{bmatrix} \times_3 \begin{bmatrix} \mathbf{B}_x & \mathbf{B}_y \end{bmatrix} \\ &= \boldsymbol{\mathcal{G}}_* \times_1 \mathbf{A}_z \times_2 \mathbf{K}_z \times_3 \mathbf{B}_z \end{aligned} \tag{5.14}$$

where $\begin{bmatrix} \mathbf{A}_x & \mathbf{A}_y \end{bmatrix} = \mathbf{A}_z \in \mathbb{R}^{2\times 4}$, $\begin{bmatrix} \mathbf{I} & \mathbf{I} \end{bmatrix} = \mathbf{K}_z \in \mathbb{R}^{2\times 4}$, $\begin{bmatrix} \mathbf{B}_x & \mathbf{B}_y \end{bmatrix} = \mathbf{B}_z \in \mathbb{R}^{2\times 4}$, and $\boldsymbol{\mathcal{G}}_* \in \mathbb{R}^{4\times 4\times 4}$ was formed by concatenating $\boldsymbol{\mathcal{G}}_x, \boldsymbol{\mathcal{G}}_y$ along the superdiagonal.

We want to show that $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{Z}}_*$, where

$$\boldsymbol{\mathcal{Z}}_* = \boldsymbol{\mathcal{G}}_z \times_1 \mathbf{A}_z \times_3 \mathbf{B}_z \tag{5.15}$$

and $\mathcal{G}_z \in \mathbb{R}^{4\times 2\times 4}$, obtained by stacking $\mathcal{G}_x$ and $\mathcal{G}_y$ s.t. the dimensionality of all modes except for the physical ones is increased, following the rules of the TN sum conjecture. Therefore all is required is to show that

$$\mathcal{G}_z = \mathcal{G}_* \times_2 \mathbf{K}_z \tag{5.16}$$

Which is equivalent to showing that

$$\mathbf{G}_{z(2)} = \mathbf{K}_z \mathbf{G}_{*(2)} \tag{5.17}$$

Define $\mathcal{G}_\alpha(:,:,j) = \hat{\mathbf{G}}_\alpha(j)$, where $\alpha \in \{x, y\}$. Hence, by direct unfolding

$$\mathbf{G}_{*(2)} = \begin{bmatrix} \hat{\mathbf{G}}_x(1)^T & \mathbf{0} & \hat{\mathbf{G}}_x(2)^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y(1)^T & \mathbf{0} & \hat{\mathbf{G}}_y(2)^T \end{bmatrix} \tag{5.18}$$

By substituting (5.11) into (5.17) and by recalling that $\mathbf{K}_z = \begin{bmatrix} \mathbf{I} & \mathbf{I} \end{bmatrix}$, after carrying out the necessary algebraic steps, we conclude that

$$\mathbf{K}_z \mathbf{G}_{*(2)} = \begin{bmatrix} \hat{\mathbf{G}}_x(1) & \mathbf{0} & \hat{\mathbf{G}}_x(2) & \mathbf{0} & \mathbf{0} & \hat{\mathbf{G}}_y(1) & \mathbf{0} & \hat{\mathbf{G}}_y(2) \end{bmatrix} \tag{5.19}$$

but the last equality is $\mathbf{G}_{z(2)}$, obtained by direct unfolding. Therefore the proposition is proven. □

In other words, Proposition 5.3 states that given tensors in TT format, for example $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times I_3}$ expressed as

$$\begin{aligned} \mathcal{X} &= \mathbf{A}_x \times_2^1 \mathcal{G}_x \times_3^1 \mathbf{B}_x \\ \mathcal{Y} &= \mathbf{A}_y \times_2^1 \mathcal{G}_y \times_3^1 \mathbf{B}_y \end{aligned} \tag{5.20}$$

where $\mathbf{A}_x \in \mathbb{R}^{I_1\times R_1}$, $\mathcal{G}_x \in \mathbb{R}^{R_1\times I_2\times R_2}$, $\mathbf{B}_x \in \mathbb{R}^{R_2\times I_3}$, and $\mathbf{B}_x \in \mathbb{R}^{I_1\times L_1}$, $\mathcal{G}_y \in \mathbb{R}^{L_1\times I_2\times L_2}$, $\mathbf{B}_y \in \mathbb{R}^{L_2\times I_3}$, then the sum $\mathcal{X}+\mathcal{Y} = \mathcal{Z} \in \mathbb{R}^{I_1\times I_2\times I_3}$ can be visualized as a new TN $\underline{\mathbf{Z}}$ expressed as

$$\mathcal{Z} = \mathbf{A}_z \times_2^1 \mathcal{G}_z \times_3^1 \mathbf{C}_z \tag{5.21}$$

where

$$\begin{aligned} \mathbf{A}_z &= \begin{bmatrix} \mathbf{A}_x & \mathbf{A}_y \end{bmatrix} \in \mathbb{R}^{I_1\times(R_1+L_1)} \\ \mathbf{B}_z &= \begin{bmatrix} \mathbf{B}_x & \mathbf{B}_y \end{bmatrix} \in \mathbb{R}^{I_3\times(R_2+L_2)} \end{aligned} \tag{5.22}$$

and $\mathcal{G}_z \in \mathbb{R}^{(R_1+L_1)\times I_2\times(R_2+L_2)}$. An illustration of the TN sum for two 3-rd order tensors expressed as TTs is shown in Figure 5.3.

5.2.2 General Properties

In a way analogous to scalar and matrix summation, TN summation presents properties of associativity and commutativity, as well as identity and inverse operators. The identity TN $\underline{\mathbf{O}}$ is composed of cores $\mathcal{O}^{(n)}$ consisting only of zero-entries. In other words, $\underline{\mathbf{O}}$ is generated from the conceptual TD of a tensor $\mathcal{O} \in \mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ with elements $o_{i_1 i_2 \ldots i_N} = 0$. Figure 5.4

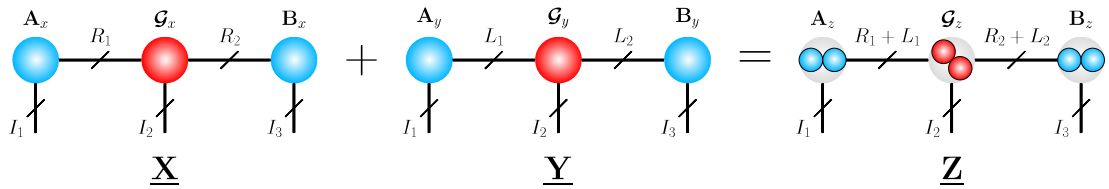


Figure 5.3: Graphical representation of the sum of tensors $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$ expressed in the TT/MPS format.

graphically represents the commutativity and the inverse operator properties, summarized in TABLE 5.1. Further properties of associativity and distributivity involving scalar multiplication can also be defined, and these are given in TABLE 5.2.

Table 5.1: Summation

Associativity	$(\underline{\mathbf{A}} + \underline{\mathbf{B}}) + \underline{\mathbf{C}} = \underline{\mathbf{A}} + (\underline{\mathbf{B}} + \underline{\mathbf{C}})$
Commutativity	$\underline{\mathbf{A}} + \underline{\mathbf{B}} = \underline{\mathbf{B}} + \underline{\mathbf{A}}$
Identity	$\underline{\mathbf{A}} + \underline{\mathbf{O}} = \underline{\mathbf{A}}$
Inverse	$\underline{\mathbf{A}} + (-\underline{\mathbf{A}}) = (-\underline{\mathbf{A}}) + \underline{\mathbf{A}} = \underline{\mathbf{O}}$ $\underline{\mathbf{A}} - \underline{\mathbf{B}} = \underline{\mathbf{A}} + (-\underline{\mathbf{B}})$

Table 5.2: Scalar multiplication

Associativity	$(a_1 a_2)\underline{\mathbf{A}} = a_1(a_2\underline{\mathbf{A}})$
Distributivity	$a_1(\underline{\mathbf{A}} + \underline{\mathbf{B}}) = a_1\underline{\mathbf{A}} + a_1\underline{\mathbf{B}}$ $(a_1 + a_2)\underline{\mathbf{A}} = a_1\underline{\mathbf{A}} + a_2\underline{\mathbf{A}}$

Remark 5.2. TN summation properties require that the summed TNs conform to the assumptions listed at the beginning of Section 5.2.

5.3 Feature Fusion via TN Summation

The TN summation operator is validated through a practical example of feature fusion in image classification. The benchmark ETH-80 dataset is considered, which consists of 3280 images, composed of 8 classes, with 10 objects per class and 41 images per object (sample images shown in Figure 5.5). For the following simulations, the images were rescaled to 32×32 pixels. Given the RGB format of the considered images, the dataset is conveniently represented through 3-rd order tensors $\boldsymbol{\mathcal{X}}_m \in \mathbb{R}^{32\times32\times3}, m = 1, \ldots, M$ (where $M = 3280$). Simulations were performed in the freely available software package for tensors: `HOTTBOX` [118].

Figure 5.6 shows that for each image, $\boldsymbol{\mathcal{X}}_m$, in the training set, upon performing TKD, the size of the corresponding core tensors $\boldsymbol{\mathcal{G}}_m \in \mathbb{R}^{R_1\times R_2\times R_3}$ was set to $R_1 = R_2 = R_3 = 3$. These values were empirically found to offer a good approximation of the original images, while at the same time maintaining a small size core tensor. To emphasize the features of each

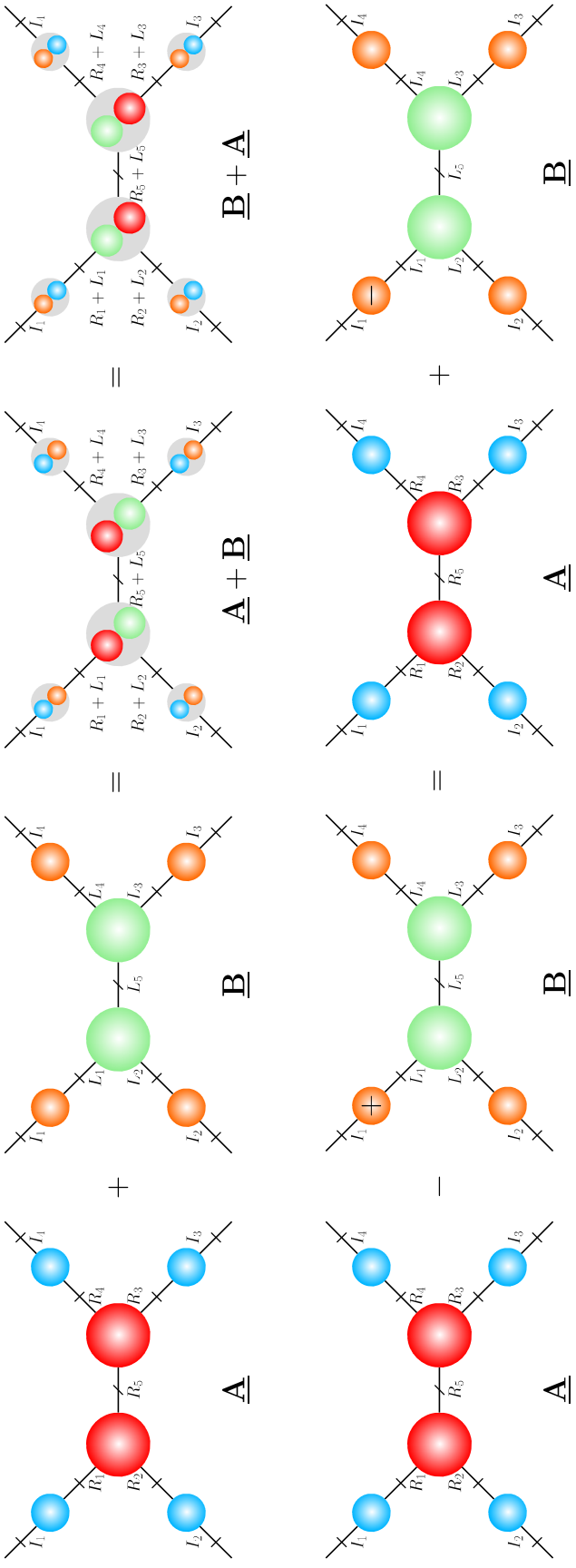


Figure 5.4: Graphical representation of commutativity and inverse operator properties. Top: commutativity. Notice how the order of arrangement of corresponding cores is preserved. Bottom: inverse operator (i.e. TN subtraction). Negation of is achieved simply by negating all the elements of any of its cores.



Figure 5.5: Sample images from the ETH-80 dataset.

image without affecting the TKD approximation, each core tensor $\mathcal{G}_m$ was normalized to unit norm and the factor matrices $\{\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m\}$ were scaled by $\eta_m^{1/3}$, where $\eta_m = ||\mathcal{G}_m||$. Through TN summation, 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 $\mathcal{G}_c$. Because the goal was to examine the mixture of actual features, focus was given 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 (this operator is referred to as tSVD$(\cdot)$), to yield matrices $\{\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t\}$. Finally, for each image, $\mathcal{X}_m$, a new core tensor was computed as

$$\mathring{\mathcal{G}}_m = \mathcal{X}_m \times_1 \mathbf{A}_t^T \times_2 \mathbf{B}_t^T \times_3 \mathbf{C}_t^T \tag{5.23}$$

The vectorized versions $\text{vec}(\mathring{\mathcal{G}}_m), m = 1, \ldots, M$, were then fed to standard machine learning classifiers for performance evaluation. During the testing stage, for each new element $\mathcal{X}_*$, $\mathring{\mathcal{G}}_*$ was computed via (5.23) using the $\{\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t\}$ matrices obtained during training, and subsequently vectorized.

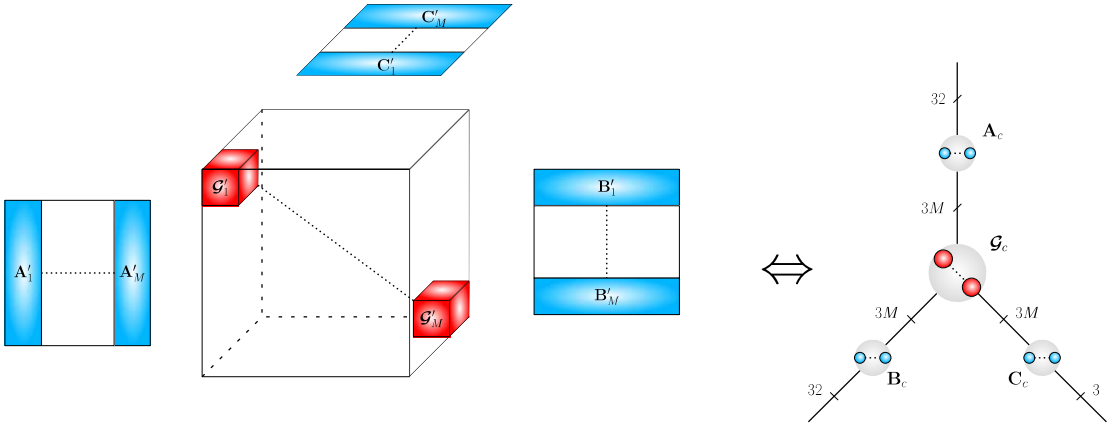


Figure 5.6: An example of TN summation for the TKD representations of the images in the training set, where $\{\mathbf{A}'_m, \mathbf{B}'_m, \mathbf{C}'_m\} = \eta_m^{1/3}\{\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m\}$ and $\mathcal{G}'_m = \mathcal{G}_m/\eta_m$ *Left*: A TKD arrangement of matrices and 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.

Remark 5.3. The physical meaning of equation (5.23) is a projection of the raw images onto the feature space that is common to the whole dataset, which greatly reduces the risk of overfitting,

hence ensuring that $\mathring{\mathcal{G}}_m$ can be used for classification purposes.

Algorithm 7. Sum of TNs for image classification

Input: Dataset $\{\mathcal{X}_m\}_{m=1}^M$, core size $\{R_1, R_2, R_3\}$
Initialize $\{\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c\}$
for each element m in dataset **do**
  $\mathcal{X}_m \leftarrow \mathcal{G}_m \times_1 \times \mathbf{A}_m \times_2 \mathbf{B}_m \times_3 \mathbf{C}_m$
  $\eta_m \leftarrow ||\mathcal{G}_m||_F$
  $\mathbf{A}_c \leftarrow \begin{bmatrix} \mathbf{A}_c & \eta_m^{1/3}\mathbf{A}_m \end{bmatrix}$
  $\mathbf{B}_c \leftarrow \begin{bmatrix} \mathbf{B}_c & \eta_m^{1/3}\mathbf{B}_m \end{bmatrix}$
  $\mathbf{C}_c \leftarrow \begin{bmatrix} \mathbf{C}_c & \eta_m^{1/3}\mathbf{C}_m \end{bmatrix}$
end for
$\mathbf{A}_t \leftarrow \text{tSVD}(\mathbf{A}_c, R_1)$
$\mathbf{B}_t \leftarrow \text{tSVD}(\mathbf{B}_c, R_2)$
$\mathbf{C}_t \leftarrow \text{tSVD}(\mathbf{C}_c, R_3)$
for each element m in dataset **do**
  $\mathring{\mathcal{G}}_m \leftarrow \mathcal{X}_m \times_1 \mathbf{A}_t^T \times_2 \mathbf{B}_t^T \times_3 \mathbf{C}_t^T$
end for
Train classifier on $\{\text{vec}(\mathring{\mathcal{G}}_m)\}_{m=1}^M$.

The procedure outlined in Algorithm 7 was applied to the ETH-80 dataset by randomly selecting 75% of the available images to serve as training data. The classifiers used were:

1. The Support Vector Machine (SVM), which employed an RBF kernel and a one-vs-one approach for multi-class classification;
2. K-Nearest-Neighbours kNN with the number of neighbours (Euclidean distance) set to three;
3. A feed-forward Multi-Layer Perceptron (MLP) neural network with two hidden layers and a ReLU activation function;
4. A Random Forest (RF) with 10 trees, the nodes of which were expanded until all leaves were pure.

Each classifier was trained based on the data features obtained via the proposed TN summation, as well as on the original data. For a fair comparison, the features extracted via another tensor method were also employed, which involves stacking the images into a higher-order tensor (see [122] for details), referred to as "tensor stacking". The classification rates were computed as an average of 50 realizations and the results are summarized in Table 5.3. Observe the benefits of the proposed method for the SVM, kNN and MLP based classifiers. In particular, with TN summation, the SVM classifier attained an improvement of about 2% over tensor stacking, and almost 5% with respect to a direct usage on the original data.

5.4 Eigenvalue Extraction via TN Summation

This section focuses on how the proposed framework of TN summation can be used for eigenvalue and eigenvector extraction of large-scale matrices. Before showing how this is done,

Table 5.3: Classification rates for the ETH-80 dataset, in %.

	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%

some additional notions are required, namely the TT decomposition in the Matrix Product Operator format (TT-MPO) [3, 123], and how tensor decompositions can be used to extract the largest eigenvalue of large matrices, and associated eigenvector. Specifically, it will be shown how the TN summation framework can be employed to extract the first K largest eigenvalues through integration with Hotelling's deflation [121].

5.4.1 TT-MPO Decomposition

Similarly to the TT-MPS, the TT-MPO can be achieved through an iterative procedure by successively applying the SVD decomposition [3]. It decomposes an N-th order tensor, usually $N \geq 6$, in a set of interconnected 3-rd and 4-th order tensors, as illustrated in Figure 5.7 for an 8-th order tensor. Mathematically, for a $2N$-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{2N}}$, the TT-MPO decomposition is expressed as

$$\begin{aligned} \boldsymbol{\mathcal{X}} &= \boldsymbol{\mathcal{G}}^{(1)} \times_3^1 \boldsymbol{\mathcal{G}}^{(2)} \times_4^1 \cdots \times_4^1 \boldsymbol{\mathcal{G}}^{(N)} \\ &= \langle\!\langle \boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)} \rangle\!\rangle \end{aligned} \tag{5.24}$$

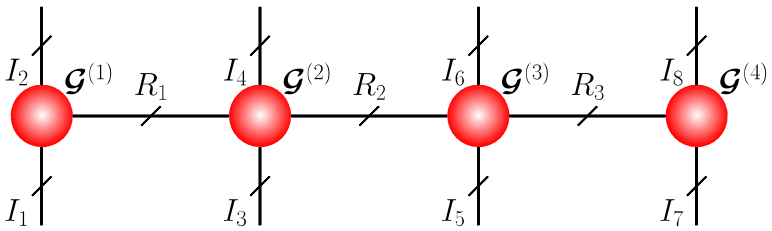


Figure 5.7: TT-MPO representation of an 8-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_8}$. Notice that, differently from the TT-MPS, the first and last cores of the TT-MPO are 3-rd order tensors instead of matrices, while the middle cores are 4-th order tensors.

5.4.2 Largest and Smallest Eigenvalue from Large Scale Matrices

Machine learning algorithms often involve the computation of the largest or smallest eigenvalue of symmetric matrices. In other words, given a matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, the eigenvalue (EVD) problem consists of finding solutions to

$$\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i, \; i = 1, \ldots, I \tag{5.25}$$

where λ_i are the eigenvalues and $\mathbf{x}_i$ the associated eigenvectors. The eigenvector with the largest eigenvalue indicates the direction of maximum variance of the data, and is often of high importance in practical applications. This can be found by maximizing the so called Rayleigh Quotient (RQ), $R(\mathbf{A}, \mathbf{x})$ [60], that is

$$\begin{aligned} \lambda_{\text{max}} &= \max_{\mathbf{x}} R(\mathbf{A}, \mathbf{x}) \\ &= \max_{\mathbf{x}} \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \\ &= \max_{\mathbf{x}} \frac{\langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \end{aligned} \tag{5.26}$$

The critical point of $R(\mathbf{A}, \mathbf{x})$ is the associated eigenvector of λ_{max}. The procedure to solve the above is very similar to inverse iteration, and is summarized in Algorithm 8.

Algorithm 8. Rayleigh Quotient

Input: Matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, initial eigenvalue $\mu \in \mathbb{R}$, initial eigenvector $\mathbf{x} \in \mathbb{R}^I$, threshold ε
$\mathbf{x} \leftarrow \mathbf{x}/||\mathbf{x}||_F$
$\mathbf{y} \leftarrow (\mathbf{A} - \mu \mathbf{I}_I)^{-1}\mathbf{x}$
$\lambda \leftarrow \mathbf{y}^T \mathbf{x}$
$\mu \leftarrow \mu + 1/\lambda$
$err \leftarrow ||\mathbf{y} - \lambda \mathbf{x}||_F / ||y||_F$
while $err > \varepsilon$ **do**
    $\mathbf{x} \leftarrow \mathbf{y}/||\mathbf{y}||_F$
    $\mathbf{y} \leftarrow (\mathbf{A} - \mu \mathbf{I}_I)^{-1}\mathbf{x}$
    $\lambda \leftarrow \mathbf{y}^T \mathbf{x}$
    $\mu \leftarrow \mu + 1/\lambda$
    $err \leftarrow ||\mathbf{y} - \lambda \mathbf{x}||_F / ||y||_F$
end while
Return: λ, $\mathbf{x}$

Although generally effective, the above method becomes prohibitive for very large scale matrices due to computational bottlenecks. This issue can be tackled through low-rank tensor approximations, by representing the cost function in suitable tensor format [60]. Specifically, a large-scale eigenvalue optimization problem can be converted to a set of smaller optimization problems. Assuming that $I = I_1 I_2 \cdots I_N$, this is achieved by directly considering the TT-MPO representation of a structured matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, that is $\boldsymbol{\mathcal{A}} = (\!|\boldsymbol{\mathcal{A}}^{(1)}, \ldots, \boldsymbol{\mathcal{A}}^{(N)}|\!)$, where $\boldsymbol{\mathcal{A}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times I_n \times R_n}$, implying an underlying conceptual $2N$-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_1 \times \cdots \times I_N \times I_N}$. The eigenvector $\mathbf{x} \in \mathbb{R}^I$ is computed by sequential optimization of the cores of the TT-MPS representation of the N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, that is $\boldsymbol{\mathcal{X}} = (\!|\boldsymbol{\mathcal{X}}^{(1)}, \ldots, \boldsymbol{\mathcal{X}}^{(N)}|\!)$, where $\mathbf{x} = \text{vec}(\boldsymbol{\mathcal{X}})$. The expression $\mathbf{x}^T \mathbf{A} \mathbf{x}$ from (5.25) is represented in TN format as shown in the top part of Figure 5.8.

Remark 5.4. The full matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$ and vector $\mathbf{x} \in \mathbb{R}^I$ are never stored in "full" format, but an underlying structure is assumed such that $\mathbf{A}$ can be considered directly in the TT-MPO format, whereas the eigenvector $\mathbf{x}$ is computed by sequential core optimization [60].

Defining,

$$\begin{aligned}\boldsymbol{\mathcal{X}}^{<n} &= (\!|\boldsymbol{\mathcal{X}}^{(1)}, \boldsymbol{\mathcal{X}}^{(2)}, \ldots, \boldsymbol{\mathcal{X}}^{(n-1)}|\!) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times R_{n-1}} \\ \boldsymbol{\mathcal{X}}^{>n} &= (\!|\boldsymbol{\mathcal{X}}^{(n+1)}, \boldsymbol{\mathcal{X}}^{(n+2)}, \ldots, \boldsymbol{\mathcal{X}}^{(N)}|\!) \in \mathbb{R}^{R_n \times I_{n+1} \times \cdots \times I_N}\end{aligned} \tag{5.27}$$

and their corresponding unfoldings

$$\begin{aligned}\mathbf{X}^{<n} &\in \mathbb{R}^{I_1 I_2 \cdots I_{n-1} \times R_{n-1}} \\ \mathbf{X}^{>n} &\in \mathbb{R}^{R_n \times I_{n+1} I_{n+2} \cdots I_N}\end{aligned} \tag{5.28}$$

it results that

$$\mathbf{x} = \mathbf{X}_{\neq n} \mathbf{x}^{(n)} \in \mathbb{R}^{I_1 I_2 \cdots I_N} \tag{5.29}$$

where $\mathbf{x}^{(n)} = \text{vec}(\boldsymbol{\mathcal{X}}^{(n)}) \in \mathbb{R}^{R_{n-1} I_n R_n}$ and

$$\mathbf{X}_{\neq n} = (\mathbf{X}^{>n})^T \otimes \mathbf{I}_{I_n} \otimes \mathbf{X}^{<n} \in \mathbb{R}^{I_1 I_2 \cdots I_N \times R_{n-1} I_n R_n} \tag{5.30}$$

Maximizing the RQ is equivalent to (see [60] for details)

$$\begin{aligned}\max_{\mathbf{x}} R(\mathbf{A}, \mathbf{x}) &= \max_{\mathbf{x}^{(n)}} R(\mathbf{A}, \mathbf{X}_{\neq n} \mathbf{x}^{(n)}) \\ &= \max_{\mathbf{x}^{(n)}} \frac{\mathbf{x}^{(n)} \overline{\mathbf{A}}^{(n)} \mathbf{x}^{(n)}}{\langle \mathbf{x}^{(n)}, \mathbf{x}^{(n)} \rangle} = \lambda_{\text{max}}\end{aligned} \tag{5.31}$$

where

$$\overline{\mathbf{A}}^{(n)} = (\mathbf{X}_{\neq n})^T \mathbf{A} \mathbf{X}_{\neq n} \in \mathbb{R}^{R_{n-1} I_n R_n \times R_{n-1} I_n R_n} \tag{5.32}$$

Each $\mathbf{x}^{(n)}$ is computed separately as the critical point of (5.31) and folded back into $\boldsymbol{\mathcal{X}}^{(n)}$. For relatively small TT-ranks, the matrices $\overline{\mathbf{A}}^{(n)}$ are much smaller than the original matrix $\mathbf{A}$, so that a large-scale eigenvalue problem can be converted into smaller sub-problems, which require solving the set of equations

$$\overline{\mathbf{A}}^{(n)} \mathbf{x}^{(n)} = \lambda \mathbf{x}^{(n)}, \text{ for } n = 1, 2, \ldots, N \tag{5.33}$$

as shown in the bottom part of Figure 5.8.

Remark 5.5. Matrix $\overline{\mathbf{A}}^{(n)}$ is never computed directly via (5.32), but is instead calculated by iteratively computing $\boldsymbol{\mathcal{L}}^{<n}$ and $\boldsymbol{\mathcal{R}}^{>n}$ at each step $n = 1, 2, \ldots, N$ and contracting them with $\boldsymbol{\mathcal{A}}^{(n)}$. This allows for an efficient and practical solution to the problem.

5.4.3 Further Eigenvalues through Deflation via TN Summation

Given a symmetric, large scale matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, it was shown in the previous section how TNs can be employed to compute the RQ and hence extract its largest eigenvalue and the associated eigenvector. Although effective, this method allows only for the computation of one eigenvalue. Given its efficient implementation, it is of interest to explore whether the same approach can be used to compute the first K largest eigenvalues and corresponding eigenvectors of $\mathbf{A}$. To this end, a deflation scheme based on TN summation is proposed, so that the same procedure can be applied in an iterative fashion.

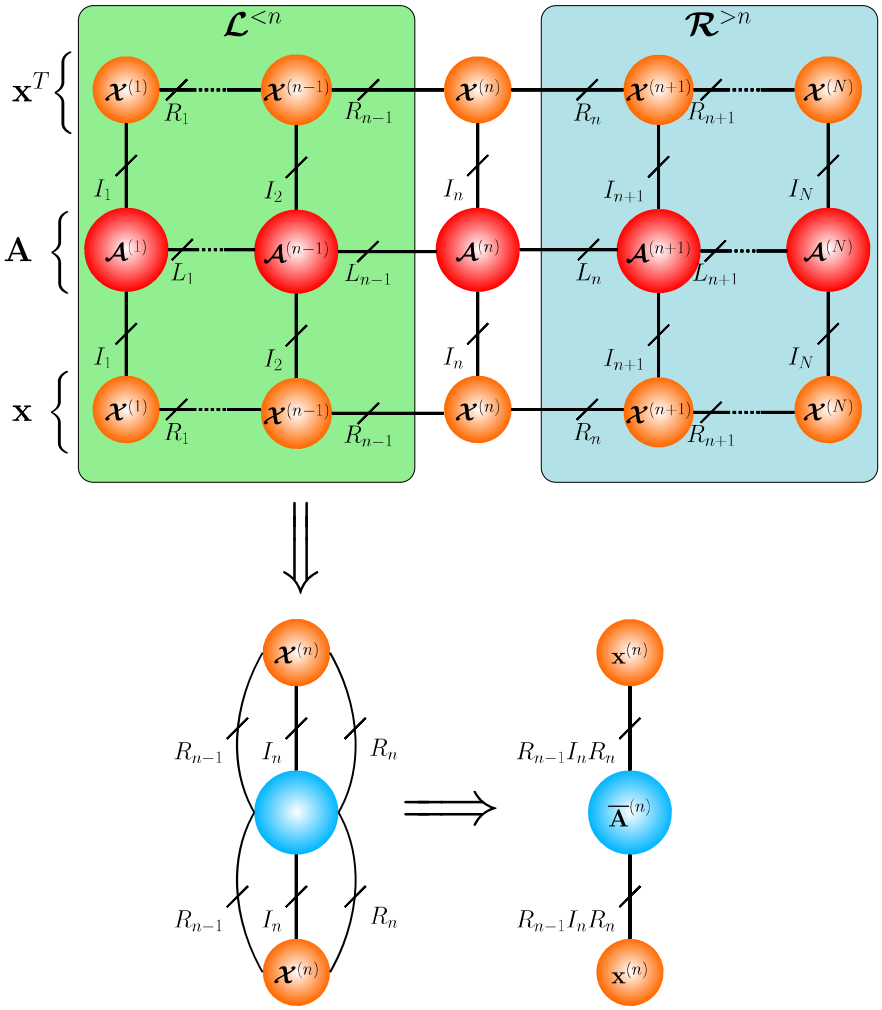


Figure 5.8: A TN composed from a combination of two TT-MPS and one TT-MPO to compute the largest (or smallest) eigenvalue of a large scale matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, and the corresponding eigenvector $\mathbf{x} \in \mathbb{R}^{I}$. Top: Overall representation. Bottom: graphical representations of (5.32) and (5.33) for sequential optimization.

The proposed method stems from the well-known Hotelling's deflation. That is, given a diagonalizable matrix $\mathbf{A}_i \in \mathbb{R}^{I \times I}$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_I$ and associated eigenvectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_I$, a matrix $\mathbf{A}_{i+1} \in \mathbb{R}^{I \times I}$ having the same spectral factorization as $\mathbf{A}_i$ but with $\lambda_i = 0$, can be obtained as

$$\mathbf{A}_{i+1} = \mathbf{A}_i - \lambda_i \mathbf{x}_i \mathbf{x}_i^T \tag{5.34}$$

The combination of the RQ and Hotelling's deflation provides a powerful tool to iteratively compute the eigenvalues and eigenvectors of a diagonalizable matrix, in which the largest eigenvalue is computed via the RQ, then removed via deflation and the process is repeated K times, $K \leq I$.

Therefore, the question is how to incorporate the results from Section 5.4.2 within the TN summation framework. If the matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$ is of huge dimensionality, e.g. $I > 10^{15}$, then a simple operation of deflation as in (5.34) may become problematic due to storage issues.

However, from the properties of TN summation in Table 5.1, this can be achieved in the TN format through a TN subtraction, as explained in the following.

Assuming that the largest i eigenvalues and eigenvectors have been appropriately computed and deflated, and that the $(i+1)$-th eigenvalue λ_{i+1} and associated eigenvector $\mathbf{x}_{i+1} \in \mathbb{R}^I$ have to be found, the first task consists in computing the outer product $\mathbf{x}_i\mathbf{x}_i^T$ within a TN environment. This is done by performing contractions among the individual cores $\boldsymbol{\mathcal{X}}_i^{(n)}$. Specifically, one can consider $\boldsymbol{\mathcal{X}}_i^{(1)} \in \mathbb{R}^{I_1 \times 1 \times R_1}$, $\boldsymbol{\mathcal{X}}_i^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times 1 \times R_n}$ for $n = 2, \dots, N-1$ and $\boldsymbol{\mathcal{X}}_i^{(N)} \in \mathbb{R}^{R_{N-1} \times I_N \times 1}$. Then the following can be defined

$$\begin{aligned}
&\boldsymbol{\mathcal{X}}_i^{(1)} \times_2^2 \boldsymbol{\mathcal{X}}_i^{(1)} = \boldsymbol{\mathcal{Y}}_i^{(1)} \in \mathbb{R}^{I_1 \times R_1 \times I_1 \times R_1}, && \text{for } n = 1 \\
&\boldsymbol{\mathcal{X}}_i^{(n)} \times_3^3 \boldsymbol{\mathcal{X}}_i^{(n)} = \boldsymbol{\mathcal{Y}}_i^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n \times R_{n-1} \times I_n \times R_n}, && \text{for } n = 2, \dots, N-1 \\
&\boldsymbol{\mathcal{X}}_i^{(N)} \times_3^3 \boldsymbol{\mathcal{X}}_i^{(N)} = \boldsymbol{\mathcal{Y}}_i^{(N)} \in \mathbb{R}^{R_{N-1} \times I_N \times R_{N-1} \times I_N}, && \text{for } n = N
\end{aligned} \tag{5.35}$$

With a slight abuse of notation the above can be permuted and reshaped such that

$$\begin{aligned}
&\boldsymbol{\mathcal{Y}}_i^{(1)} \in \mathbb{R}^{I_1 \times I_1 \times R_1^2}, && \text{for } n = 1 \\
&\boldsymbol{\mathcal{Y}}_i^{(n)} \in \mathbb{R}^{R_{n-1}^2 \times I_n \times I_n \times R_n^2}, && \text{for } n = 2, \dots, N-1 \\
&\boldsymbol{\mathcal{Y}}_i^{(N)} \in \mathbb{R}^{R_{N-1}^2 \times I_N \times I_N}, && \text{for } n = N
\end{aligned} \tag{5.36}$$

where $R_1^2 = \overline{R_1 R_1}$, $R_{n-1}^2 = \overline{R_{n-1} R_{n-1}}$, $R_{N-1}^2 = \overline{R_{N-1} R_{N-1}}$. An illustration of this procedure is shown in Figure 5.9.

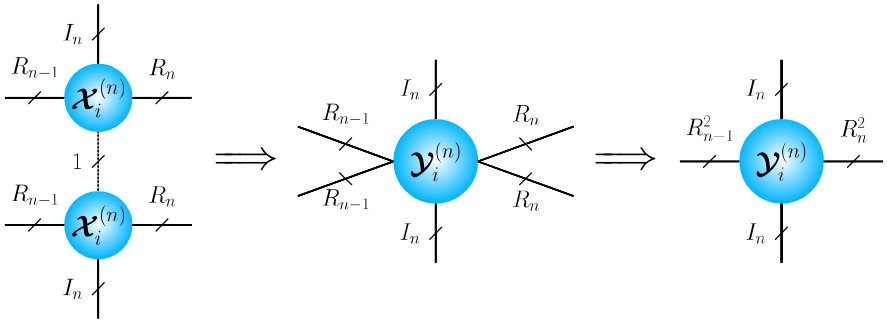


Figure 5.9: Graphical representation of how to perform the outer product $\mathbf{x}_i\mathbf{x}_i^T$, where $\mathbf{x}_i = \text{vec}(\boldsymbol{\mathcal{X}}_i)$, by utilizing only the cores obtained from the TT representation $\boldsymbol{\mathcal{X}}_i = \langle\!\langle \boldsymbol{\mathcal{X}}_i^{(1)}, \dots, \boldsymbol{\mathcal{X}}_i^{(N)} \rangle\!\rangle$.

This defines a new TT-MPO $\boldsymbol{\mathcal{Y}}_i = \langle\!\langle \boldsymbol{\mathcal{Y}}_i^{(1)}, \dots, \boldsymbol{\mathcal{Y}}_i^{(N)} \rangle\!\rangle$, referred to as $\underline{\mathbf{Y}}$, which is physically equivalent to the outer product $\mathbf{x}_i\mathbf{x}_i^T$. Notice that $\underline{\mathbf{Y}}$ is topologically equivalent to $\underline{\mathbf{A}}$, the TT-MPO representation of matrix $\mathbf{A}$. This implies that $\underline{\mathbf{A}}$ and $\underline{\mathbf{Y}}$ can be combined via the proposed framework of TN summation. Specifically, at each iteration $i = 1, 2, \dots, K$, where K is the required number of eigenvalues (and eigenvectors) to be computed, TN $\underline{\mathbf{A}}$ can be updated as $\underline{\mathbf{A}}_{i+1} = \underline{\mathbf{A}}_i - \lambda_i \underline{\mathbf{Y}}_i$, as shown in Figure 5.10.

Remark 5.6. Although at each iteration the dimensions of the cores $\boldsymbol{\mathcal{A}}_i = \langle\!\langle \boldsymbol{\mathcal{A}}_i^{(1)}, \dots, \boldsymbol{\mathcal{A}}_i^{(N)} \rangle\!\rangle$ increase due to concatenation with $\boldsymbol{\mathcal{X}}_i = \langle\!\langle \boldsymbol{\mathcal{X}}_i^{(1)}, \dots, \boldsymbol{\mathcal{X}}_i^{(N)} \rangle\!\rangle$, this has limited impact on computational efficiency for the following reasons.

1. Since TDs attempt to have the fewest possible number of parameters, in general $L_n, R_n \ll$

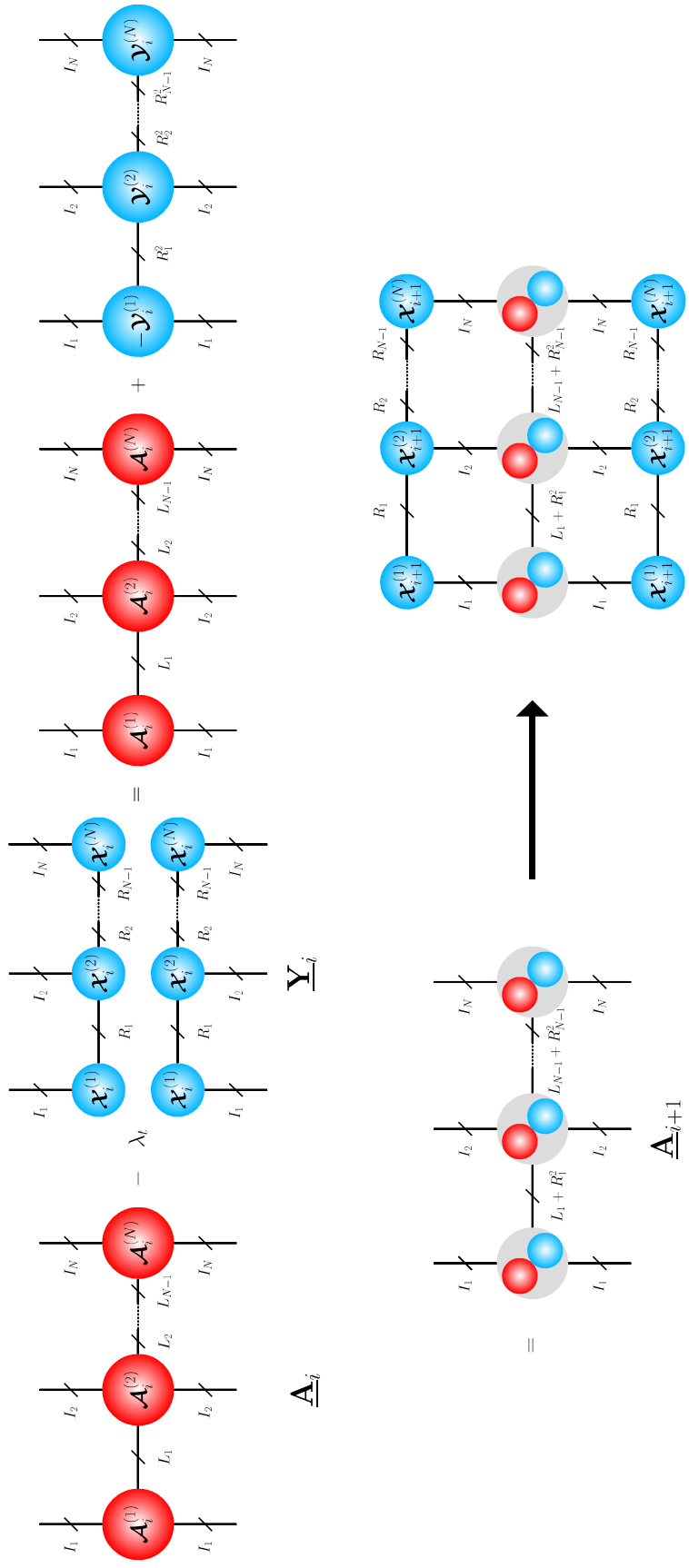


Figure 5.10: Hotelling's deflation via TN summation at iteration i, where $i = 1, \ldots, K$. λ_i is the i-th computed eigenvalue and $\boldsymbol{\mathcal{X}}_i = \langle\!\langle \boldsymbol{\mathcal{X}}_i^{(1)}, \ldots, \boldsymbol{\mathcal{X}}_i^{(N)} \rangle\!\rangle$ are the cores associated to the corresponding eigenvector. The bottom part of the figure shows the starting point for the computation of eigenvector $\mathbf{x}_{i+1}$ represented in TT-MPS format as $\boldsymbol{\mathcal{X}}_{i+1} = \langle\!\langle \boldsymbol{\mathcal{X}}_{i+1}^{(1)}, \ldots, \boldsymbol{\mathcal{X}}_{i+1}^{(N)} \rangle\!\rangle$.

I_n.

2. Optimizations are done sequentially on the individual $\boldsymbol{\mathcal{X}}^{(n)}$'s, and the overall dimensions of matrix $\overline{\mathbf{A}}^{(n)}$ (equation (5.32)) are not affected.

3. The single cores $\boldsymbol{\mathcal{A}}_i = (\!|\boldsymbol{\mathcal{A}}_i^{(1)}, \ldots, \boldsymbol{\mathcal{A}}_i^{(N)}|\!)$ may be stored on individual machines, further ensuring computational and storage memory tractability.

4. Although the dimensions of the cores $\boldsymbol{\mathcal{A}}_i = (\!|\boldsymbol{\mathcal{A}}_i^{(1)}, \ldots, \boldsymbol{\mathcal{A}}_i^{(N)}|\!)$ increase at each iteration, i.e. every time that a new eigenvalue and eigenvector of the underlying matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$ is computed, generally, the requested number of eigenvalues K is s.t. $K \ll I$, as it is well-known that only the few largest eigenvalues and eigenvectors carry most of the physically relevant information.

5.5 Chapter Summary

A framework for Tensor Network (TN) summation has been proposed. This has been achieved via an intelligent block arrangement of the cores of two or more TNs, on the condition that they are isomorphic and have physical modes of equal dimensionality. This yields a new, isomorphic, TN mathematically equivalent to the sum of the underlying tensors. The approach has been formally validated for chains of matrices, for TNs in the Tucker format, and for Tensor Trains (TTs). The proposed framework has been further formalized by introducing properties of addition for TNs in Section 5.2.2, which are found to be analogous to their counterparts in standard scalar and matrix addition. These are the associativity and commutativity properties, and the existence of the identity element and the inverse operator.

Benefits of the TN summation framework have first been demonstrated through an analogy with feature fusion in Section 5.3, whereby a new algorithm for image classification was proposed, which rests solely upon the proposed sum of TNs. Tests on the ETH-80 dataset have demonstrated that the mixture of features obtained via TN summation can significantly enhance classification rates of standard machine learning classifiers.

Further advantages of the proposed framework have been explored in light of eigenvalue and eigenvector extraction for huge matrices. This was achieved through employment of the Rayleigh Quotient (RQ) expressed in tensor format, followed by an implementation of Hotelling's deflation executed directly in TN format in Section 5.4.3. Expanding on the method presented in Section 5.4.2, which allows to extract only the single largest eigenvalue and associated eigenvector, the proposed approach, which lies on the TN summation framework, allows to iteratively extract several largest eigenvalues and associated eigenvectors of matrices of high dimensionality, carrying a high practical importance in large scale problems.

Chapter 6

The Support Tensor Machine and its Applications to Financial Forecasting

> Every success story is a tale of constant adaption, revision and change.
>
> *Richard Branson*

Chapter Overview

This chapter introduces the Support Tensor Machine (STM) as an extension of the well-known Support Vector Machine (SVM) to the tensor realm, with a focus on time-series analysis. The fundamental difference between the two is that while the SVM deals with vector inputs, the STM operates directly on tensors. Vectorization of the original data destroys its underlying structure, implying that it is expected for the STM to perform better than the SVM if the input data has underlying patterns from which useful information can be retrieved.

The chapter first opens with an overview of the SVM. The mathematical theory of the SVM is then smoothly extended to accommodate tensors, presenting the STM. A computationally efficient least-squares formulation of the STM (LS-STM) is considered, while a novel method to allow kernel usage in the LS-STM is introduced. Equipped with this new functionality, the usefulness of the LS-STM is investigated within the financial application of forecasting the daily direction of movement of price of the S&P 500 financial index. The LS-STM based results are interpreted probabilistically via Platt scaling, while performance is evaluated comprehensively in terms of accuracy rate, annualized Sharpe ratio, and annualized volatility. All performance metrics conclusively demonstrate the superiority of LS-STM over standard SVM.

6.1 Introduction

Analysis of time-series underpins numerous practical applications, and is a major paradigm for the Statistical Signal Processing and Machine Learning (ML) communities. Among the existing methods, the Support Vector Machine (SVM), introduced by Cortes and Vapnik in 1995 [124], is particularly interesting as it employs the Structural Risk Minimization (SRM) principle [125]. The aim is thereby to minimize the upper bound of the generalization error, instead of minimizing the training error [126], which significantly reduces the risk of overfitting. In addition, since the training of the SVM is equivalent to solving linearly constrained quadratic programming problems, the SVM solution is both unique and always attains the global minimum. The underpinning flexibility and physical intuition behind SVMs made them an algorithmic workhorse in a wide variety of applications, from biological sciences, through to text

categorization, to image classification, and time-series prediction [127, 128, 129, 130].

Motivated by recent results across different research areas [18, 24, 63, 81] which show that the exploration of structural information can lead to better expressive power of the employed algorithms, the SVM framework has been extended to its tensor version, referred to as the Support Tensor Machine (STM) [85]. The STM operates directly on tensor-valued inputs, which has shown to bring noticeable improvements over vector-valued SVMs in applications such as real world image and video processing [131]. However, the full potential of the STM is yet to be explored, especially within the time-series forecasting paradigm. To this end, in this chapter the problem of financial time series forecasting is considered [86], as it is one of the most challenging and practically-relevant problems in time-series in general. Also, given that the SVM has been extensively used within a financial context, it is natural to ask whether a generalization to the STM would inherit the advantages of tensor-variate vs matrix- and vector-variate modelling of multichannel data. Examples of the SVM in finance include the works in [130, 132, 133, 134].

The focus in this chapter will be devoted to the least-squares (LS) formulation of STM [60], a more computationally efficient variant over standard STM, similarly to LS-SVM over standard SVM [135]. A rigorous method to allow for kernel usage in LS-STM is proposed, based on a mode-wise least-squares approximation of the related weighting parameters. The importance of tensorization in exploiting structural information is highlighted, then training is performed using a Radial Basis Function (RBF) kernel, and the outputs of LS-STM are interpreted probabilistically via Platt scaling [136]. The so established LS-STM framework is employed to forecast the daily direction of movement of the S&P 500, over the period from January 2006 to January 2017, using as predictors past samples from the VIX (the implied index of volatility for the S&P 500), the GC1 gold commodity, and the S&P 500 itself. The S&P 500 has been chosen for its *efficiency* [137, 138], as, according to the Efficient Market Hypothesis (EMH) [139], it reflects well the available information in markets, thus making forecasting rather difficult. A comprehensive evaluation is performed by comparing the results obtained with LS-STM against those obtained with SVM on the same dataset. The superiority of the LS-STM based proposed framework is demonstrated, in terms of both accuracy rates and other common financial metrics.

6.2 From the Support Vector Machine to the Support Tensor Machine

In the following, the theories behind the SVM and STM are introduced, and it is shown how the STM can be thought of an ensemble of SVMs, one per mode, acting on tensor inputs.

6.2.1 Support Vector Machine

The classical SVM is a binary classifier which aims to determine a hyperplane able to discriminate elements within a dataset of size M into a "positive" or "negative" class. That is, given a dataset $\{\mathbf{x}_m\}_{m=1}^{M}$, $\mathbf{x}_m \in \mathbb{R}^N$, the SVM operates on the pairs $\{\mathbf{x}_m, t_m\}$, where the binary label $t_m \in \{+1, -1\}$ indicates the membership of $\mathbf{x}_m$ to the positive or negative class, respectively. The separating hyperplane is entirely defined by a projection vector $\mathbf{w} \in \mathbb{R}^N$ and a bias term

$b \in \mathbb{R}$, and an unlabelled, or test, datapoint $\mathbf{x}_* \in \mathbb{R}^N$ is assigned a label, t_*, by

$$\begin{aligned} y(\mathbf{x}_*) &= \mathbf{w}^T \mathbf{x}_* + b \\ t_* &= \text{sign}(y(\mathbf{x}_*)) \end{aligned} \tag{6.1}$$

The optimal $\mathbf{w}$ and b are determined from the training data, by solving the following quadratic programming problem,

$$\begin{aligned} &\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_F^2 + C\sum_{m=1}^{M} \xi_m \\ &\text{s.t. } t_m(\mathbf{w}^T \mathbf{x}_m + b) \geq 1 - \xi_m \\ &\xi_m > 0,\ m = 1, \ldots, M \end{aligned} \tag{6.2}$$

where the slack variables $\xi_m, m = 1, \ldots, M$ are introduced to deal with non-linearly separable data, and $C > 0$ is a regularization parameter. Due to the presence of slack variables, (6.2) is also referred to as "soft-margin" SVM.

The classical SVM can be simplified into its least-squares formulation, the LS-SVM, proposed in [135], where the optimization problem is modified to

$$\begin{aligned} &\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_F^2 + \frac{\gamma}{2}\sum_{m=1}^{M} \epsilon_m^2 \\ &\text{s.t. } t_m(\mathbf{w}^T \mathbf{x}_m + b) = 1 - \epsilon_m,\ m = 1, \ldots, M \end{aligned} \tag{6.3}$$

where ϵ_m, $m = 1, \ldots, M$ are the losses with respect to the targets $t_m \in \{+1, -1\}$. The replacement of the slack variables in (6.2) with the square losses in (6.3), and the inequalities with equalities, makes it such that the problem can be expressed as a series of linear equations, allowing the LS-SVM to be solved with simple matrix multiplications, making it computationally cheaper than the soft-margin SVM, while still being able to maintain comparable performance [60, 135].

6.2.2 Support Tensor Machine

Consider now a set of M measurements $\{\boldsymbol{\mathcal{X}}_m, t_m\}_{m=1}^M$, where each sample is an N-th order tensor $\boldsymbol{\mathcal{X}}_m \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The STM trains an N-th order weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in order to assign a label to $\boldsymbol{\mathcal{X}}_*$ in a way analogous to (6.1), by

$$\begin{aligned} y(\boldsymbol{\mathcal{X}}_*) &= \langle \boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{X}}_* \rangle + b \\ t_* &= \text{sign}(y(\boldsymbol{\mathcal{X}}_*)) \end{aligned} \tag{6.4}$$

Representing the weight tensor $\boldsymbol{\mathcal{W}}$ as a rank-1 tensor, that is,

$$\boldsymbol{\mathcal{W}} = \mathbf{w}_1 \circ \mathbf{w}_2 \circ \cdots \circ \mathbf{w}_N \tag{6.5}$$

where, $\mathbf{w}_n \in \mathbb{R}^{I_n}$, $n = 1, \ldots, N$, then the classification problem of STM can be formulated as

$$\begin{aligned} &\arg\min_{\mathbf{w}_n, b} \frac{1}{2}\left|\left| \bigotimes_{n=N}^{1} \mathbf{w}_n \right|\right|_F^2 + C \sum_{m=1}^{M} \xi_m, \ n = 1, \ldots, N \\ &\text{s.t. } t_m \left(\boldsymbol{\mathcal{X}}_m \overline{\times}_{n=1}^{N} \mathbf{w}_n + b \right) \geq 1 - \xi_m \\ &\xi_m > 0, \ m = 1, \ldots, M \end{aligned} \tag{6.6}$$

where $\boldsymbol{\mathcal{X}}_m \bar{\times}_{n=1}^{N} \mathbf{w}_n = \boldsymbol{\mathcal{X}}_m \bar{\times}_1 \mathbf{w}_1 \bar{\times}_2 \cdots \bar{\times}_N \mathbf{w}_N$.

Remark 6.1. From the formulation in (6.6), the STM can be thought as an ensemble of SVMs, each acting on one mode of the tensor.

The above comprises N quadratic programming sub-problems, where the n-th sub-problem can be reformulated via the following optimization,

$$\begin{aligned} &\arg\min_{\mathbf{w}_n, b} \frac{1}{2}||\mathbf{w}_n||_F^2 \prod_{\substack{i=1 \\ i \neq n}}^{N} \left(||\mathbf{w}_i||_F^2 \right) + C \sum_{m=1}^{M} \xi_m \\ &\text{s.t. } t_m \left(\mathbf{w}_n^T \left(\boldsymbol{\mathcal{X}}_m \overline{\times}_{\substack{i=1 \\ i \neq n}}^{N} \mathbf{w}_i \right) + b \right) \geq 1 - \xi_m \\ &\xi_m > 0, \ m = 1, \ldots, M \end{aligned} \tag{6.7}$$

Analogously to the SVM, also the STM admits a least-squares version, namely the LS-STM. Its solution is obtained by the following ALS iteration, through the procedure summarized in Algorithm 9.

$$\begin{aligned} &\arg\min_{\mathbf{w}_n, b} \frac{1}{2}||\mathbf{w}_n||^2 \prod_{1 \leq i \leq N}^{i \neq n} \left(||\mathbf{w}_i||^2 \right) + \frac{\gamma}{2} \epsilon^T \epsilon \\ &\text{s.t. } t_m \left(\mathbf{w}_n^T \left(\boldsymbol{\mathcal{X}}_m \overline{\times}_{\substack{i=1 \\ i \neq n}}^{N} \mathbf{w}_i \right) + b \right) = 1 - \epsilon_m \\ &m = 1, \ldots, M \end{aligned} \tag{6.8}$$

Finally, a label is assigned to a new datapoint $\boldsymbol{\mathcal{X}}_*$, by

$$t_* = \text{sign}\big(y(\boldsymbol{\mathcal{X}}_*)\big) = \text{sign}\big(\boldsymbol{\mathcal{X}}_* \times_1 \mathbf{w}_1 \times_2 \cdots \times_N \mathbf{w}_N + b\big) \tag{6.9}$$

6.2.3 Visualizing the STM

Due to its nature, it is difficult to visualize the STM in a similar fashion to the SVM, which can be easily represented on a 2-dimensional plane with vectors shown as points. However, one can visualize the decision operation in (6.9) by noting that an inner product $\langle \boldsymbol{\mathcal{X}}_*, \boldsymbol{\mathcal{W}} \rangle$ is taking place, which can be represented diagrammatically. An example 3-rd order tensors is shown in Figure 6.1. The inner product of a 3-rd order tensor with the outer product of three vectors yields a scalar: notice no free edges on the right-hand side of the figure.

Algorithm 9. Least-Squares Support Tensor Machine (LS-STM)

Input: Dataset $\{\boldsymbol{\mathcal{X}}_m, t_m\}, m = 1, \ldots, M$, with $\boldsymbol{\mathcal{X}}_m \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
Output: Set of weights $\{\mathbf{w}_n\}, n = 1, \ldots, N$ and bias b.

1: Initialize randomly $\{\mathbf{w}_n\}, n = 1, \ldots, N$
2: **while** not converged or *max iterations* reached **do**
3: **for** $n = 1$ to N **do**
4: $\eta \leftarrow \prod_{i \neq n} ||\mathbf{w}_i||^2$
5: $\mathbf{x}_m = \boldsymbol{\mathcal{X}}_m \bar{\times}_{i \neq n} \mathbf{w}_i$
6: Find $\mathbf{w}_n$ by optimizing:

$$\min_{\mathbf{w}_n, b, \epsilon} \frac{\eta}{2} ||\mathbf{w}_n||^2 + \frac{C}{2} \epsilon^T \epsilon$$
$$\text{s.t.} \quad t_m(\mathbf{w}_n^T \mathbf{x}_m + b) = 1 - \epsilon_m, \quad m = 1, \ldots, M$$

7: **end for**
8: **end while**
9: **Return** $\{\mathbf{w}_n\}, n = 1, \ldots, N$ and b.

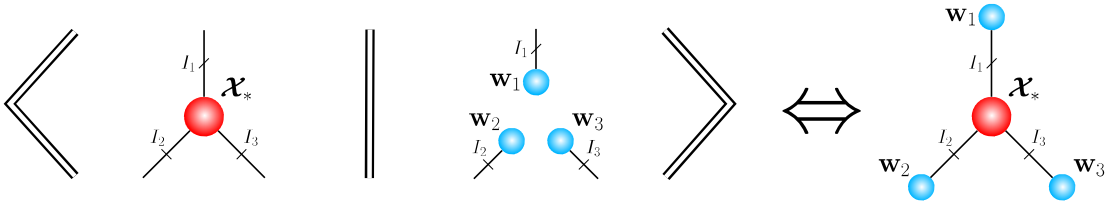


Figure 6.1: Graphical representation of the inner product $\langle \boldsymbol{\mathcal{X}}_*, \boldsymbol{\mathcal{W}} \rangle$, where $\boldsymbol{\mathcal{X}}_* \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\boldsymbol{\mathcal{W}} = \mathbf{w}_1 \circ \mathbf{w}_2 \circ \mathbf{w}_3$, and $\mathbf{w}_1 \in \mathbb{R}^{I_1}$, $\mathbf{w}_2 \in \mathbb{R}^{I_2}$, $\mathbf{w}_3 \in \mathbb{R}^{I_3}$.

6.3 Kernel Usage in STM

To rectify issues related to linear non-separability of a dataset, in classical SVM and LS-SVM, $\{\mathbf{x}_m\}_{m=1}^M$, a mapping

$$\phi : \mathbf{x} \in \mathbb{R}^N \mapsto \phi(\mathbf{x}) \in \mathbb{R}^L, \quad L > N \tag{6.10}$$

can be defined to make the data separable in a higher L-dimensional space. The class label, t_*, of a new datapoint, $\mathbf{x}_*$, is now obtained by taking the sign of $y(\mathbf{x}_*) = \mathbf{w}^T \phi(\mathbf{x}_*) + b$. The weights $\mathbf{w}$ are then found as

$$\mathbf{w} = \sum_{m=1}^{M} \alpha_m t_m \phi(\mathbf{x}_m) \tag{6.11}$$

where α_m, $m = 1, \ldots, M$ are the Lagrange multipliers associated to the dual of the original problem [124]. Hence, a test point $\mathbf{x}_*$ is assigned a label by taking the sign of

$$y(\mathbf{x}_*) = \sum_{m=1}^{M} \alpha_m t_m \big(\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_*)\big) + b \tag{6.12}$$

Notice that it is often intractable to compute ϕ explicitly, as the dimension of a space resulting from the mapping in (6.10) may be significantly higher than the original one, or even infinite. To this end, the kernel trick is used, where the inner product $\phi(\mathbf{x})^T \phi(\mathbf{x}_*)$ is implicitly

performed in the original space by the kernel function

$$k(\mathbf{x}, \mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}) = \phi(\mathbf{x})^T \phi(\mathbf{x}_*) \tag{6.13}$$

Therefore, the label in (6.12) becomes

$$y(\mathbf{x}_*) = \sum_{m=1}^{M} \alpha_m t_m k(\mathbf{x}_m, \mathbf{x}_*) + b \tag{6.14}$$

which implies that $\mathbf{w}$ is not calculated explicitly; the same is valid for LS-SVM [135]. However, for each mode $n = 1, \ldots, N$, the LS-STM (Algorithm 9) computes $\mathbf{w}_n$, which is strictly required since, at each iteration and for each mode, a set of $\{\mathbf{x}_m = \boldsymbol{\mathcal{X}}_m \bar{\times}_{i \neq n} \mathbf{w}_i\}_{m=1}^{M}$ needs to be computed. The use of kernels is known to generally increase the performance of SVM and LS-SVM significantly, and it is thus natural to explore the same concept in LS-STM.

To circumvent the problem of requiring a $\mathbf{w}_n$ for each mode, an approach is proposed [86] in which the associated α_m's are found using the kernel and then (6.14) is employed to compute $\mathbf{w}_n$ in a least-squares fashion. Specifically, consider a dataset $\{\boldsymbol{\mathcal{X}}_m\}_{m=1}^{M}$, $\boldsymbol{\mathcal{X}}_m \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, and define

$$\boldsymbol{\Omega}_n = \begin{bmatrix} \mathbf{x}_1, & \mathbf{x}_2, & \ldots, & \mathbf{x}_M \end{bmatrix}^T \in \mathbb{R}^{M \times I_n} \text{ for } n = 1, \ldots, N \tag{6.15}$$

Now, assume that the $\mathbf{x}_m$'s are descriptive of the features in the dataset and also assume a large enough M, to yield

$$y(\mathbf{x}_*) \simeq y(\mathbf{x}_*)_{m \neq i} = \sum_{\substack{m=1 \\ m \neq i}}^{M} \alpha_m t_m k(\mathbf{x}_m, \mathbf{x}_*) + b = \boldsymbol{\psi}_i^* + b \tag{6.16}$$

where

$$\boldsymbol{\psi}_i^* = \sum_{\substack{m=1 \\ m \neq i}}^{M} \alpha_m t_m k(\mathbf{x}_m, \mathbf{x}_*) \tag{6.17}$$

By splitting the dataset $\boldsymbol{\Omega}_n$ into M smaller data subsets, each consisting of $(M-1)$ training elements and one validation element, the optimal $\mathbf{w}_n$ can be found using the values of $y(\mathbf{x}_m)$, i.e. using the elements in the training set. Upon defining $y(\mathbf{x}_m)_{m \neq i} := y_m$, each y_m is computed by

$$\begin{bmatrix} y_1, & y_2, & \ldots, & y_M \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{\psi}_1^1, & \boldsymbol{\psi}_2^2, & \ldots, & \boldsymbol{\psi}_M^M \end{bmatrix}^T + b \tag{6.18}$$

where b is found by solving the LS-SVM problem in Algorithm 9. Finally, the $\mathbf{w}_n$'s are found through an equality constraint

$$\begin{aligned} \begin{bmatrix} y_1, & y_2, & \ldots, & y_M \end{bmatrix}^T &= \begin{bmatrix} \mathbf{x}_1^T \mathbf{w}_n, & \mathbf{x}_2^T \mathbf{w}_n, & \ldots, \mathbf{x}_M^T \mathbf{w}_n \end{bmatrix}^T + b \\ &= \boldsymbol{\Omega}_n \mathbf{w}_n + b \end{aligned} \tag{6.19}$$

For $\mathbf{y} = [y_1, y_2, \ldots, y_M]^T$, a $\mathbf{w}_n$ is required such that

$$\mathbf{y} = \mathbf{\Omega}_n \mathbf{w}_n + b \tag{6.20}$$

It is generally true in practice that if $M > I_n$, then $\mathbf{\Omega}_n^+ \mathbf{\Omega}_n = \mathbf{I}_{I_n}$ where

$$\mathbf{\Omega}_n^+ = \left(\mathbf{\Omega}_n^T \mathbf{\Omega}_n\right)^{-1} \mathbf{\Omega}_n^T \tag{6.21}$$

Therefore, $\mathbf{w}_n$ can be estimated in a least-squares fashion as

$$\mathbf{w}_n = \mathbf{\Omega}_n^+ (\mathbf{y} - b) \tag{6.22}$$

When executing the LS-STM algorithm, the above is achieved for every mode $n = 1, \ldots, N$, until convergence. This allows for the usage of a kernel, while at the same time retrieving the required $\mathbf{w}_n$. For the application of financial forecasting, an RBF kernel is employed due to its suitability to work with non-linearly separable data, that is

$$k(\mathbf{x}, \mathbf{x}_*) = \exp\left(- \frac{||\mathbf{x} - \mathbf{x}_*||_F^2}{2\sigma^2} \right) \tag{6.23}$$

The parameter σ is selected using a data driven approach, as described in [140, 141].

6.4 Implementation

This section is devoted to explaining how the data was prepared for the LS-STM to be employed on the problem of forecasting the S&P 500.

6.4.1 Financial Terms

First, some financial definitions are in order. Daily closing prices, S_t and traded volumes, V_t are considered. Daily percentage returns, r_t, are defined as

$$r_t = \frac{S_t - S_{t-1}}{S_{t-1}} \tag{6.24}$$

while percentage change in traded volume, v_t, takes the form of

$$v_t = \frac{V_t - V_{t-1}}{V_{t-1}} \tag{6.25}$$

The annualized return, R_A, is defined as

$$R_A = 250\bar{r} \tag{6.26}$$

where $\bar{r} = \frac{1}{L}\sum_{t=1}^{L} r_t$ and L is the total number of samples. Annualized volatility, σ_A, is given by

$$\sigma_A = \sqrt{250}\sqrt{\frac{1}{L-1}\sum_{t=1}^{L}(r_t - \bar{r})^2} \tag{6.27}$$

Finally, the Sharpe ratio, SR_A, used to quantify the "goodness" of a financial strategy, is given as

$$\text{SR}_A = \frac{R_A}{\sigma_A} \tag{6.28}$$

The following terms are also defined

1. Long position: Buying an asset unit at market open and selling it at market close, for a given trading day;
2. Short position: Selling a borrowed asset unit at market open, purchasing it back at market close and returning it to the lender.

Remark 6.2. Long positions are profitable if the asset increases in value, short positions if the asset decreases in value.

6.4.2 Data Tensorization

In this application, the goal is to forecast the daily direction of movement of the S&P 500, using historical data from the VIX, the implied index of volatility of the S&P 500, and the GC1 gold commodity (data freely available online). The VIX and gold are chosen for the investigation since they are generally considered to be good predictors of the S&P 500 [142, 143]. Moreover, the S&P 500 itself is added as a predictor, in an attempt to infer additional information about its future movements from its past. Daily closing prices and volumes for these three time-series were collected from January 2006 to January 2017.

Owing to the flexibility of the tensor framework, the data were arranged into a third order tensor and the analysis performed with sliding windows of 250 days (one financial year), as shown in Figure 6.2. The second frontal slice is the replicated matrix from the first slice, but lagged by one sample, so as to introduce additional autocorrelation structure among the returns of the three time series. For each slice, columns were standardized using a hyperbolic tangent, to reduce the possibility of introducing biases in LS-STM.

Given a "tensor batch" of 250 time samples, the training dataset, $\{\boldsymbol{\mathcal{X}}_m, t_m\}_{m=1}^{248}$, is formed by selecting each $\boldsymbol{\mathcal{X}}_m \in \mathbb{R}^{2\times 3\times 3}$ as shown in Figure 6.2, while the label t_m is assigned as $t_m \leftarrow \text{sign}(r_{m+1})$, where r_{m+1} is obtained from the S&P 500 column of the first tensor slice. For each sliding window, the test data consists of a single pair $\{\boldsymbol{\mathcal{X}}_t, t_t\}$, at the time instant $t = 249$. In other words, the test label is the sign of the bottom-right entry of the first slice of the current batch. Essentially, at time t, the model aims to predict the direction of movement of the S&P 500 for the time instant $(t+1)$, using all information available up to time t.

6.4.3 Platt Scaling

For better performance, the outputs of the LS-STM are interepreted probabilistically via Platt scaling. Standard binary classifiers do not take into account posterior probabilities. In order to

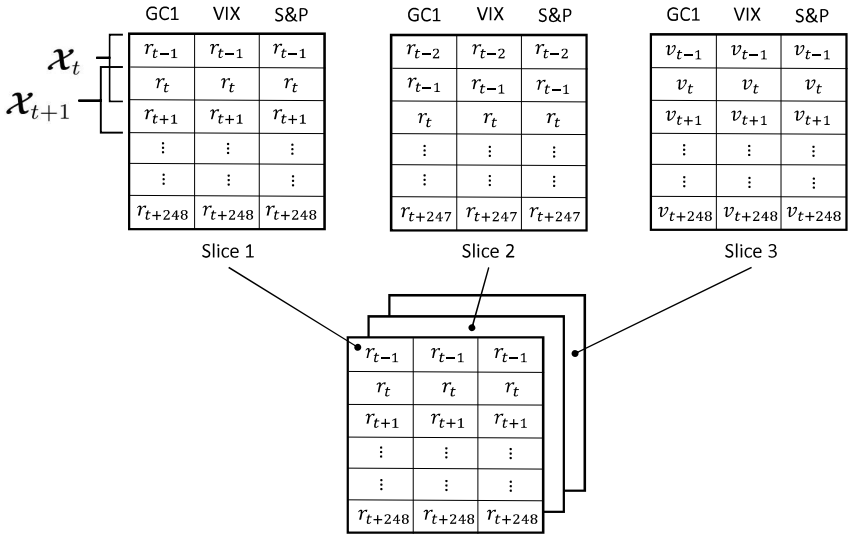


Figure 6.2: One batch of tensor data obtained through a sliding window of 250 time samples, i.e. one financial year. The first (and second) slice include (lagged) percentage change in prices, r_t, while the third slice comprises daily percentage change in trading volume, v_t.

assess the level of confidence in the classification model, its outputs (or scores) may be transformed into a distribution over classes. This can be achieved via Platt scaling, a method devised by John Platt in 1999 [136].

Consider a new test sample, $\mathbf{x}_* \in \mathbb{R}^N$, and the value of the discriminant function, $y_* := y(\mathbf{x}_*)$, then Platt scaling produces probability estimates in the form

$$P(t_* = 1|y_*) = \frac{1}{1 + \exp(Ay_* + B)} \tag{6.29}$$

which is a logistic transformation of the classifier score, y_*, where $A, B \in \mathbb{R}$ are parameters learnt by fitting a sigmoid to the model outputs. Platt-scaling has shown to be effective in improving and understanding classifications models in general [144].

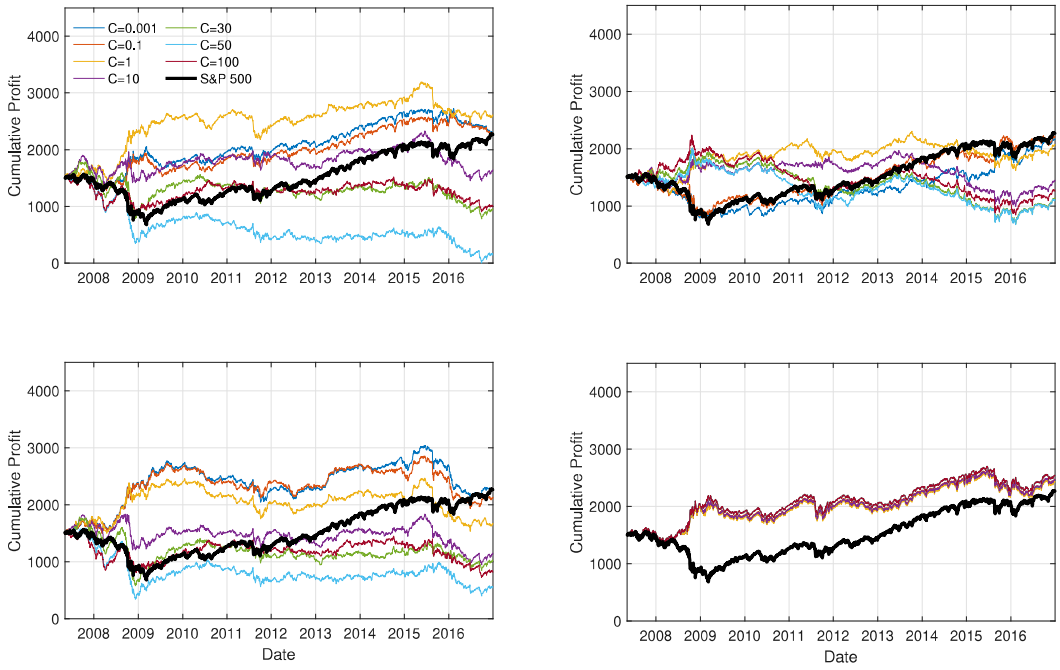
6.5 Simulations and Discussions

An RBF kernel was employed within the LS-STM according to the method presented in Section 6.3, and tuned according to the data-driven method outlined in [140]. For each time sample, the assigned label of $+1$ indicates a *long position* (forecasting an increase in price), and a label of -1 a *short position* (forecasting a decrease in price). The accuracy was calculated as the percentage of LS-STM correctly forecasting the S&P 500 price movement, with true movements from historical data serving as ground truth. The performance of LS-STM was evaluated against that of SVM, for a varying $C = \{0.001, 0.1, 1, 10, 30, 50, 100\}$. A summary of results is shown in Table 6.1, while the corresponding generated profits are presented in Figure 6.3.

For almost all the considered values of C, simulation results generally show the superiority of LS-STM over SVM. The SVM only outperformed LS-STM in certain metrics, and for values

Table 6.1: Results for pre- and post-Platt scaling for SVM and LS-STM, for varying values of C. The best results are highlighted in bold. The subscript $(\cdot)_P$ indicates post-Platt scaling.

	$C = 0.001$			$C = 0.1$			$C = 1$			$C = 10$			$C = 30$			$C = 50$			$C = 100$		
	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A
SVM	52.27	0.39	0.15	52.32	0.38	0.15	**52.55**	**0.58**	**0.13**	51.81	0.14	0.16	51.07	-0.12	0.22	50.14	0.30	5.24	50.56	-0.10	0.23
SVM_P	51.90	0.44	**0.13**	51.44	0.37	**0.13**	51.25	0.16	0.14	51.11	-0.06	0.19	50.88	-0.05	0.25	50.65	-0.14	0.37	50.42	-0.17	0.25
LS-STM	**52.46**	0.29	0.23	52.04	0.31	0.21	51.02	0.31	0.16	49.16	0.06	0.18	48.93	-0.03	0.22	49.07	-0.05	0.23	49.30	-0.01	0.20
LS-STM_P	52.32	**0.46**	0.14	**52.37**	**0.46**	0.14	52.32	0.45	0.15	**52.41**	**0.46**	**0.14**	**52.41**	**0.49**	**0.14**	**52.41**	**0.49**	**0.14**	**52.41**	**0.49**	**0.14**



(a) SVM profits. Top: pre-Platt scaling. Bottom: post-Platt scaling.

(b) LS-STM profits. Top: pre-Platt scaling. Bottom: post-Platt scaling.

Figure 6.3: Cumulative profits generated using SVM and LS-STM based strategies, for varying values of C.

of $C = \{0.001, 0.1, 1\}$. For $C = \{0.001, 0.1\}$, the Platt-scaled SVM, SVM$_P$, attained better values in terms of σ_A, while the accuracy rates and SR$_A$ were still superior for LS-STM and LS-STM$_P$. Although for $C = 1$ SVM did perform better than LS-STM in all three metrics, a closer inspection on the top graph in Figure 6.3a reveals that such promising results were mostly a consequence of the high market predictability of 2008, due to the sub-prime mortgage crisis. In fact, the profits associated to $C = 1$ tend to decrease in more recent times, and become comparable to those generated by other values of C. This suggests that the overall SVM performance for $C = 1$ can mostly be attributed to a lucky choice of parameters, which, moreover, was particularly beneficial over a relatively short time-frame as in later years it clearly lost its initial momentum. To further emphasize this point, notice how the use of Platt scaling did not bring any major advantage to the SVM output, although the method is generally known to improve performance.

On the other hand, results for LS-STM and, in particular, LS-STM$_P$ exhibit far better stability than their SVM counterparts. Of particular interest is the effect that Platt-scaling has on LS-STM, as shown in Figure 6.3b. For the vast majority of testing points, adding a probabilistic interpretation on LS-STM causes its outputs to converge to equal values independently from the choice of parameters used. Differently from SVM in which it had no positive effect, the advantages that Platt scaling brings to LS-STM are mostly two-fold:

1. It introduces enhanced "stability" to LS-STM, in the sense that results can be attributed mostly to data-driven information and not to convenient parameter tuning. For the considered problem of S&P 500 forecasting, this is significantly useful in practice as informed decisions can be made based on data structure and thus with a reduced risk of overfitting.

2. The LS-STM$_P$ generates better values than LS-STM in terms of accuracy rate, SR$_A$, and σ_A, for all considered values of C, with the sole exception of the accuracy rate for $C = 0.001$, which is higher for pre-Platt scaled LS-STM. Moreover, for $C = \{10, 30, 50, 100\}$, the LS-STM$_P$ always attains the best overall performance for all three metrics.

6.5.1 Effects of Platt-Scaling

The effects of Platt-scaling can be explained through inspection of its parameters, as shown in Figure 6.4, where the A parameter (the slope of the sigmoid) was the same for every choice of C, whereas the parameter B (the bias of the sigmoid) followed similar patterns for all considered values of C. This outcome may be attributed to the superior encoding of structural information that tensors inherently have over vectors. Tensorizing the data as described in Figure 6.2 introduced additional information along each mode which was captured by the optimized LS-STM weights $\{\mathbf{w}_n\}_{n=1}^3$. This further highlights the general benefits of tensors over vectors, and indicates that the chosen method for tensorization was indeed suitable for the considered application.

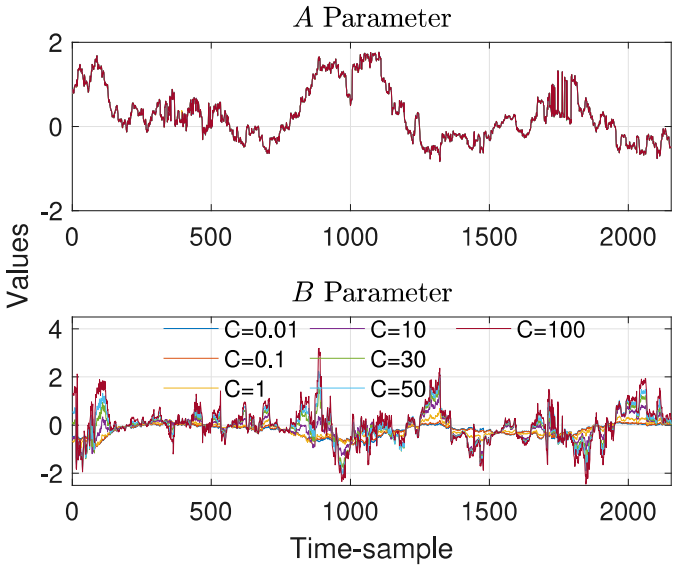


Figure 6.4: The time evolution of the A and B parameters of the sigmoid in (6.29), obtained by applying Platt-scaling on LS-STM.

6.5.2 Introducing Confidence Measures

Given the nature of the application, it is of practical interest to devise a methodology that would aid making decisions with an added degree of confidence. One way to achieve this is to

add an additional possible label, 0, to the predicted data, which indicates neither a *long* nor a *short* strategy. In this way, when the algorithm is not confident enough in prediction, it decides not to take any action, or to *pass*. The outputs of LS-STM and LS-STM$_P$ are combined to yield LS-STM$_C$, the outputs of which, indexed by t, are

$$\text{LS-STM}_C(i) = \begin{cases} 1, & \text{if LS-STM}(t) = \text{LS-STM}_P(t) = 1 \\ -1, & \text{if LS-STM}(t) = \text{LS-STM}_P(t) = -1 \\ 0, & \text{otherwise} \end{cases} \tag{6.30}$$

In addition, the actual value of the LS-STM decision function defined in Equation (6.9) is incorporated to quantitatively define a confidence interval. This generates a thresholded LS-STM, denoted by LS-STM$_T$, the outputs of which are defined as

$$\text{LS-STM}_T(t) = \begin{cases} 1, & \text{if LS-STM}_C(t) = 1, \text{ and } y(\boldsymbol{\mathcal{X}}_t) > T \\ 0, & \text{otherwise} \end{cases} \tag{6.31}$$

where $T > 0$ is an arbitrary threshold. Notice that LS-STM$_T$ generates only *pass* or *long* signals. This has the advantage of avoiding *shorting* the S&P 500, which is generally a more complicated process than *going long*, and, from a practical perspective, incurs into higher transaction costs.

An example of the benefits of the added degree of confidence is shown in Figure 6.5, for the choice of $C = 0.001$ and $T = 0.07$. Notice how no activity is present during the crash of 2008, and how the generated profits always tend to keep a positive trend, even for 2015 onwards which was never the case for the results shown in Figure 6.3. The associated accuracy rate, SR$_A$ and σ_A are 55.23%, 0.64, and 0.10, respectively. This shows that introducing a degree of confidence in the decision making does in fact improve the LS-STM performance, generating very promising results for the considered application.

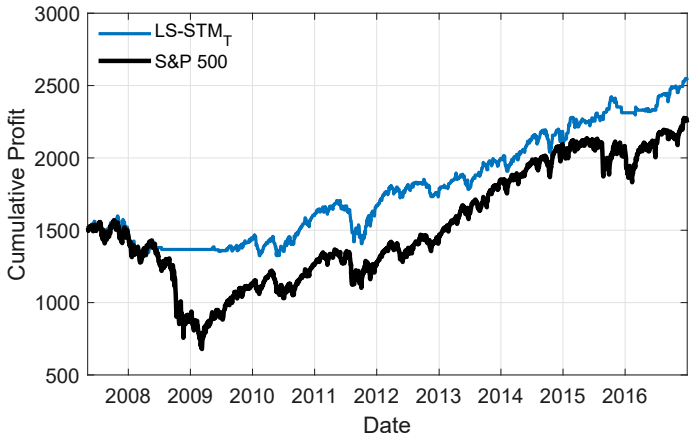


Figure 6.5: Cumulative profits generated using LS-STM$_T$ for $C = 0.001$ and $T = 0.07$.

6.6 Chapter Summary

The Support Tensor Machine (STM), a generalization of the Support Vector Machine (SVM) to tensor valued data, has been proposed for financial forecasting, in the form of its least-squares variant (LS-STM). A viable technique has been proposed for data tensorization, and a novel method to introduce kernels in LS-STM has been devised, allowing to employ the well-known Radial Basis Function (RBF) kernel. The proposed framework has been validated on the prediction of the daily direction of movement of the S&P 500 financial index for the period from January 2006 to January 2017, whereby the implied index of volatility, VIX, the gold commodity, GC1, and the S&P 500 itself have been employed as predictors. Simulation results have verified the superiority of LS-STM over SVM, while a probabilistic interpretation of the results via Platt scaling has demonstrated robustness to parameter changes, owing to the exploitation of cross-modal information which is inherent to tensors. Furthermore, it has been shown how performance can be improved by incorporating the output of the decision function of LS-STM as well as confidence measures. This resulted in a significant increase of the accuracy rate and Sharpe ratio, and at the same time lower volatility, which translates to generating robust profits based on a daily forecast of price movement for the S&P 500.

Conclusion to Part II

In **Part II**, tensors and Tensor Networks (TNs) have been applied to machine learning applications stemming from important signal processing concepts, such as feature fusion, eigenvalue extraction, and the Support Vector Machine (SVM). The TN summation framework draws an analogy with feature fusion, whereby the cores of the sum are a combination of the cores of the summand TNs, and provides a methodology to extract eigenvalues from large scale matrices. The Support Tensor Machine (STM) can similarly be interpreted as an ensemble of SVMs, each acting on one mode of the tensor, which harnesses the underlying data structure to make better predictions. The ability of tensors to exploit latent structure is further emphasized in the next and final part of this thesis, which is devoted to merging tensors with neural networks.

Part III

Tensors for Deep Learning

Chapter 7

Compression and Interpretability of Neural Networks via the Tucker Tensor Layer

> It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.
>
> *Carl Friedrich Gauss*

Chapter Overview

This chapter aims to help resolve the two main stumbling blocks in the application of Deep Neural Networks (DNNs), that is, the exceedingly large number of trainable parameters and their physical interpretability. This is achieved through a tensor valued approach, based on the proposed Tucker Tensor Layer (TTL), as an alternative to the dense weight-matrices of DNNs. This allows the weight-matrices of general DNNs to be treated as a matrix unfolding of a higher order weight-tensor. By virtue of the compression properties of tensor decompositions, this makes it possible to introduce a novel and efficient framework for exploiting the multi-way nature of the weight-tensor in order to dramatically reduce the number of DNN parameters. The tensor valued back-propagation algorithm is derived within the TTL framework, by extending the notion of matrix derivatives to tensors. In this way, the physical interpretability of the Tucker decomposition is exploited to gain physical insights into the NN training, through the process of computing gradients with respect to each factor matrix. The proposed framework is validated on both synthetic data, and the benchmark datasets MNIST, Fashion-MNIST, and CIFAR-10. Overall, through the ability to provide the relative importance of each data feature in training, the TTL back-propagation is shown to help mitigate the "black-box" nature inherent to NNs. Experiments also illustrate that the TTL achieves a 66.63-fold compression on MNIST and Fashion-MNIST, while, by simplifying the VGG-16 network, it achieves a 10% speed up in training time, at a comparable performance.

7.1 Introduction

Deep Neural Networks (DNNs) are the state-of-the-art machine-learning methodology which has delivered improved performance in numerous areas involving large-scale data, such as computer vision, speech recognition, and time-series analysis [145]. Within DNNs, Recurrent Neural Networks (RNNs) are among the most successful machine learning approaches for sequence modelling [146, 147], while Convolutional Neural Networks (CNNs) are particularly well-suited for image classification [148, 149]. However, despite the involvement of specialised

high performance hardware, the application of DNNs has also highlighted the necessity to reduce the typically very long processing times, as DNN models often comprise thousands of nodes and millions of learning parameters [150].

One way to tackle this issue is by employing a tensor approach to reduce the sheer scale of the DNN learning problem through the super-compression ability inherent to tensors. Indeed, owing to their compression properties, tensor decompositions (TDs) are just emerging as means to optimise NNs, by approximating an original "raw" network with its low-rank tensor representation which exhibits several orders of magnitude fewer parameters, while minimally affecting its overall performance.

In [57], the Tensor Train (TT) decomposition was employed to efficiently represent the dense weight matrix of the fully connected layers NNs, requiring only a few parameters. The ability of this approach to super-compress the NN weight matrix at a little sacrifice in accuracy was demonstrated on standard datasets. The work in [76] proposed the Tensor Contraction Layer (TCL) and Tensor Regression Layer (TRL) approaches, to reduce input dimensionality while preserving its multilinear tensor structure, and expressing the outputs as a mapping between two tensors. This work was extended in [77] with low-rank constraints imposed on the TRL, while the work in [78] tackled the question of improving training time by merging CNNs and the Canonical Polyadic Decomposition (CPD). Further research in the area, in particular involving the combination of the TKD with NNs, includes the work in [79] and [151]. In [79] the Tucker Decomposition (TKD) was used as a convolution kernel for mobile applications, while the authors in [151] adopt a more theoretical approach and propose a method to adaptively adjust the dimensions of the weight tensor in a layer-wise fashion. Both methods have been shown to achieve significant NN parameter compression and consequent reduction in training time.

Despite these advances, there is still a void in the literature when it comes to directly deriving the back-propagation algorithm for DNNs in terms of tensor latent factors within TDs, while at the same time providing physical meaning to the model. The authors of the seminal paper [57] are the only to have analytically re-derived back-propagation [87] in terms of the tensor factors stemming from the TT decomposition, however, the dimensionality and order of the employed TTs were arbitrarily chosen, so that, despite the achieved compression, the results were not physically interpretable. Most other research in the field tends to rely on automatic differentiation for back-propagation [76, 77, 79, 151]. Although this is convenient for implementation purposes, as automatic differentiation is quite efficient, there is still a necessity for a deep analytical understanding of how the errors are propagated; this is fundamental to the explainability of DNNs. In this work, this void is addressed by introducing the Tucker Tensor Layer (TTL) [88], which replaces the dense weight-matrix of a fully connected layer within DNNs with TKD factors. The TKD is chosen by virtue of its ability to provide low-rank representations of tensors while at the same time preserving the latent information within their structure [24]. Within this approach, the weight-matrix is treated as the unfolding (flattening) of a higher order weight-tensor, and back-propagation is fully derived in terms of the underlying tensor factors, that is, directly in the employed tensor format.

This work [88] therefore aims to introduce a novel analytical framework for the optimization of NNs through compression of their fully-connected layers, together with increasing the

interpretability of DNN models by exploiting the desirable properties of TKD. For rigour, the problem is addressed starting from the very core aspects, basing the analysis on the fundamentals of vector and matrix derivatives, and extending these to tensors, while providing rigorous mathematical arguments for their manipulation. In this way, this novel and theoretically well-founded derivation is demonstrated to be a prerequisite for obtaining physical insights into the training process. For example, it is shown that the ability of the proposed approach to keep track of the evolution of the gradients of each TKD factor during training makes it possible to reveal valuable information regarding the significance of input features within the DNN model. This, in turn, promises to be of high practical importance as:

1. New physically meaningful insights are provided into how NNs perform classification, thus mitigating their inherent well-known "black-box" nature;

2. Inherent means to perform computationally cheap data augmentation follow naturally.

The proposed approach is verified through application case studies on synthetic data and the benchmark MNIST [152], Fashion-MNIST [153], and CIFAR-10 [154] datasets. In addition, owing to its compressive properties, the TTL achieves a 66.63 fold compression for NNs trained on MNIST and Fashion-MNIST. It also allows to simplify the VGG-16 CNN [149] applied to CIFAR-10, resulting in a 10% training time speed up, at a little accuracy trade-off thanks to data augmentation based on gradient information. Comparable performance to the uncompressed NNs is achieved in all cases.

The contributions of this chapter can be summarized as follows:

1. The proposed Tucker Tensor Layer (TTL) is introduced and a full analytic derivation of back-propagation directly in terms of the TKD factors is provided;

2. it is demonstrated that the TTL can achieve significant compression of DNN parameters while maintaining comparable performance to uncompressed NNs;

3. By leveraging on the developed theoretical framework, novel insights into the training process are provided via gradient inspection, hence offering a new interpretability to results;

4. The information from point (3) is employed to perform cheap data augmentation, ultimately leading to the simplification and training speed-up of more complex NNs.

7.2 Preliminaries

This section provides additional theoretical background required to develop the theory behind the proposed TTL.

7.2.1 Properties of the Kronecker Product

If $\mathbf{A} \in \mathbb{R}^{M \times N}$ and $\mathbf{B} \in \mathbb{R}^{P \times Q}$, then the operation $\mathbf{A} \otimes \mathbf{B}$ yields a matrix $\mathbf{C} \in \mathbb{R}^{MP \times NQ}$, with elements $c_{P(r-1)+v,Q(s-1)+w} = a_{rs}b_{vw}$. The Kronecker product is a bilinear and associative

operator, that is

$$\begin{aligned}
&\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C} \\
&(\mathbf{A} + \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{C} \\
&(k\mathbf{A}) \otimes \mathbf{B} = \mathbf{A} \otimes (k\mathbf{B}) = k(\mathbf{A} \otimes \mathbf{B}) \\
&(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})
\end{aligned} \tag{7.1}$$

The Kronecker product is non-commutative

$$\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A} \tag{7.2}$$

but it is permutation equivalent, that is

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{P}(\mathbf{B} \otimes \mathbf{A})\mathbf{Q} \tag{7.3}$$

for some permutation matrices $\mathbf{P}$ and $\mathbf{Q}$. Throughout this chapter, the transpose property of the Kronecker product,

$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T \tag{7.4}$$

and the following identity

$$\text{vec}(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}) \tag{7.5}$$

will be extensively used.

7.2.2 Vector and Matrix Derivatives

Here the theory related to vector and matrix derivatives is presented, which will be extended to tensors in later sections. For convenience, a similar notation as in [155] is adopted.

Definition 7.1. *Let ϕ be a differentiable scalar function of a vector $\mathbf{x} \in \mathbb{R}^{N\times 1}$. Then the vector in $\mathbb{R}^{N\times 1}$*

$$D\phi(\mathbf{x}) = \frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial \phi(x_1)}{\partial x_1}, \quad \frac{\partial \phi(x_2)}{\partial x_2}, \quad \ldots, \quad \frac{\partial \phi(x_N)}{\partial x_N}\right]^T \tag{7.6}$$

is called the derivative of ϕ. If f is an $M \times 1$ vector function of $\mathbf{x}$, i.e. $f(\mathbf{x}) \in \mathbb{R}^{M\times 1}$, then the matrix in $\mathbb{R}^{M\times N}$

$$Df(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \Big[Df_1(\mathbf{x}),\ Df_2(\mathbf{x}),\ \ldots,\ Df_M(\mathbf{x})\Big]^T \tag{7.7}$$

is called the derivative or Jacobian of f.

Definition 7.2. *A ball in $\mathbb{R}^{N\times Q}$ of radius r is denoted by*

$$B(\mathbf{C}; r) = \{\mathbf{X} : \mathbf{X} \in \mathbb{R}^{N\times Q}, ||\mathbf{X} - \mathbf{C}||_F < r\} \tag{7.8}$$

where $\mathbf{C} \in \mathbb{R}^{N\times Q}$ is an interior point of a set S in $\mathbb{R}^{N\times Q}$.

Definition 7.3. *Let S be a set in $\mathbb{R}^{N\times Q}$, and let $F : S \mapsto \mathbb{R}^{M\times P}$ be a matrix function operating on S. Let the matrix $\mathbf{C} \in \mathbb{R}^{N\times Q}$ be an interior point of S, and let $B(\mathbf{C}; r) \subset S$ be a ball with centre $\mathbf{C}$ and radius r. Let $\mathbf{E}$ be an arbitrary matrix in $\mathbb{R}^{N\times Q}$, with $||\mathbf{E}||_F < r$, so that $(\mathbf{C} + \mathbf{E}) \in B(\mathbf{C}; r)$. If there*

exist a matrix $\mathbf{A} \in \mathbb{R}^{MP \times NQ}$*, which depends on* $\mathbf{C}$ *but not on* $\mathbf{E}$ *so that*

$$vec(F(\mathbf{C}+\mathbf{E})) = vec(F(\mathbf{C})) + \mathbf{A}(\mathbf{C})vec(\mathbf{E}) + vec\big(R_{\mathbf{C}}(\mathbf{E})\big) \tag{7.9}$$

for all $\mathbf{E} \in \mathbb{R}^{N \times Q}$ *with* $||\mathbf{E}||_F < r$*, where the remainder* $R_{\mathbf{C}}(\mathbf{E})$ *is of smaller order than* $||\mathbf{E}||_F$ *as* $\mathbf{E} \to \mathbf{0}$*, that is*

$$\lim_{\mathbf{E}\to\mathbf{0}} \frac{R_{\mathbf{C}}(\mathbf{E})}{||\mathbf{E}||_F} = \mathbf{0} \tag{7.10}$$

then the function F *is said to be differentiable at* $\mathbf{C}$*. Define*

$$\mathbf{A}(\mathbf{C})vec(\mathbf{E}) = vec\big(dF(\mathbf{C};\mathbf{E})\big) \tag{7.11}$$

then the matrix $dF(\mathbf{C};\mathbf{E}) \in \mathbb{R}^{M \times P}$ *is called the first differential of* F *at* $\mathbf{C}$ *with increment* $\mathbf{E}$*.*

In view of the previous definition, the calculus properties of vector functions can be immediately extended to the matrix case, because, instead of considering the matrix function $F : S \mapsto \mathbb{R}^{M \times P}$ we can consider the vector function $f : \text{vec}(S) \mapsto \mathbb{R}^{MP \times 1}$ defined by

$$f(\text{vec}(\mathbf{X})) = \text{vec}(F(\mathbf{X})) \tag{7.12}$$

Then the differentials of F and f are related by

$$\text{vec}(dF(\mathbf{C};\mathbf{E})) = df(\text{vec}(\mathbf{C}); \text{vec}(\mathbf{E})) \tag{7.13}$$

and this justifies the following definition.

Definition 7.4. *Consider a matrix* $\mathbf{X} \in \mathbb{R}^{N \times Q}$ *and the differentiable matrix function* $F : \mathbf{X} \mapsto \mathbb{R}^{M \times P}$*. Then the matrix in* $\mathbb{R}^{MP \times NQ}$

$$\frac{\partial F(\mathbf{X})}{\partial \mathbf{X}} = D(F(\mathbf{X})) = Df(vec(\mathbf{X})) = \frac{\partial f(vec(\mathbf{X}))}{\partial vec(\mathbf{X})} \tag{7.14}$$

is called the derivative or Jacobian of F *at* $\mathbf{X}$*.*

Remark 7.1. From Definition 7.3 and the "identification theorems" in [155], it follows that the conditions on the first differential in (7.11) hold if and only if $\mathbf{A}(\mathbf{C}) = D(F(\mathbf{C}))$.

For example, let $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{M \times Q}$ be matrix constants, and $\mathbf{X} \in \mathbb{R}^{J \times M}$ be a matrix variable. Define $F(\mathbf{X}) = \mathbf{AXB}$. Then,

$$\begin{aligned} DF(\mathbf{X}) = \frac{\partial f(\text{vec}(\mathbf{X}))}{\partial \text{vec}(\mathbf{X})} &= \frac{\partial (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X})}{\partial \text{vec}(\mathbf{X})} \\ &= (\mathbf{B}^T \otimes \mathbf{A}) \in \mathbb{R}^{QI \times MJ} \end{aligned} \tag{7.15}$$

7.3 Tensor Derivatives

To map the parameters of DNNs into the proposed tensor framework, it is a prerequisite to extend the notion of vector and matrix derivatives to higher-order tensors. To this end, Defini-

tions 7.2 to 7.4 need to reformulated as in the following.

Definition 7.5. *A ball in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of radius r and centre $\boldsymbol{\mathcal{C}}$ is denoted by*

$$B(\boldsymbol{\mathcal{C}}; r) = \{\boldsymbol{\mathcal{X}} : \boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, ||\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{C}}||_F < r\} \tag{7.16}$$

where $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an interior point of a set S in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

Definition 7.6. *Let S be a set in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and let $F : S \mapsto \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ be a tensor-valued function operating on S. Notice that, in general, $N \neq M$, that is, F does not necessarily map a tensor to a tensor of the same order. Let the tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an interior point of S, and let $B(\boldsymbol{\mathcal{C}}; r) \subset S$ be a ball with centre $\boldsymbol{\mathcal{C}}$ and radius r. Let $\boldsymbol{\mathcal{E}}$ be an arbitrary tensor in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with $||\boldsymbol{\mathcal{E}}||_F < r$, so that $(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{E}}) \in B(\boldsymbol{\mathcal{C}}; r)$. If there exists a matrix $\mathbf{A} \in \mathbb{R}^{J_1 J_2 \cdots J_M \times I_1 I_2 \cdots I_N}$, which depends on $\boldsymbol{\mathcal{C}}$ but not on $\boldsymbol{\mathcal{E}}$, so that*

$$vec(F(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{E}})) = vec(F(\boldsymbol{\mathcal{C}})) + \mathbf{A}(\boldsymbol{\mathcal{C}}) vec(\boldsymbol{\mathcal{E}}) + vec\big(R_{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{E}})\big) \tag{7.17}$$

for all $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $||\boldsymbol{\mathcal{E}}||_F < r$, where the remainder, $R_{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{E}})$, is of a smaller order than $||\boldsymbol{\mathcal{E}}||_F$ as $\boldsymbol{\mathcal{E}} \to 0$, that is

$$\lim_{\boldsymbol{\mathcal{E}} \to \mathbf{0}} \frac{R_{\boldsymbol{\mathcal{C}}}(\boldsymbol{\mathcal{E}})}{||\boldsymbol{\mathcal{E}}||_F} = \mathbf{0} \tag{7.18}$$

then the function F is said to be differentiable at $\boldsymbol{\mathcal{C}}$. Define

$$\mathbf{A}(\boldsymbol{\mathcal{C}}) vec(\boldsymbol{\mathcal{E}}) = vec\big(dF(\boldsymbol{\mathcal{C}}; \boldsymbol{\mathcal{E}})\big) \in \mathbb{R}^{J_1 J_2 \cdots J_M} \tag{7.19}$$

then the tensor $dF(\boldsymbol{\mathcal{C}}; \boldsymbol{\mathcal{E}}) \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ is called the first differential of F at $\boldsymbol{\mathcal{C}}$ with increment $\boldsymbol{\mathcal{E}}$.

In view of Definition 7.6, the properties of vector calculus can be readily extended to tensors, because, instead of considering the tensor function $F : S \mapsto \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$, the vector function $f : \text{vec}(S) \mapsto \mathbb{R}^{J_1 J_2 \cdots J_M \times 1}$ defined by

$$f(\text{vec}(\boldsymbol{\mathcal{X}})) = \text{vec}(F(\boldsymbol{\mathcal{X}})) \tag{7.20}$$

may be considered. Then it follows that

$$\text{vec}(dF(\boldsymbol{\mathcal{C}}; \boldsymbol{\mathcal{E}})) = df(\text{vec}(\boldsymbol{\mathcal{C}}); \text{vec}(\boldsymbol{\mathcal{E}})) \tag{7.21}$$

This justifies the following definition.

Definition 7.7. *Consider a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and the differentiable tensor function $F : \boldsymbol{\mathcal{X}} \mapsto \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$. Then, the matrix in $\mathbb{R}^{J_1 J_2 \cdots J_M \times I_1 I_2 \cdots I_N}$, given by*

$$\frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \boldsymbol{\mathcal{X}}} = D(F(\boldsymbol{\mathcal{X}})) = Df(vec(\boldsymbol{\mathcal{X}})) = \frac{\partial f(vec(\boldsymbol{\mathcal{X}}))}{\partial vec(\boldsymbol{\mathcal{X}})} \tag{7.22}$$

is referred to as the derivative of F at $\boldsymbol{\mathcal{X}}$.

Remark 7.2. From Definition 7.6 and the "identification theorems" in [155], it follows that the conditions on the first differential in (7.19) hold if and only if $\mathbf{A}(\boldsymbol{\mathcal{C}}) = D(F(\boldsymbol{\mathcal{C}}))$.

With the above analysis, a connection between tensor differentiability and the permutation of tensor entries can now be established.

Theorem 7.1. Consider a matrix $\mathbf{X} \in \mathbb{R}^{M\times P}$ and the differentiable tensor function $F : \mathbf{X} \mapsto \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. Consider the tensor mapping $\boldsymbol{\mathcal{Y}} = F(\mathbf{X})$, and denote by $\mathbf{Y}_{(n)}$ the mode-n unfolding of $\boldsymbol{\mathcal{Y}}$. Then $\forall n \in \{1, 2, \cdots, N\}$, $D(\mathbf{Y}_{(n)})$ is a permuted version of $D(\boldsymbol{\mathcal{Y}}) = D(F(\boldsymbol{\mathcal{X}}))$, i.e. $D(\boldsymbol{\mathcal{Y}})$ and $D(\mathbf{Y}_{(n)})$ are matrices containing the same elements, but arranged differently. This can be written as

$$D(\mathbf{Y}_{(n)}) = \mathcal{P}(D(\boldsymbol{\mathcal{Y}})) \tag{7.23}$$

where $\mathcal{P}$ is a permutation operator.

Proof. Let $\mathbf{X} \in \mathbb{R}^{M\times P}$ and $\boldsymbol{\mathcal{Y}} = F(\mathbf{X}) \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. From Definition 7.4,

$$D(\boldsymbol{\mathcal{Y}}) = \frac{\partial \boldsymbol{\mathcal{Y}}}{\partial \mathbf{X}} = \frac{\partial \text{vec}(\boldsymbol{\mathcal{Y}})}{\partial \text{vec}(\mathbf{X})} \in \mathbb{R}^{I_1 I_2\cdots I_N\times MP} \tag{7.24}$$

At the same time, this results in $\mathbf{Y}_{(n)} \in \mathbb{R}^{I_n\times I_1 I_2\cdots I_{n-1} I_{n+1}\cdots I_N}$. Next,

$$D(\mathbf{Y}_{(n)}) = \frac{\partial \mathbf{Y}_{(n)}}{\partial \mathbf{X}} = \frac{\partial \text{vec}(\mathbf{Y}_{(n)})}{\partial \text{vec}(\mathbf{X})} \in \mathbb{R}^{I_n I_1\cdots I_{n-1} I_{n+1}\cdots I_N\times MP} \tag{7.25}$$

Hence, $D(\boldsymbol{\mathcal{Y}})$ and $D(\mathbf{Y}_{(n)})$ have the same dimensionality. Because $\text{vec}(\boldsymbol{\mathcal{Y}})$ and $\text{vec}(\mathbf{Y}_{(n)})$ clearly have the same elements but arranged differently, $D(\mathbf{Y}_{(n)})$ is a permuted version of $D(\boldsymbol{\mathcal{Y}})$. □

Corollary 1. Consider a matrix $\mathbf{X} \in \mathbb{R}^{M\times P}$ and the differentiable tensor function $F : \mathbf{X} \mapsto \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. Let $\boldsymbol{\mathcal{Y}} = F(\boldsymbol{\mathcal{X}})$, and denote by $\mathbf{Y}_{(n)}$ and $\mathbf{Y}_{(m)}$ respectively the mode-n and mode-m unfoldings of $\boldsymbol{\mathcal{Y}}$. Then $D(\mathbf{Y}_{(n)})$ and $D(\mathbf{Y}_{(m)})$ are of the same dimensionality, but with permuted elements. This can be formalized as

$$D(\mathbf{Y}_{(n)}) = \mathcal{P}(D(\mathbf{Y}_{(m)})) \tag{7.26}$$

where $\mathcal{P}$ is a permutation operator.

Proof. Follows immediately from Theorem 7.1. □

7.4 The Tucker Tensor Layer and Tensor-Valued Back-Propagation

In this section, the Tucker Tensor Layer (TTL) is introduced. Tensor-valued back-propagation is derived, and the analytically obtained gradients are rigorously verified. For simplicity, non-linearities shall initially be omitted, although they are considered later.

7.4.1 Representation

In general, a fully connected layer within a NN can be expressed as

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{7.27}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the output, $\mathbf{x} \in \mathbb{R}^P$ the input, and $\mathbf{W} \in \mathbb{R}^{M \times P}$ is the connecting matrix of weights. Now, consider an input tensor of order N, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and a weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N+1}}$, of order $(N+1)$. The proposed tensor-valued derivation of back-propagation will be based on the investigation of the function $F : \boldsymbol{\mathcal{X}} \mapsto \mathbb{R}^{I_{N+1}}$, where I_{N+1} is the number of classes, defined by

$$F(\boldsymbol{\mathcal{X}}) = \mathbf{W}_{(N+1)}\text{vec}(\boldsymbol{\mathcal{X}}) + \mathbf{b} \tag{7.28}$$

where $\mathbf{b} \in \mathbb{R}^{I_{N+1}}$ is a bias vector, $\mathbf{W}_{(N+1)} \in \mathbb{R}^{I_{N+1} \times I_1 I_2 \cdots I_N}$ is the $(N+1)$-mode unfolding of tensor $\boldsymbol{\mathcal{W}}$, and $\text{vec}(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{I_1 I_2 \cdots I_N}$ is the vectorization of tensor $\boldsymbol{\mathcal{X}}$. As shown in the following, the representation in (7.28) has the advantage of allowing for a compressed version of the NN layer, thanks to tensor decompositions.

Recall that a tensor $\boldsymbol{\mathcal{W}}$ can be represented in the TKD format as

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_{N+1} \mathbf{A}^{(N+1)} \tag{7.29}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_{N+1}}$ is the core tensor, and $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ the corresponding factor matrices. This implies that the $(N+1)$-mode matrix unfolding of $\boldsymbol{\mathcal{W}}$, denoted by $\mathbf{W}_{(N+1)}$, can be expressed as

$$\begin{aligned} \mathbf{W}_{(N+1)} &= \mathbf{A}^{(N+1)}\mathbf{G}_{(N+1)}\left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^T \\ &= \mathbf{A}^{(N+1)}\mathbf{G}_{(N+1)} \bigotimes_{i=N}^{1} \mathbf{A}^{(i)T} \end{aligned} \tag{7.30}$$

Substituting (7.30) into (7.28) yields

$$F(\boldsymbol{\mathcal{X}}) = \left[\mathbf{A}^{(N+1)}\mathbf{G}_{(N+1)} \bigotimes_{i=N}^{1} \mathbf{A}^{(i)T}\right]\text{vec}(\boldsymbol{\mathcal{X}}) + \mathbf{b} \tag{7.31}$$

Remark 7.3. The compression of the DNN parameters through the TTL approach is achieved by selecting the size of the modes of the core tensor $\boldsymbol{\mathcal{G}}$ within the TKD to be smaller than those of the weight tensor $\boldsymbol{\mathcal{W}}$, that is $R_n < I_n$, for $n = 1, 2, \ldots, N+1$.

7.4.2 Learning via Tensor-Valued Back-Propagation

Neural networks are generally trained with stochastic gradient descent algorithms, where at each step the gradient is computed using the back-propagation procedure [87]. Back-propagation starts by computing the gradient of a loss function L with respect to the NN output, then, given the gradient $\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})}$, proceeds sequentially through the layers of the NN, but in a reversed order. When applied to the fully connected layer as expressed in (7.28) back-propagation computes

[57]

$$\begin{aligned}\frac{\partial L}{\partial \text{vec}(\boldsymbol{\mathcal{X}})} &= \mathbf{W}_{(N+1)}^T \frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \\ \frac{\partial L}{\partial \mathbf{b}} &= \frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \\ \frac{\partial L}{\partial \mathbf{W}_{(N+1)}} &= \frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \text{vec}(\boldsymbol{\mathcal{X}})^T\end{aligned} \tag{7.32}$$

Since L is a scalar, the gradients in (7.32) are of dimensionality

$$\begin{aligned}\frac{\partial L}{\partial \text{vec}(\boldsymbol{\mathcal{X}})} &\in \mathbb{R}^{I_1 I_2 \cdots I_N} \\ \frac{\partial L}{\partial \mathbf{b}} &\in \mathbb{R}^{I_{N+1}} \\ \frac{\partial L}{\partial \mathbf{W}_{(N+1)}} &\in \mathbb{R}^{I_{N+1} \times I_1 I_2 \cdots I_N}\end{aligned} \tag{7.33}$$

This allows, at each iteration of the algorithm, for an update of the weight matrix as

$$\mathbf{W}_{(N+1)_{t+1}} = \mathbf{W}_{(N+1)_t} + \mu \frac{\partial L}{\partial \mathbf{W}_{(N+1)_t}} \tag{7.34}$$

where μ is a step size.

To derive the tensor-valued back-propagation based on the Tucker model in (7.29), first note that the gradient $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ is a function of $\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})}$, $\frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}}$, $\frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{A}^{(n)}}$, to give the derivative

$$\frac{\partial L}{\partial \mathbf{A}^{(n)}} = \Omega\left(\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})}, \frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}}, \frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{A}^{(n)}}\right) \tag{7.35}$$

Then, define the following shorthand notation for the dimensionalities involved:

$$\begin{aligned}\underline{R}_{-n} &= R_1 R_2 \cdots R_{n-1} R_{n+1} \cdots R_{N+1} \\ \underline{I}_{-n} &= I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_{N+1} \\ \underline{I}_N &= I_1 I_2 \cdots I_N \\ \underline{I}_{N+1} &= I_1 I_2 \cdots I_{N+1}\end{aligned} \tag{7.36}$$

From the rules of the Kronecker product,

$$\frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}} = \text{vec}(\boldsymbol{\mathcal{X}})^T \otimes \mathbf{I}_{I_{N+1}} \in \mathbb{R}^{I_{N+1} \times \underline{I}_{N+1}} \tag{7.37}$$

A closer inspection of equation (7.35) shows that $\frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{\underline{I}_{N+1} \times R_n I_n}$, by Corollary 1, can be found as a permutation of $\frac{\partial \mathbf{W}_{(n)}}{\partial \mathbf{A}^{(n)}}$, which is a matrix of the same dimensionality and containing the same elements. In other words,

$$\frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{A}^{(n)}} = \mathbf{P}\left(\frac{\partial \mathbf{W}_{(n)}}{\partial \mathbf{A}^{(n)}}\right) = \mathbf{P}_n \frac{\partial \mathbf{W}_{(n)}}{\partial \mathbf{A}^{(n)}} \tag{7.38}$$

where $\mathbf{P}_n \in \mathbb{R}^{\underline{I}_{N+1} \times \underline{I}_{N+1}}$ is a permutation matrix that satisfies

$$\text{vec}(\mathbf{W}_{(n)}) = \mathbf{P}_n \text{vec}(\mathbf{W}_{(N+1)}) \tag{7.39}$$

The task hence boils down to computing

$$\frac{\partial L}{\partial \mathbf{A}^{(n)}} = \Omega\left(\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})}, \frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}}, \mathbf{P}_n \frac{\partial \mathbf{W}_{(n)}}{\partial \mathbf{A}_{(n)}}\right) \tag{7.40}$$

The mode-n unfolding of the weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N+1}}$ is given by

$$\begin{aligned} \mathbf{W}_{(n)} &= \mathbf{A}^{(n)} \mathbf{G}_{(n)} \left(\mathbf{A}^{(N+1)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^T \\ &= \mathbf{A}^{(n)} \mathbf{G}_{(n)} \bigotimes_{\substack{i=N+1 \\ i \neq n}}^{1} \mathbf{A}^{(i)T} \in \mathbb{R}^{I_n \times \underline{I}_{-n}} \end{aligned} \tag{7.41}$$

while its partial derivative with respect to $\mathbf{A}^{(n)}$ now becomes

$$\begin{aligned} \frac{\partial \mathbf{W}_{(n)}}{\partial \mathbf{A}^{(n)}} &= \left(\mathbf{A}^{(N+1)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right) \mathbf{G}_{(n)}^T \otimes \mathbf{I}_{I_n} \\ &= \bigotimes_{\substack{i=N+1 \\ i \neq n}}^{1} \mathbf{A}^{(i)} \mathbf{G}_{(n)}^T \otimes \mathbf{I}_{I_n} \in \mathbb{R}^{\underline{I}_{N+1} \times R_n I_n} \end{aligned} \tag{7.42}$$

From (7.35), $\frac{\partial L}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{I_n \times R_n}$ is required, which is the matrix of $\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})} \in \mathbb{R}^{R_n I_n}$. Therefore, to find $\frac{\partial L}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{I_n \times R_n}$, it is sufficient to compute $\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})} \in \mathbb{R}^{R_n I_n}$ and reshape the result accordingly. Since

$$\begin{aligned} &\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \in \mathbb{R}^{I_{N+1}} \\ &\frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}} \in \mathbb{R}^{I_{N+1} \times \underline{I}_{N+1}} \\ &\frac{\partial \mathbf{W}(n)}{\partial \mathbf{A}_{(n)}} \in \mathbb{R}^{\underline{I}_{N+1} \times R_n I_n} \\ &\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})} \in \mathbb{R}^{R_n I_n} \end{aligned} \tag{7.43}$$

this results in

$$\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})} = \left(\mathbf{P}_n \frac{\partial \mathbf{W}(n)}{\partial \mathbf{A}_{(n)}}\right)^T \frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}}^T \frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \tag{7.44}$$

Similarly, since $\frac{\partial L}{\partial \mathbf{G}_{(N+1)}} \in \mathbb{R}^{R_{N+1} \times \underline{R}_N}$ can be computed by considering $\frac{\partial L}{\partial \text{vec}(\mathbf{G}_{(N+1)})} \in \mathbb{R}^{\underline{R}_{N+1}}$, then

$$\frac{\partial L}{\partial \text{vec}(\mathbf{G}_{(N+1)})} = \frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{G}_{(N+1)}}^T \frac{\partial F(\boldsymbol{\mathcal{X}})}{\partial \mathbf{W}_{(N+1)}}^T \frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} \tag{7.45}$$

where,

$$\frac{\partial \mathbf{W}_{(N+1)}}{\partial \mathbf{G}_{(N+1)}} = (\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(1)}) \otimes \mathbf{A}^{(N+1)} \in \mathbb{R}^{\underline{I}_{N+1} \times \underline{R}_{N+1}} \tag{7.46}$$

7.4.3 Gradient Verification

Equations (7.44) and (7.45) can be verified based on Definition 7.6. In particular, defining the cost function as a general function

$$L : (F(\boldsymbol{\mathcal{X}}), \mathbf{d}) \mapsto \mathbb{R} \tag{7.47}$$

where $\mathbf{d}$ is a desired output and $F(\boldsymbol{\mathcal{X}})$ is defined as in (7.31), the gradient $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ (i.e. a permuted version of $\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})}$) can be verified by treating all inputs to L as constants except for $\mathbf{A}^{(n)}$, and confirming that

$$\lim_{\mathbf{E}\to\mathbf{0}} \frac{R_{\mathbf{A}^{(n)}}(\mathbf{E})}{||\mathbf{E}||_F} = \mathbf{0} \tag{7.48}$$

where, from Remark 7.1,

$$\begin{aligned} R_{\mathbf{A}^{(n)}}(\mathbf{E}) &= L(\mathbf{A}^{(n)} + \mathbf{E}) - L(\mathbf{A}^{(n)}) - \mathbf{A}(\mathbf{A}^{(n)})\text{vec}(\mathbf{E}) \\ &= L(\mathbf{A}^{(n)} + \mathbf{E}) - L(\mathbf{A}^{(n)}) - \left(\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)})}\right)^T \text{vec}(\mathbf{E}) \end{aligned} \tag{7.49}$$

Similarly, to verify the gradient $\frac{\partial L}{\partial \boldsymbol{\mathcal{G}}}$ (i.e. a permuted version of $\frac{\partial L}{\partial \text{vec}(\mathbf{G}_{(N+1)})}$), it is necessary to treat all inputs to L as constants, except for $\boldsymbol{\mathcal{G}}$, and confirm that

$$\lim_{\boldsymbol{\mathcal{E}}\to\mathbf{0}} \frac{R_{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{E}})}{||\boldsymbol{\mathcal{E}}||_F} = \mathbf{0} \tag{7.50}$$

where

$$\begin{aligned} R_{\boldsymbol{\mathcal{G}}}(\boldsymbol{\mathcal{E}}) &= L(\boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}}) - L(\boldsymbol{\mathcal{G}}) - \mathbf{A}(\boldsymbol{\mathcal{G}})\text{vec}(\boldsymbol{\mathcal{E}}) \\ &= L(\boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}}) - L(\boldsymbol{\mathcal{G}}) - \left(\frac{\partial L}{\partial \text{vec}(\boldsymbol{\mathcal{G}})}\right)^T \text{vec}(\boldsymbol{\mathcal{E}}) \end{aligned} \tag{7.51}$$

To demonstrate the correctness of this approach, a 3-rd order data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{5\times5\times5}$ was defined, the elements of which were drawn from a normal distribution, such that

$$\text{vec}(\boldsymbol{\mathcal{X}}) \sim \mathcal{N}(0, 1) \tag{7.52}$$

and a 4-th order weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{5\times5\times5\times3}$, also with elements drawn from a normal distribution, i.e.

$$\text{vec}(\boldsymbol{\mathcal{W}}) \sim \mathcal{N}(0, 1) \tag{7.53}$$

From $\boldsymbol{\mathcal{W}}$, the core tensor $\boldsymbol{\mathcal{G}}$ and the factor matrices $\mathbf{A}^{(n)}$, $n = 1, \ldots, 4$ were obtained through a TKD with multi-linear rank $(5, 5, 5, 3)$. The desired output $\mathbf{d} \in \mathbb{R}^3$ was arbitrarily assigned, and the squared error was employed as a cost function, that is

$$L = \frac{1}{2}\left|\left|F(\boldsymbol{\mathcal{X}}) - \mathbf{d}\right|\right|_F^2 \tag{7.54}$$

implying

$$\frac{\partial L}{\partial F(\boldsymbol{\mathcal{X}})} = F(\boldsymbol{\mathcal{X}}) - \mathbf{d} \tag{7.55}$$

The gradients were computed according to the procedure in Section 7.4.2, and were verified according to (7.48)-(7.51). The results are shown in Figure 7.1, and conform with the above analysis.

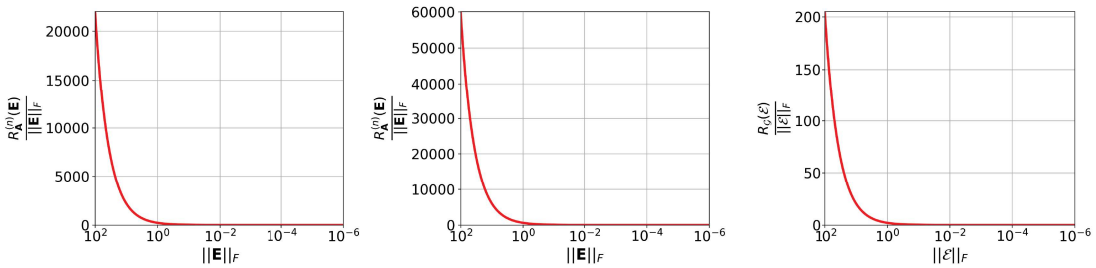


Figure 7.1: Verification of the gradients by confirming that the remainder tends to 0 faster than the magnitude of the perturbation, as per (7.48) and (7.50). Left: Gradient verification for $\frac{\partial L}{\partial \mathbf{A}^{(2)}}$. Centre: Gradient verification for $\frac{\partial L}{\partial \mathbf{A}^{(4)}}$. Right: Gradient verification for $\frac{\partial L}{\partial \mathcal{G}}$. Notice the inverted abscissa and the logarithmic scale.

7.5 Algorithm Implementation

To illustrate the algorithmic implementation of the results obtained in Section 7.4, consider a dataset consisting of N-th order tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and their corresponding labels, that is $\{\boldsymbol{\mathcal{X}}_m, y_m\}_m^M$, $m = 1, \ldots, M$, where M is the size of the dataset. For an efficient representation, the data are organized in a matrix $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_N \times M}$, defined as

$$\underline{\mathbf{X}} = \Big[\text{vec}(\boldsymbol{\mathcal{X}}_1), \quad \text{vec}(\boldsymbol{\mathcal{X}}_2), \quad \ldots, \quad \text{vec}(\boldsymbol{\mathcal{X}}_M)\Big] \tag{7.56}$$

so that Equation (7.28) can be re-written as

$$F(\underline{\mathbf{X}}) = \mathbf{W}_{(N+1)}\underline{\mathbf{X}} + \mathbf{B} \equiv \underline{\mathbf{F}} \in \mathbb{R}^{I_{N+1} \times M} \tag{7.57}$$

where $\mathbf{B} \in \mathbb{R}^{I_{N+1} \times M}$ is a matrix of biases. Note that $\mathbf{B}^{(l)} \in \mathbb{R}^{I_{N+1}^{(l)} \times M}$ must apply the same biases to each element in the dataset, and, as such, has the structure of $\mathbf{B}^{(l)} = \mathbf{b}^{(l)}\mathbf{1}_M^T$

$$\mathbf{B}^{(l)} = \mathbf{b}^{(l)}\mathbf{1}_M^T \tag{7.58}$$

where $\mathbf{b}^{(l)} \in \mathbb{R}^{I_{N+1}^{(l)}}$ are the actual biases.

Consider now a neural network composed of layers indexed by $l = 0, 1, 2, \ldots, L$, where $l = 0$ is the input layer, $l = L$ is the output layer, and $l = 1, \ldots, L-1$ are the hidden layers. For generality, assume that all layers are TTLs. With a slight modification in our notation, for each layer we denote inputs and outputs as summarized in Table 7.1.

Upon calculating the gradient of the cost function L w.r.t. the input of the last layer of the network $\underline{\mathbf{Z}}^{(L)}$, that is

$$\frac{\partial L}{\partial \underline{\mathbf{F}}^{(L)}} \in \mathbb{R}^{I_{N+1}^{(L)} \times M} \tag{7.59}$$

the error is then back-propagated through layers $l = L-1, L-2, \ldots, 1$. As a result the gradients

Table 7.1: Inputs and outputs of network layers.

$\underline{\mathbf{Z}}^{(0)} = \underline{\mathbf{X}}$	Output of layer $l = 0$: the input to the network
$\mathbf{W}^{(l)}_{(N+1)}$	Weight matrix for layers $l = 1, \ldots, L$
$\mathbf{A}^{(n)}_{(l)}$, $\mathbf{G}^{(l)}_{(N+1)}$	Factor matrices and core tensor associated to $\mathbf{W}^{(l)}_{(N+1)}$ for layers $l = 1, \ldots, L$
$\underline{\mathbf{F}}^{(l)} = \mathbf{W}^{(l)}_{(N+1)}\underline{\mathbf{Z}}^{(l-1)} + \mathbf{B}^{(l)}$	Input to layers $l = 1, \ldots, L$
$\underline{\mathbf{Z}}^{(l)} = \sigma(\underline{\mathbf{F}}^{(l)})$	Output of layers $l = 1, \ldots, L$, where $\sigma(\cdot)$ is a point-wise activation function

of the cost function with respect to the factor matrices and the core tensor of the l-th Tucker tensor layer are computed as

$$\begin{aligned}\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)}_{(l)})} &= \left(\mathbf{P}_n \frac{\partial \mathbf{W}^{(l)}_{(n)}}{\partial \mathbf{A}^{(n)}_{(l)}}\right)^T \frac{\partial \underline{\mathbf{F}}^{(l)}}{\partial \mathbf{W}^{(l)}_{(N+1)}}^T \text{vec}(\underline{\mathbf{D}}^{(l)}) \\ \frac{\partial L}{\partial \text{vec}(\mathbf{G}^{(l)}_{(N+1)})} &= \frac{\partial \mathbf{W}^{(l)}_{(N+1)}}{\partial \mathbf{G}^{(l)}_{(N+1)}}^T \frac{\partial \underline{\mathbf{F}}^{(l)}}{\partial \mathbf{W}^{(l)}_{(N+1)}}^T \text{vec}(\underline{\mathbf{D}}^{(l)})\end{aligned} \tag{7.60}$$

where

$$\underline{\mathbf{D}}^{(l)} = \begin{cases} \frac{\partial L}{\partial \underline{\mathbf{Z}}^{(L)}} \frac{\partial \underline{\mathbf{Z}}^{(L)}}{\partial \underline{\mathbf{F}}^{(L)}} = \frac{\partial L}{\partial \underline{\mathbf{F}}^{(L)}}, & \text{if } l = L \\ \mathbf{W}^{(l+1)T}_{(N+1)} \underline{\mathbf{D}}^{(l+1)} \odot \sigma'(\underline{\mathbf{F}}^{(l)}), & \text{if } l = L-1, \ldots, 1 \end{cases} \tag{7.61}$$

and

$$\frac{\partial \underline{\mathbf{F}}^{(l)}}{\partial \mathbf{W}^{(l)}_{(N+1)}} = \underline{\mathbf{Z}}^{(l-1)T} \otimes \mathbf{I}_{I^{(l)}_{N+1}} \tag{7.62}$$

Substituting (7.46) and (7.62) into (7.60), allows the gradients for the factor matrices $\mathbf{A}^{(n)}_{(l)}$ and the core tensors $\mathcal{G}^{(l)}$ to be written as

$$\begin{aligned}\frac{\partial L}{\partial \text{vec}(\mathbf{A}^{(n)}_{(l)})} &= \left(\mathbf{P}_n \frac{\partial \mathbf{W}^{(l)}_{(n)}}{\partial \mathbf{A}^{(n)}_{(l)}}\right)^T \text{vec}\left(\underline{\mathbf{D}}^{(l)} \underline{\mathbf{Z}}^{(l-1)T}\right) \\ \frac{\partial L}{\partial \mathbf{G}^{(l)}_{(N+1)}} &= \mathbf{A}^{(N+1)T}_{(l)} \underline{\mathbf{D}}^{(l)} \underline{\mathbf{Z}}^{(l-1)T} \left(\bigotimes_{i=N}^{1} \mathbf{A}^{(i)}_{(l)}\right)\end{aligned} \tag{7.63}$$

Moreover, if $n = N + 1$, by making use of (7.42), the following result is obtained

$$\frac{\partial L}{\partial \mathbf{A}^{(N+1)}_{(l)}} = \underline{\mathbf{D}}^{(l)} \underline{\mathbf{Z}}^{(l-1)T} \left(\bigotimes_{i=N}^{1} \mathbf{A}^{(i)}_{(l)}\right) \mathbf{G}^{(l)T}_{(N+1)} \tag{7.64}$$

Similarly, $\frac{\partial L}{\partial \mathbf{b}^{(l)}} \in \mathbb{R}^{I_{N+1}^{(l)}}$ can be computed first by considering $\frac{\partial L}{\partial \mathbf{B}^{(l)}} \in \mathbb{R}^{I_{N+1}^{(l)} \times M}$,

$$\begin{aligned} \frac{\partial L}{\partial \text{vec}(\mathbf{B}^{(l)})} &= \frac{\partial \underline{\mathbf{F}}^{(l)}}{\partial \mathbf{B}^{(l)}} \text{vec}(\underline{\mathbf{D}}^{(l)}) \\ &= (\mathbf{I}_M \otimes \mathbf{I}_{I_{N+1}^{(l)}}) \text{vec}(\underline{\mathbf{D}}^{(l)}) \end{aligned} \tag{7.65}$$

Hence, from the rules of the Kronecker product, $\frac{\partial L}{\partial \mathbf{B}^{(l)}} = \underline{\mathbf{D}}^{(l)}$, which implies that,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{b}^{(l)}} &= \left(\frac{\partial \mathbf{B}^{(l)}}{\partial \mathbf{b}^{(l)}} \right)^T \text{vec}(\underline{\mathbf{D}}^{(l)}) \\ &= \left(1_M^T \otimes \mathbf{I}_{I_{N+1}^{(l)}} \right) \text{vec}(\underline{\mathbf{D}}^{(l)}) \end{aligned} \tag{7.66}$$

The forward and back-propagation procedures are summarized in Algorithm 10.

Algorithm 10. Forward and back-propagation for TTLs

Input: Dataset $\underline{\mathbf{X}} \in \mathbb{R}^{I_N \times M}$

$\underline{\mathbf{Z}}^{(0)} = \underline{\mathbf{X}}$, and initialize all $\mathbf{A}_{(l)}^{(n)}$ and $\mathbf{G}_{(N+1)}^{(l)}$

Forward propagation:
for $l = 1, \ldots, L$ **do**
  Compute $\mathbf{W}_{(N+1)}^{(l)}$ via (7.30)
  $\underline{\mathbf{F}}^{(l)} = \mathbf{W}_{(N+1)}^{(l)} \underline{\mathbf{Z}}^{(l-1)} + \mathbf{B}^{(l)}$
  $\underline{\mathbf{Z}}^{(l)} = \sigma(\underline{\mathbf{F}}^{(l)})$
  Store $\underline{\mathbf{F}}^{(l)}$, $\underline{\mathbf{Z}}^{(l)}$
  Compute and store $\sigma'(\underline{\mathbf{F}}^{(l)})$
end for

Back-propagation:
for $l = L, L-1 \ldots, 1$ **do**
  Compute $\underline{\mathbf{D}}^{(l)}$ via (7.61)
  Compute the gradients via (7.63)-(7.66)
  $\mathbf{A}_{(l)}^{(n)} \leftarrow \mathbf{A}_{(l)}^{(n)} - \eta \left(\frac{\partial L}{\partial \mathbf{A}_{(l)}^{(n)}} \right)$
  $\mathbf{G}_{(N+1)}^{(l)} \leftarrow \mathbf{G}_{(N+1)}^{(l)} - \eta \left(\frac{\partial L}{\partial \mathbf{G}_{(N+1)}^{(l)}} \right)$
  $\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \left(\frac{\partial L}{\partial \mathbf{b}^{(l)}} \right)$
end for

7.6 Simulation Results

A comprehensive experimental validation of the proposed concept includes the discussion of physical interpretability, followed by intuitive simulation results on synthetic data as well as standard benchmark datasets.

7.6.1 Physical Interpretability

Based on the general equation for the TTL model in (7.31), it is natural to ask whether the matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ are physically related to the data tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_1 \times \cdots \times I_N}$. To this end, consider the identity,

$$\begin{aligned} F(\boldsymbol{\mathcal{X}}) &= \left[\mathbf{A}^{(N+1)}\mathbf{G}_{(N+1)} \bigotimes_{i=N}^{1} \mathbf{A}^{(i)T}\right]\text{vec}(\boldsymbol{\mathcal{X}}) + \mathbf{b} \\ &= \mathbf{A}^{(N+1)}\mathbf{G}_{(N+1)}\text{vec}\big(\boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times \cdots \times_N \mathbf{A}^{(N)T}\big) + \mathbf{b} \end{aligned} \tag{7.67}$$

This implies that the n physical modes, of dimensionality I_n, of tensors $\boldsymbol{\mathcal{X}}$ are directly intertwined with the corresponding factor matrices $\mathbf{A}^{(n)}$. As a direct consequence, certain gradient terms $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ will carry more significant information than others, depending on which modes I_n of the original data tensors are richer in structure. This is of great practical importance, as one of the most well-known problems with NNs is their black-box nature, while the proposed method offers viable means to aid the understanding of which data features have a higher influence on their training process. Figure 7.2 offers a graphical representation of (7.67).

Remark 7.4. From Figure 7.2, it becomes apparent that each factor matrix, $\mathbf{A}^{(n)}$, and hence each gradient, $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$, is associated with the respective mode-n, suggesting that the gradients themselves will carry valuable information related to the underlying data structure.

Remark 7.5. A more rigorous mathematical argument for the above can be provided via inspection of (7.44) and (7.42), which shows that $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ is dependent on $\mathbf{G}_{(n)}$, and hence $\mathbf{W}_{(n)}$. Also, the unfolding of the weight tensor along mode-n directly reveals the connection with mode-n of the input data $\boldsymbol{\mathcal{X}}$, and it is therefore natural that each $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ has modal information embedded within. This conclusion could not have been reached without a solid underlying mathematical analysis.

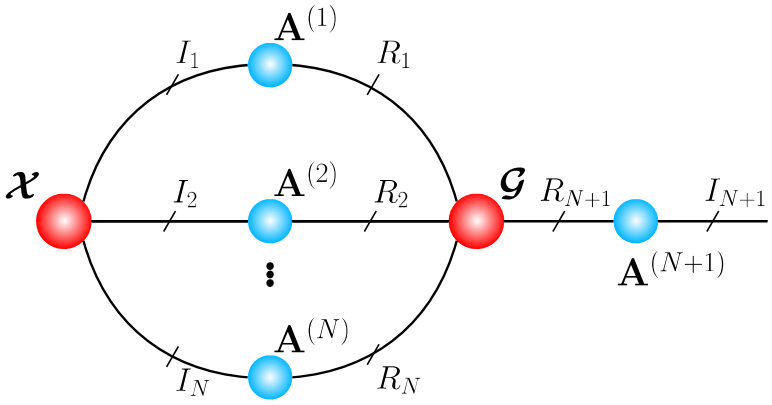


Figure 7.2: Graphical representation of (7.67).

7.6.2 Synthetic Data

To provide more concrete intuition into the above discussion, the proposed model was evaluated on two synthetic datasets, referred to as Synthetic Dataset 1 (SD1), and Synthetic Dataset 2 (SD2). Each dataset was composed of 28×28 grey-scale images (2-nd order tensors), which were white everywhere except for randomly selected rows for SD1 and randomly selected columns for SD2, designated black. Examples of images from these datasets are shown in Figure 7.3.

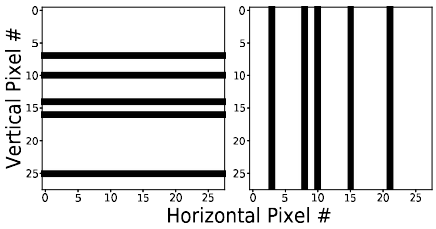


Figure 7.3: Sample images from two synthetic datasets. *Left*: Synthetic Dataset 1. *Right*: Synthetic Dataset 2.

Such a structure was chosen for SD1 and SD2 to ensure that one tensor mode is far richer in structure than the other. It is therefore expected that $\frac{\partial L}{\partial \mathbf{A}^{(1)}}$ will be more important than $\frac{\partial L}{\partial \mathbf{A}^{(2)}}$ for SD1, and vice-versa for SD2. For quantitative assessment, at each epoch we stored the normalized Frobenius norm of each gradient, defined as

$$\frac{\partial L^*}{\partial \mathbf{A}^{(n)}} = \left|\left| \frac{\partial L}{\partial \mathbf{A}^{(n)}} \right|\right|_F \Big/ (I_n R_n) \tag{7.68}$$

Simulations were run for 50 epochs and the results are shown in the left-most column of Figure 7.4, which suggest that the proposed model is capable of capturing well the relative importance of the modal structure within data. In the case of SD1, mode-1 is richer in structure, and this is indicated by the fact that $\frac{\partial L^*}{\partial \mathbf{A}^{(1)}} > \frac{\partial L^*}{\partial \mathbf{A}^{(2)}}$ for all epochs. The converse is true for SD2. This demonstrates that the TTL implicitly resolves the notorious black-box nature of NNs, by offering physically meaningful information inferred from the gradients of the trained weights.

7.6.3 MNIST Dataset

The desirable compression properties of the TTL are now demonstrated by applying it to the MNIST dataset [152] for the task of handwritten digit recognition. The MNIST dataset is composed of 60000 28×28 greyscale images for training and 10000 for testing. Samples from the dataset are shown in Figure 7.5. A neural network with 1 hidden layer comprising 300 nodes was employed. A Rectified Linear Unit (ReLU) activation function was used, and the hidden layer was next substituted by the proposed TTL. The weight tensor, $\mathcal{W} \in \mathbb{R}^{28 \times 28 \times 300}$, is in uncompressed format if its core $\mathcal{G} \in \mathbb{R}^{28 \times 28 \times 300}$. The compression factor (CF) was computed as the ratio of the number of elements of a "full" core tensor and factor matrices to the number of elements in a compressed core and its respective matrices. Table 7.2 shows results for different CFs, over a training period of 500 epochs.

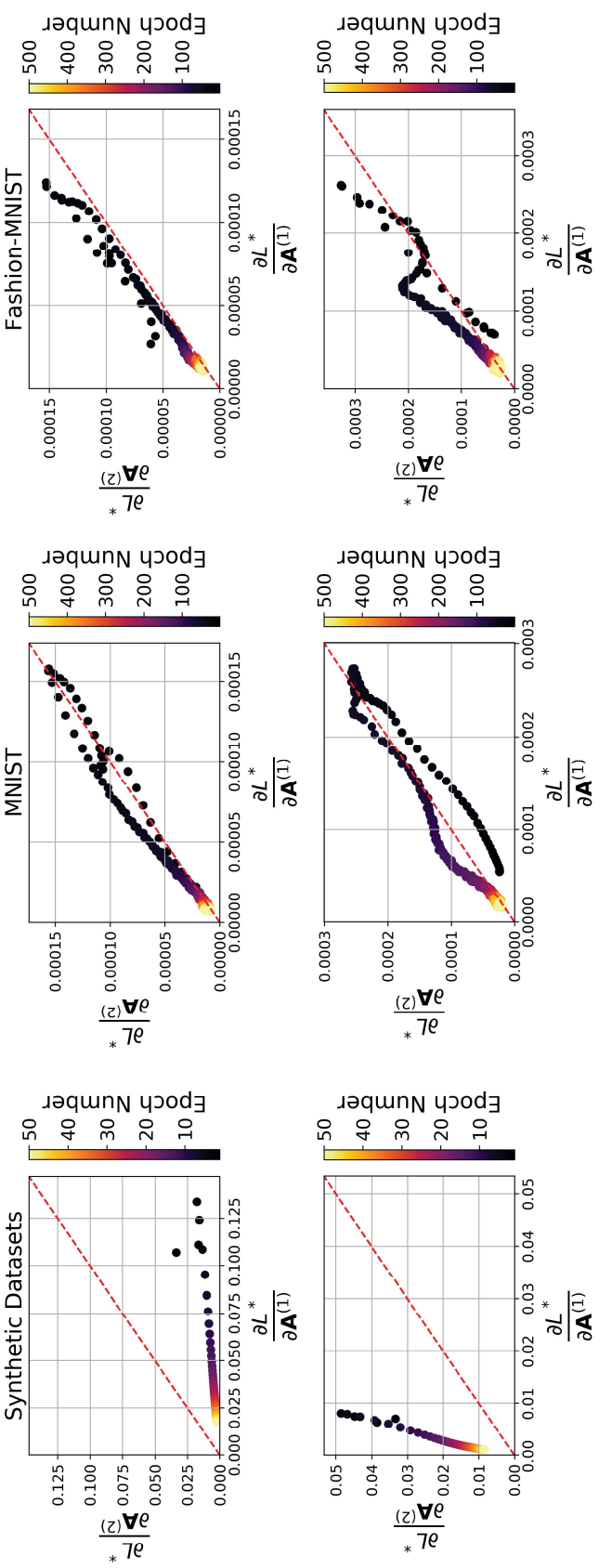


Figure 7.4: Gradients across epochs – TTL applied to various datasets. *Left column. Top row:* SD1; *bottom row:* SD2. The disparity in modal structural information is reflected in the magnitude of the normalized gradients. Middle column. Top row: MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{10\times10\times30}$. Bottom row: MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{5\times5\times10}$. No difference in modal structural information is observed. Right column. Top row: Fashion-MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{10\times10\times30}$; tbottom row: Fashion-MNIST and TTL with $\mathcal{G} \in \mathbb{R}^{5\times5\times10}$. Results suggest that the second mode carries more structural information.



Figure 7.5: Sample images from the MNIST dataset.

The model achieved a compression factor (CF) of 18.73 with accuracy of 95.6%, and CF of 66.63 with accuracy of 93.3%, corresponding to a decrease of only slightly over 2% from the original uncompressed network. Figure 7.4 shows scatter plots of the normalized gradients, $\frac{\partial L^*}{\partial \mathbf{A}^{(1)}}$ and $\frac{\partial L^*}{\partial \mathbf{A}^{(2)}}$, for both the compression factors in Table 7.2. Observe that, for MNIST, both modes have comparable structural significance, as suggested by a simple visual perspective on the images (white numbers on black background).

7.6.4 Fashion-MNIST Dataset

The proposed TTL model was further evaluated on the Fashion-MNIST dataset [153], which is intended to serve as a direct, more complex replacement of the original MNIST. Fashion-MNIST consists of 28×28 greyscale images depicting fashion products belonging to a total of 10 different classes, examples of which are shown in Figure 7.6. The training and testing sets consist of 60000 and 10000 images, respectively. A neural network with 2 hidden layers of sizes 300 ($l = 1$) and 200 ($l = 2$), with ReLU activation functions was employed. The layer $l = 1$ was replaced with the proposed TTL, and the layer was compressed by the same degree as for MNIST. The uncompressed network attained an accuracy of 86.3%, while, at a loss of less than 4% accuracy, the TTL achieved a CF of 66.63, obtaining a 82.3% performance.



Figure 7.6: Sample images from the Fashion-MNIST dataset.

Differently from MNIST, Fashion-MNIST entails a disparity in modal structural information. For example, a coat and a dress occupy roughly the same vertical space on the images (refer to Figure 7.6), while the difference between the two products is revealed horizontally – mode-1 and mode-2. This disparity implies that mode-2 should play a relatively more important role than mode-1 in training. This is reflected in Figure 7.4, where it is shown that, on average, $\frac{\partial L^*}{\partial \mathbf{A}^{(1)}} > \frac{\partial L^*}{\partial \mathbf{A}^{(2)}}$. This further supports the arguments in Section 7.6.1, that, within the proposed TTL, each gradient $\frac{\partial L}{\partial \mathbf{A}^{(n)}}$ is directly intertwined with its corresponding mode-n.

Table 7.2: TTL performance for various Compression Factors (CF).

Dataset	CF	Core Size	Accuracy (%)
MNIST	1	$28 \times 28 \times 300$	95.9%
	18.73	$10 \times 10 \times 30$	95.6%
	66.63	$5 \times 5 \times 10$	93.3%
Fashion MNIST	1	$28 \times 28 \times 300$	86.3%
	18.73	$10 \times 10 \times 30$	85.4%
	66.63	$5 \times 5 \times 10$	82.3%

Some gradients will be more informative about the training process, depending on which data mode carries more structure. This is made explicit owing to the analytical derivation of back-propagation, in terms of the TKD, which allowed to gain insight from the expressive power of tensors.

7.6.5 CIFAR-10 Dataset



Figure 7.7: Sample images from the CIFAR-10 dataset.

The same network employed in Sections 7.6.3-7.6.4 was applied on the CIFAR-10 dataset [154], with the TTL substituting the first hidden layer comprised of 300 nodes. Since CIFAR-10 is composed of $32 \times 32 \times 3$ RGB images (50000 for training), the decomposed uncompressed weight tensor corresponds to a core of size $\mathcal{G} \in \mathbb{R}^{32\times32\times3\times300}$. Samples from the dataset are shown in Figure 7.7. The layer was compressed by a factor of 152.45 by employing a core of size $\mathcal{G} \in \mathbb{R}^{10\times10\times3\times10}$. Simulations were ran for both the uncompressed and compressed networks, achieving accuracy rates of 53.8% and 48.03% respectively. Such a drop in performance compared to the MNIST and the Fashion-MNIST datasets was expected as the CIFAR-10 dataset is naturally more complex. However, by virtue of the proposed framework, it was possible to analyze the relative importance of the modal structure within the NN. Indeed, Figure 7.8 shows that mode-3, corresponding to RGB information, is more significant than the other two for classification purposes; in turn, it can be used as an additional input feature to more complex networks.

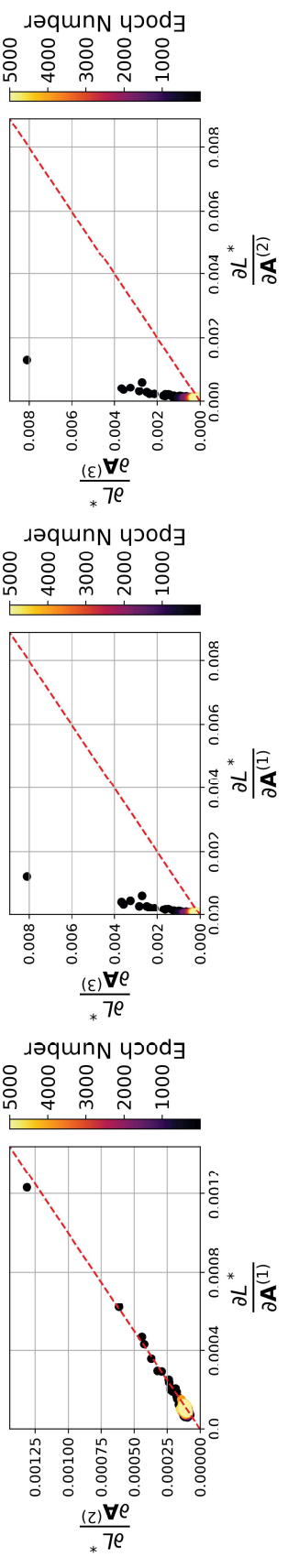


Figure 7.8: Gradients across epochs – TTL applied to CIFAR-10. A TTL with a core tensor $\mathcal{G} \in \mathbb{R}^{10\times10\times3\times10}$ is employed, to unveil more structural information within mode-3 than within the other modes, thus indicating that the colour scheme of the images within CIFAR-10 plays a major role in classification tasks.

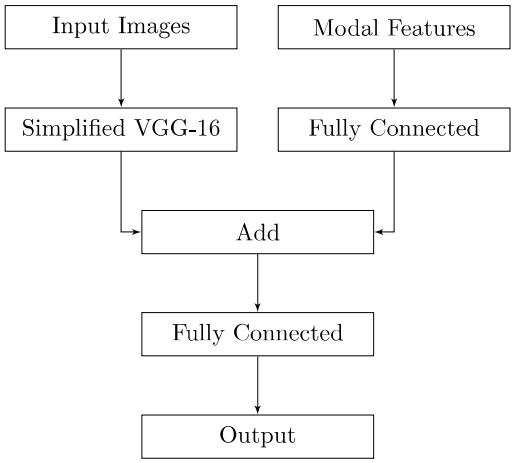


Figure 7.9: Modal-VGG architecture. Modal features are passed to the network in vector format as expressed in (7.69), while the simplified VGG-16 is the normal VGG-16 without the last two convolutional layers.

This was achieved by combining the well-known VGG-16 convolutional network [149] with a simple network consisting of a single fully connected layer, of 512 units, which receives as input modal information for each sample image, as illustrated in Figure 7.9. This model is referred to as Modal-VGG. Specifically, for each image $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{32\times32\times3}$ the vectorized 3-rd mode covariance was fed, that is

$$\mathbf{x} = \text{vec}(\mathbf{X}_{(3)}^T \mathbf{X}_{(3)}) \in \mathbb{R}^9 \tag{7.69}$$

To highlight the significance of the 3-rd mode, the VGG-16 was first simplified by removing its last two convolutional layers, then trained on CIFAR-10 in two ways:

1. Alone;

2. In combination with another network which analyzes modal information.

The outputs of the first fully connected layer of the simplified VGG-16 were added to those of the second network, before being fed through a final fully connected layer.

Remark 7.6. The Modal-VGG is an example of a straightforward methodology which is very useful in practice. Namely, if simple networks fail, instead of discarding completely their results, the TTL offers opportunities to retain some information of their training, which can then be fed to more complex networks. This, in turn, allows large NNs to be simplified and reduce their computational costs thanks to the so performed data augmentation.

Simulations were ran on an NVIDIA GeForce RTX 2080 Ti GPU. Table 7.3 shows that, for case (i) above, training speed was increased by over 12%, but the model accuracy decreased from 91.3% (for the full VGG-16) to 86.2%. The benefits of the proposed framework are evident in case (ii), where augmenting the data with the relevant modal features allowed for the accuracy to increase to 89.2%, while still maintaining about a 10% reduction in overall training time when compared to the full VGG-16 model. For rigour, the same experiment was carried out on the 1-st and 2-nd mode vectorized covariances, for which the Modal-VGG in Figure 7.9

Table 7.3: Modal-VGG performance.

Network	Accuracy (%)	Training Time Speed-up (%)
Original VGG-16;	91.3	–
Modal-VGG; no modal information	86.2	~ 12
Modal-VGG; mode-1	86.4	~ 10
Modal-VGG; mode-2	86.9	~ 10
Modal-VGG; mode-3	89.2	~ 10

attained accuracies of 86.4% and 86.9%, respectively. This further highlights the importance of mode-3 for the CIFAR-10 dataset and demonstrates that the TTL can indeed capture modal information, which, in turn, may be exploited to enhance physical meaning in training and reduce training time while at the same time maintaining comparable performance.

7.7 Chapter Summary

The Tucker Tensor Layer (TTL) has been introduced as an alternative to the dense weight matrices of fully connected layers in Deep Neural Networks (DNNs). This has been achieved by treating the weight-matrix as an unfolding of a higher-order tensor, followed by its decomposition and compression via the Tucker decomposition (TKD). By extending the notion of matrix derivatives to tensors, the tensor-valued back-propagation was derived within the proposed framework and the physical interpretability of the proposed model has been demonstrated. Experiments on synthetic data, Fashion-MNIST and CIFAR-10, show that, owing to the expressive power of tensors and in particular the TKD, the proposed framework has been able to provide insights into the network training process, by providing explicit information on the relative importance of each data mode in training. The exploitation of the relative importance of data modes has made it possible to train a data-augmented network comprising a simplified version of the VGG-16, which resulted in a 10% decrease in training time, at a small cost in terms of final accuracy. Moreover, the compression benefits of the proposed model have been demonstrated on both the MNIST and Fashion-MNIST datasets, with the TTL achieving a 66.63 fold compression while maintaining comparable performance to the uncompressed NN. This further illustrates the ability of the proposed TTL to help resolve the two main bottlenecks of NNs, namely their black-box nature and more efficient training of its millions of parameters.

Chapter 8

Tensor-Train Recurrent Neural Networks for Financial Forecasting

Life is a train, get on board.

Khaled Hosseini

Chapter Overview

This chapter is to serve as an extension to the results from the previous chapter. A more practical-oriented approach is adopted, and, similarly to **Chapter 6**, results are focused on financial applications. In particular, attention is given to Recurrent Neural Networks (RNNs), which are well known to be the de facto standard machine learning tool for sequence modelling, owing to their expressive power and long memory. Despite their advantages, however, the corresponding exponential increase in the number of parameters represents a computational bottleneck when dealing with large dimensional data. This has highlighted the necessity to equip RNNs with the ability to deal with the curse of dimensionality, which has led to the development of the Tensor-Train RNN (TT-RNN) model [57]. Despite promising results, the full potential of the TT-RNN is yet to be explored, especially in large-scale multi-modal scenarios, as in financial modelling. To address this issue, the potential of the TT-RNN is examined in the financial forecasting of currencies, based on financial indices from four different asset classes. Simulation results highlight the superiority of the TT-RNN over standard RNNs, both in terms of enhanced regularization and model accuracy, leading to higher generated profits. Similarly to **Chapter 7**, it is shown through the analysis of the gradients of the TT-factors how the TT-RNN model aids the interpretability of results, thus mitigating the notorious "black-box" issue associated with neural networks.

8.1 Introduction

Recurrent Neural Networks (RNNs) [147] and their variants, such as LSTMs and GRUs [156, 157, 158], have proven to be among the most successful machine learning approaches for sequence modelling, and have found application in Natural Language Processing (NLP), video data, and general time-series analysis [146, 159, 160]. Despite considerable success, RNNs are notoriously difficult to train when considering large-dimensional inputs [161]. One way to tackle these issues is through the inherent expressive power and super-compression properties of tensors.

Although substantial research has been carried out on the topic [76, 79, 86, 151], tensors and multilinear algebra have firstly been considered in conjunction with neural networks only

in [57]. The authors employed the TT decomposition to compress a fully connected neural network via *TT-layers* while maintaining comparable performance. Extensions of this work in [162] and [163] considered sequence modelling, through a compression of RNN architecture via TT decomposition (TT-RNN).

These advances have further highlighted the enormous potential stemming from joint consideration of multilinear algebra and neural networks such uncharted territory is the application to financial modelling, an area of great global importance which despite a huge volume and variety of data, still relies on "flat-view" matrix methods. One such uncharted territory is the application to financial modelling, an area of great global importance which despite a huge volume and variety of data, still mostly relies on "flat-view" matrix methods, although the Support Tensor Machine in **Chapter 6** and [86] has been already shown to bring practical advantages. To address this issue, the work [89] presented in this chapter sets out to extend the findings of [57] and [162], and employs TT-RNNs to predict the direction of movement of the Japanese Yen based on a collection of financial assets and indicators.

Financial forecasting is a linchpin of modern investment strategies, however, the performance of corresponding machine learning models tend to vary massively, depending on the type of analysis, assumptions, underlying assets, and time frames [164]. While there are no standard templates for comparing such models, the general consensus is that the models with strong regularization characteristics, such as Support Vector Machines, are preferred in this high uncertainty environment. This further underpins the use of TT as a regularization tool for RNNs, a subject of this study.

The results presented in this chapter demonstrate the benefits of the TT-RNN over a standard RNN trained on the same features, with accuracy increasing by as much as 30%. Owing to their super-compression capabilities, TTs were able to compress the model by a factor of 244, while vastly reducing overfitting. Finally, analogously to the findings of **Chapter 7** the inherent expressiveness of tensors has enabled the TT-RNN to offer new perspectives on the interpretability of results, by indicating which TT-cores, and consequently which tensor modes [86], play a more prominent role in forecasting, thus mitigating the "black-box" nature of neural networks.

8.2 Tensor-Train Neural Networks

This section outlines hot the TT decomposition can be integrated into neural networks, starting from fully connected networks, then extending the concept to recurrent networks.

8.2.1 Application to Fully Connected Networks

Recall from Section 2.2.3 that an N-th order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, can be expressed in the TT-format as

$$\begin{aligned} \boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_N) &= \mathbf{G}^{(1)}(i_1)\mathbf{G}^{(2)}(i_2)\cdots\mathbf{G}^{(N)}(i_N) \\ &= \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\cdots\sum_{r_{(N-1)}=1}^{R_{(N-1)}} g^{(1)}_{1 i_1 r_1} g^{(2)}_{r_1 i_2 r_2}\cdots g^{(N)}_{r_{(N-1)} i_N 1} \end{aligned} \tag{8.1}$$

where $\mathbf{G}^{(n)}(i_k), i_k = 1, \ldots, I_k$ are the i_k-th vertical slices of the core tensors

$$\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}, \quad n = 1, \ldots, N \tag{8.2}$$

with $R_0 = R_N = 1$. Also, a neural network fully connected layer can generally be expressed as

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{8.3}$$

with $\mathbf{y} \in \mathbb{R}^M$ being the output, $\mathbf{x} \in \mathbb{R}^P$ the input, $\mathbf{W} \in \mathbb{R}^{M \times P}$ the weight matrix, and $\mathbf{b} \in \mathbb{R}^M$ the bias vector. Under the assumption that $P = \prod_{n=1}^{N} I_n$ and that $M = \prod_{n=1}^{N} J_n$, vectors $\mathbf{y}, \mathbf{x}, \mathbf{b}$ can be folded into N-th order tensors $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, while matrix $\boldsymbol{\mathcal{W}}$ can be folded into a $2N$-th order tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_N \times J_N}$. Moreover, a bijection $\boldsymbol{\mu}$ can be established between the coordinate $m \in \{1, \ldots, M\}$ of vectors $\mathbf{x}$ and $\mathbf{b}$ and the N-dimensional vector index of the corresponding tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{B}}$, that is

$$\boldsymbol{\mu} : m \mapsto (\mu_1(m), \ldots, \mu_N(m)) \tag{8.4}$$

where $\mu_n(m) \in \{1, \ldots, J_n\}$. An example for the 3-rd order case is shown in Figure 8.1.

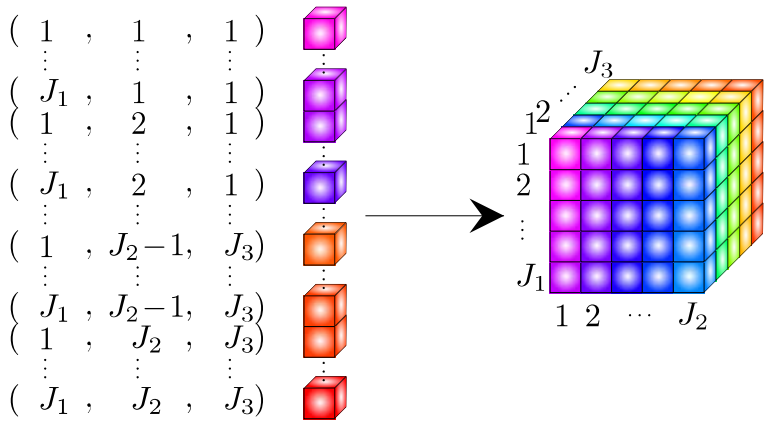


Figure 8.1: Example of bijection of the indices of a general vector $\mathbf{x} \in \mathbb{R}^M$ to a 3-dimensional vector index (J_1, J_2, J_3) of a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$.

Similarly, bijections

$$\begin{aligned} \boldsymbol{\mu} &: m \mapsto (\mu_1(m), \ldots, \mu_N(m)) \\ \boldsymbol{\nu} &: p \mapsto (\nu_1(p), \ldots, \nu_N(p)) \end{aligned} \tag{8.5}$$

map row and column indices m and p of matrix $\mathbf{W} \in \mathbb{R}^{M \times P}$ to the $2N$-th order tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_N \times J_N}$. With a slight abuse of notation, the latter can be folded into an N-th order tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N}$, which, in turn, can be converted into the TT-format as,

$$\begin{aligned} \mathbf{W}(m, p) &= \boldsymbol{\mathcal{W}}\big((\mu_1(m), \nu_1(p)), \ldots, (\mu_N(m), \nu_N(p))\big) \\ &= \mathbf{G}^{(1)}\big((\mu_1(m), \nu_1(p)\big) \cdots \mathbf{G}^{(N)}\big((\mu_N(m), \nu_N(p)\big) \end{aligned} \tag{8.6}$$

Therefore, the linear transformation (8.2) of a fully connected layer can be expressed in the

TT-format as,

$$\mathcal{Y}(j_1, j_2, \ldots, j_N) = \sum_{j_1,\ldots,j_N} \mathbf{G}^{(1)}(j_1, i_1) \cdots \mathbf{G}^{(N)}(j_N, i_N)\mathcal{X}(i_1, i_2, \ldots i_N) + \mathcal{B}(i_1, i_2, \ldots i_N) \quad (8.7)$$

For further details, refer to [57, 89].

8.2.2 Application to Recurrent Networks

Recurrent neural networks (RNNs) capture time-varying dependencies via feedback weights that contain information related to previous time steps [147], that is

$$\mathbf{h}_t = \sigma(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{xh}\mathbf{x}_t + \mathbf{b}^h) \quad (8.8)$$

where $\sigma(\cdot)$ is an element-wise activation function, and the hidden state at time t, $\mathbf{h}_t \in \mathbb{R}^M$, is generated as a function of

1. The previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^M$;
2. The input $\mathbf{x}_t \in \mathbb{R}^P$;
3. The recurrent matrix $\mathbf{W}^{hh} \in \mathbb{R}^{M\times M}$;
4. The input weight matrix $\mathbf{W}^{xh} \in \mathbb{R}^{M\times P}$;
5. A bias vector $\mathbf{b}^h \in \mathbb{R}^M$.

It has been shown that the hidden state $\mathbf{h}_t \in \mathbb{R}^M$ can be folded into an N-th order tensor [162], $\mathcal{H}_t \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$, while the input $\mathbf{x}_t \in \mathbb{R}^P$ into an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. This allows the input matrix $\mathbf{W}^{xh} \in \mathbb{R}^{M\times P}$ to be expressed into the TT-format according to (8.6).

Remark 8.1. The storage complexity for $\mathbf{W}^{xh}$ within the TT-RNN decreases from an exponential $\prod_{n=1}^{N} I_n J_n = PM$ in the original format to a linear $\sum_{n=1}^{N} I_n J_n R_{n-1} R_n$, which is highly efficient for small TT-ranks and mode sizes.

8.3 Application to Finance and Discussions

In this section the TT-RNN is applied to the concrete problem of financial forecasting. First, the input data structure is outlined, then experimental results are presented and analysed.

8.3.1 Input Data Structure

The task is to predict the movements of JPYUSD. To this end, 24 information sources were considered, spanning four different asset classes: equities, currencies, commodities, and fixed income. Each class comprises six assets or indicators (financial components). In other words, to make full use of expressive power of tensors, several financial time-series with multi-way structure were selected from Bloomberg, as shown in Table 8.1. The input feature vector at time instant t for i-th component, $\mathbf{x}_t^i \in \mathbb{R}^{20}$, consists of 20 features, outlined below

1. Daily log-differences,
$$r_t = \log(S_t) - \log(S_{t-1}) \tag{8.9}$$
where S_t is the daily closing value (**1 feature**);

2. Rolling statistics of log-differences, that is the average, standard deviation, skewness, and kurtosis over $5, 10$, and 22 time-steps, corresponding to 1 week, 2 weeks, and 1 month of trading data (**12 features**);

3. Relative price min-max, P_{mm},
$$\mathrm{P}_{mm} = \frac{p_t - \min_{t\in T}(p_t)}{\max_{t\in T}(p_t) - \min_{t\in T}(p_t)} \tag{8.10}$$
where $T = \{t - L + 1, \ldots, t - 1, t\}$ and L is a window of 5, 10, and 22 time-steps (**3 features**);

4. Relative high-low-close, HLC_t,
$$\mathrm{HLC}_t = \frac{p_t - l_t}{h_t - l_t} \tag{8.11}$$
where l_t and h_t are the *low* and *high* price for day t (**1 feature**);

5. High-low spread, HLS_t,
$$\mathrm{HLS}_t = \frac{h_t - l_t}{l_t} \tag{8.12}$$
corresponding to (**1 feature**);

6. Amount traded, i.e. the volume v_t, at time t (**1 feature**);

7. Units of unsettled future contracts at time t (**1 feature**).

For every time-step, t, all 24 feature vectors $\mathbf{x}_t^i \in \mathbb{R}^{20}$, $i = 1, \ldots, 24$, were arranged into a 3-rd order tensor, $\boldsymbol{\mathcal{Z}}_t \in \mathbb{R}^{20\times 6\times 4}$ (features $\times$ components $\times$ asset classes), which was further reshaped into a 5-th order tensor, $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{2\times 2\times 5\times 6\times 4}$, such that the first three modes correspond to the 20 features while the last two correspond to the components and asset classes. Note that a value of 0 is given to any missing feature from Bloomberg.

Remark 8.2. The choice of a 5-th order tensor $\boldsymbol{\mathcal{X}}_t$ instead of the original 3-rd order tensor $\boldsymbol{\mathcal{Z}}_t$, improves compression due to smaller mode sizes. This comes without loss in physical interpretability, as the modes corresponding to features, components, and asset classes are not mixed.

8.4 Experimental Results

The obtained results are portrayed in this section [89]. The considered financial data is shown in Table 8.1 in the period from May 2006 to May 2019. The first 90% of data, up to February 2018, was used for training, and the remaining 10% for testing. Each feature was centred about its mean and normalized by its standard deviation. A neural network consisting of one recurrent and one fully connected layer was employed, the architecture of which is specified in Table 8.2

Table 8.1: Multi-way financial data selected from Bloomberg.

Asset Class	Symbol	Description
Equities (Index)	SPX	US stock index
Equities (Index)	MXCA	Canadian stock index
Equities (Index)	UKX	UK stock index
Equities (Index)	FTSEMIB	Italian stock index
Equities (Index)	SHSZ300	Chinese stock index
Equities (Index)	NKY	Japanese stock index
Currencies (FX Pair)	CHFUSD	Swiss Frank
Currencies (FX Pair)	CADUSD	Canadian Dollar
Currencies (FX Pair)	GBPUSD	British Pound
Currencies (FX Pair)	EURUSD	Euro
Currencies (FX Pair)	CNYUSD	Chinese Yuan
Currencies (FX Pair)	JPYUSD	Japanese Yen
Commodities (Futures)	GC1	Gold
Commodities (Futures)	HG1	Copper
Commodities (Futures)	CL1	Crude Oil
Commodities (Futures)	NG1	Natural Gas
Commodities (Futures)	S1	Soybeans
Commodities (Futures)	C1	Corn
Fixed Income (Index)	USGG10YR	US 10 Yrs Yield
Fixed Income (Index)	GCAN10YR	Canadian 10 Yrs Yield
Fixed Income (Index)	GUKG10	UK 10 Yrs Yield
Fixed Income (Index)	GBTPGR10	Italian 10 Yrs Yield
Fixed Income (Index)	GCNY10YR	Chinese 10 Yrs Yield
Fixed Income (Index)	GJGB10	Japanese 10 Yrs Yield

for the uncompressed case. The output labels $\{+1, 0, -1\}$ were assigned to the Japanese Yen time-series, according to whether the price at time step $(t+1)$ increased $(+1)$ or decreased (-1) with respect to time step t; the value 0 was assigned if $|r_t| \leq 10^{-4}$. Each input sample consisted of 10 time-steps.

The TT-RNN model was employed to compress the weight matrix $\mathbf{W}^{xh} \in \mathbb{R}^{M \times P}$ given in Table 8.2 by reshaping and expressing it in the TT-format, in accordance to the dimensions of the input tensors $\boldsymbol{\mathcal{X}}_t$, such that $P = 2 \times 2 \times 5 \times 6 \times 4$ and $M = 4 \times 4 \times 4 \times 4 \times 4$. An illustration of the interaction between input $\boldsymbol{\mathcal{X}}_t$, and the tensorized weight matrix, $\mathbf{W}^{xh}$, is shown in Figure 8.2.

Remark 8.3. Figure 8.2 indicates that keeping track of the variation of each core throughout the training can reveal hidden information about the relative importance of the each mode, as shown in **Chapter 7** [86, 89].

Table 8.2: Recurrent neural network architecture

RNN element	Recurrent Layer	Fully-Connected Layer
Weight Matrix, $\mathbf{W}^{xh}$	$(480, 1024)$	$(1024, 3)$
Feedback Matrix, $\mathbf{W}^{hh}$	$(1024, 1024)$	N.A.
Bias Vector, $\mathbf{b}^{h}$	$(1024, 1)$	(3,1)
No. of Parameters	1541120	3075
Activation	tanh	softmax

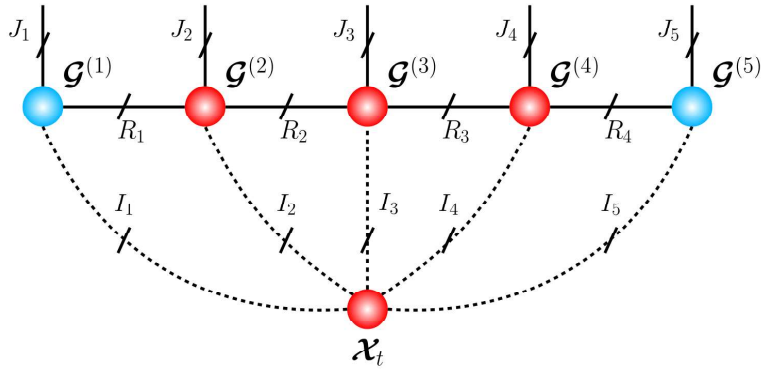


Figure 8.2: Modal connections between every mode of the input data tensor, $\mathcal{X}_t$, and the TT-cores, $\mathcal{G}_n$, of the tensorized weight matrix $\mathbf{W}^{xh}$ in TT-format.

The established TT-RNN model was trained using stochastic gradient descent with a learning rate of 10^{-5} over 20 epochs, with a batch size of 66 and using the categorical cross-entropy loss function. The TT-ranks (R_1, R_2, R_3, R_4) for the TT-RNN model were empirically found to be optimal for rank $R_i = 6$, $i = 1, \ldots, 4$. It was also found that higher values of R_i would result in overfitting, and lower values in under-fitting. In turn, this suggests that the TT-ranks may be used for regularization in addition to compression, which is extremely useful in practice as RNNs are generally prone to overfitting [165].

Figure 8.3 shows the model loss and accuracy as a function of epochs, with the blue and green plots showing respective results for training and validation samples. The loss monotonically decreased both for the training and validation data, while the accuracy rate increased to 56.63% for the training data and increased on average for the validation data, attaining a final value of 52.98% for the test set. In the process, the weight matrix, $\mathbf{W}^{xh} \in \mathbb{R}^{1024\times480}$, was compressed by a factor of 244 with the number of free parameters decreasing from the original $\prod_{n=1}^{N} I_n J_n = 491520$, to an affordable $\sum_{n=1}^{N} I_n J_n R_{n-1} R_n = 2016$.

Remark 8.4. Recall from **Chapter 6** that performance of above 50% in accuracy is considered significant for financial applications.

The benefits of the TT-RNN model were further illustrated through a comparison with its uncompressed RNN counterpart, trained using the same configuration of hyper-parameters. The corresponding learning curves are shown in Figure 8.3. Observe the clear case of overfitting, as the training and validation losses diverged, resulting in an accuracy of 64.83% for

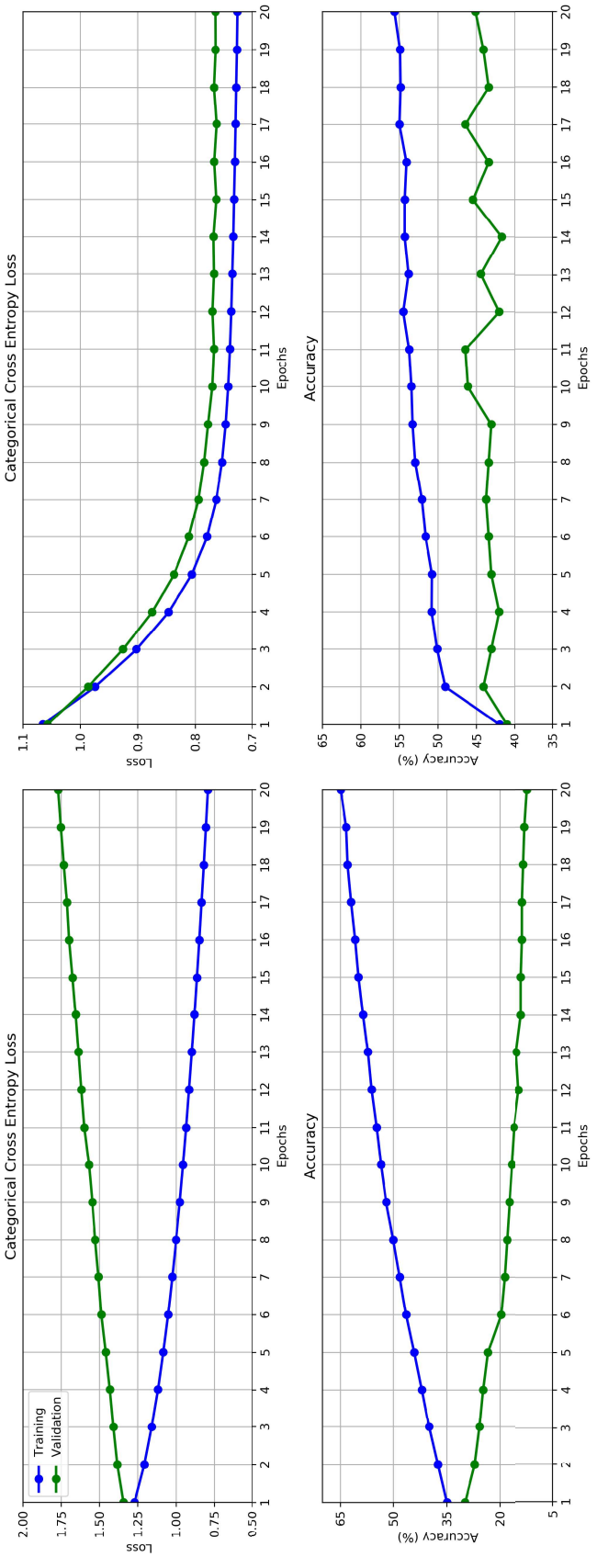


Figure 8.3: Training curves of the simple RNN model compared to the TT-RNN. It is clear that the TT-RNN presents less overfitting, and can hence be employed as a regularization tool.

training and 21.32% for testing (most were classified as 0).

Figure 8.4 shows simulated cumulative profits generated from model predictions, which further highlights the superiority of the TT-RNN model, that managed to outperform both the RNN model and the long-only (i.e. buy and hold) trading strategies. The profits were generated by being long or short (i.e. betting for or against the underlying asset) according to the predictions of the model. The trading decisions were updated on a daily basis using the features from previous days, while the sizes of trades were scaled according to the probability estimates of the model, which ranged from 0 (not holding a position at all) to 1 (fully invested).



Figure 8.4: Cumulative profits generated from the long-only strategy, shown in black, the TT-RNN predictions, shown in red and the simple RNN predictions, shown in purple.

Following a similar approach as from Section 7.6, the architecture of the TT-RNN can be exploited to gain insights into the network's training process, by inspecting how the gradients corresponding to different factors evolve over time. Figure 8.5 shows the normalized core change, that is

$$\frac{||\Delta \boldsymbol{\mathcal{G}}_e^{(n)}||_F^2}{I_n J_n R_{n-1} R_n} \tag{8.13}$$

where

$$\Delta \boldsymbol{\mathcal{G}}_e^{(n)} = \boldsymbol{\mathcal{G}}_e^{(n)} - \boldsymbol{\mathcal{G}}_{e-1}^{(n)} \tag{8.14}$$

and $\boldsymbol{\mathcal{G}}_e^{(n)}$ is the value of $\boldsymbol{\mathcal{G}}^{(n)}$ at epoch $e = 2, \ldots, 20$. Results show that the cores $\boldsymbol{\mathcal{G}}^{(1)}$ and $\boldsymbol{\mathcal{G}}^{(5)}$ exhibited greatest changes, suggesting that the features mentioned in Section 8.3.1, and the asset classes from Table 8.1, account for most of the predictive power of TT-RNN for our problem. Inspecting the changes in individual factors throughout training aids the interpretability of results. For instance, the greater change in $\boldsymbol{\mathcal{G}}^{(5)}$ over $\boldsymbol{\mathcal{G}}^{(4)}$ suggests larger importance of inter-class dependencies over intra-class ones. This insight is only possible with the neural network represented in its TT format, which brings the benefit of interpretability in addition to that of compression.

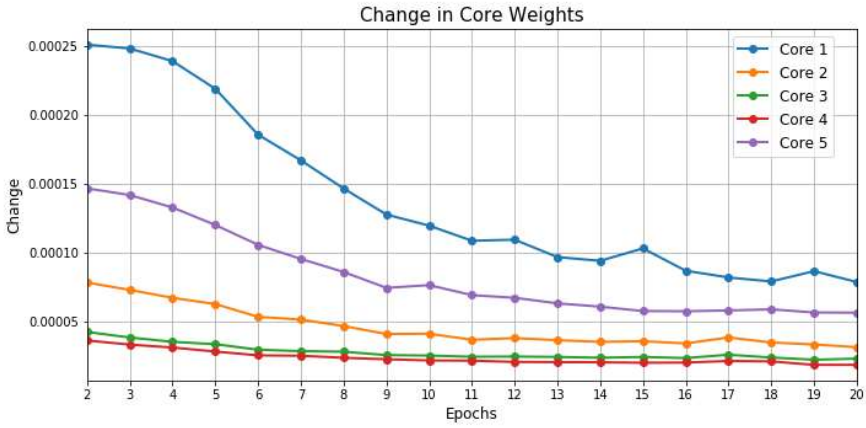


Figure 8.5: Normalized change of core weights of TT-RNN.

8.5 Chapter Summary

Traditional currency forecasting models are typically based on intra-class variables only, despite the wide availability of other strongly or weakly related asset classes. To address this issue, the ability of the Tensor-Train Recurrent Neural Network (TT-RNN) has been investigated in the context of financial forecasting of currencies. This has been achieved based on multi-way financial data from four asset classes: equities, currencies, commodities, and fixed income. The model has been trained on tensorized data from May 2006 to February 2018, and tested on the period from February 2018 to May 2019. Simulation results have verified the desirable compression and regularization properties of the TT-RNN, which outperformed its standard RNN counterpart by over 30% in accuracy, while reducing the number of parameters by a factor of 244. Moreover, the TT-RNN has been demonstrated to allow for interpretability, thus mitigating the well-known black-box issue inherent to neural networks. Indeed, the inspection of individual TT-cores during the training process, has suggested the greater importance of inter-class over intra-class dependencies for forecasting the movements of the Japanese Yen.

Conclusion to Part II

This chapter concludes the third and final part of this thesis, devoted to the application of tensors to neural networks. **Chapter 7** introduced the Tucker Tensor Layer (TTL) as an alternative to the dense fully connected layers of neural networks, while **Chapter 8** applied the Tensor Train (TT) decomposition to Recurrent Neural Networks (RNNs). All considered applications have demonstrated a significant compression of network parameters owing to the desirable properties of tensors, at a small cost in accuracy. Moreover, the notions from **Chapter 3** were once again employed to elucidate important concepts, finding practical relevance in both chapters through the development of methodologies to gain insights into the networks' training process, via gradient inspection of the core factors.

Chapter 9
Conclusions & Future Work

> Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience.
>
> *Roger Bacon*

9.1 Conclusion

This thesis addressed the problem of developing novel and interpretable tensor algorithms and showing their utility in practical applications. The key motivators for this were the existence of opportunities to apply tensors within the fields of image processing, time-series analysis, and deep learning, together with the necessity to improve interpretability and mitigate the "black-box" nature of existing models. Drawing upon tensor fundamentals, and, in particular, their graphical representation, it was possible to achieve this goal and introduce new results in the fields of tensor decompositions, signal processing oriented machine learning, and deep learning. 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 provided the general perspective on tensors for signal processing, by introducing the 4V's of Big Data and summarizing the field's historical background. The Curse of Dimensionality was defined, together with the Blessing of Dimensionality, which is based upon the superior expressiveness that tensor algebra offers over matrix algebra. The problem statement was formulated, and the thesis outline was provided.

- **Chapter 2: Tensor Notations and Decompositions.** Chapter 2 provided the required background for multi-linear algebra, by presenting important decompositions and fundamental concepts, such as tensorization, folding and unfolding, and contractions. The main tensor decompositions were introduced, that is, the Canonical Polyadic, Tucker, and Tensor Train decompositions (CPD, TKD, TT, respectively). Notions on their computation and uniqueness properties were provided. The building blocks of Tensor Networks (TNs) were also briefly outlined and explored in depth in the following chapter.

- **Chapter 3: A Tale of Nodes and Edges**. Chapter 3 is the first chapter of **Part I: Insights into Tensor Decompositions** of this thesis. It extensively introduced the graphical methodologies to represent tensors and tensor operations as TNs, via diagrams composed of nodes and edges. This chapter demonstrated that any decomposition can be interpreted as a TN, which promotes the use of graphical representations to gain intuitive insights which would otherwise be difficult to uncover. New perspectives on even

the most fundamental concepts such as folding and inner products were provided, which would then be of use in later sections of the thesis. To emphasize the benefits of a graphical approach, a concrete example was provided by introducing the novel Tensor-Train Contraction Product (TTCP), the computational benefits of which were demonstrated through numerical simulations. Each chapter of this thesis makes use of at least one occasion of the concepts presented in Chapter 3, which are thus to be considered as a central theme of this work.

- **Chapter 4: Easing the Computational Burden on the Canonical Polyadic Decomposition.** This was the second chapter of **Part I**, and put the CPD under closer mathematical scrutiny. A practical usage of the CPD can be rather complicated, due to its computational procedure which may be prone to local minima, as it employs an Alternating Least Squares (ALS) approach, and due to the difficulties associated with the choice of the tensor rank, R. For these reasons, two approaches to ease the computational burden on the CPD were introduced. The first one proposed to exploit the so-called TT-CPD duality to reduce the CPD of higher order tensors to computing the CPD of 3-rd order tensors. This was done by operating on a tensor in the TT-format, then appropriately converting it to the CPD format. The proposed approach was shown to reduce the complexity of a "direct" CPD from $\mathcal{O}(NRI^N)$ to $\mathcal{O}(NIR^3)$. The second approach focused instead on the choice of the tensor rank, R, by introducing a lower bound on its value. This was done by considering various tensor unfoldings, and it was shown that the highest possible bound is obtained as the matrix rank of the maximally square unfolding of the underlying tensor. Mild conditions for the attainment of such a bound were also provided, allowing for rank detectability, the significance of which was shown through numerical examples.

- **Chapter 5: Tensor Network Summation.** This was the first chapter of **Part II: Tensors for Shallow Learning**. This chapter heavily relied on the concepts from **Chapter 3** and introduced a framework to combine TNs via summation, motivated by the feature localization associated with TNs which results in feature fusion through their combination. It was found that summation is achieved through an intelligent block arrangement of the corresponding TN cores, and that two or more TNs can be summed if they present physical modes with equal dimensions and if they are isomorphic (same topology). The properties of TN summation were formalized by introducing the inverse (subtraction) operator and the identity element. Benefits of the proposed framework were first demonstrated by devising an algorithm for image classification which was tested on the ETH-80 dataset. Further advantages of the proposed framework were explored in light of eigenvalue and eigenvector extraction for huge matrices. This was achieved through employment of the Rayleigh Quotient (RQ) expressed in tensor format, followed by an implementation of Hotelling's deflation executed directly in TN format. This was shown to allow to iteratively extract several largest eigenvalues and associated eigenvectors of matrices of high dimensionality, which is of practical significance in applications.

- **Chapter 6: The Support Tensor Machine and its Applications to Financial Forecasting.** This chapter focused on the tensor extension of the Support Vector Machine (SVM),

namely the Support Tensor Machine (STM). A description of the technique was first provided, with aiding graphical representation where required. A computational efficient least-squares formulation of the STM (LS-STM) was considered, and a novel method to allow kernel usage in LS-STM in an analogous way to SVM and LS-SVM was introduced. The usefulness of the LS-STM was investigated within the financial application of forecasting the daily direction of movement of price of the S&P 500 financial index, using the implied index of volatility, VIX, the GC1 gold commodity, and the S&P 500 itself as predictors. Results were interpreted probabilistically via Platt scaling, which demonstrated robustness to parameter changes. Furthermore, evaluation of the LS-STM with common financial metrics showed its superiority over standard SVM.

- **Chapter 7: Compression and Interpretability of Neural Networks via the Tucker Tensor Layer.** This was the first chapter of **Part III: Tensors for Deep Learning**. Chapter 7 aimed to help resolve the two main stumbling blocks in the application of Deep Neural Networks (DNNs), that is, the exceedingly large number of trainable parameters and their physical interpretability. To achieve this, a tensor valued approach was employed by introducing the Tucker Tensor Layer (TTL) as an alternative to the dense weight-matrices of DNNs, which allowed the weight-matrices of general DNNs to be treated as a matrix unfolding of a higher order weight-tensor. In this chapter, the tensor-valued backpropagation algorithm was derived in terms of the tensor factors, and verified through rigorous mathematical analysis. The proposed framework was validated on synthetic data, and the MNIST, Fashion-MNIST and CIFAR-10 benchmark datasets. Experiments showed that, owing to the expressive power of tensors and in particular the TKD, the proposed framework was able to provide insights into the network training process, by providing explicit information on the relative importance of each data mode in training. The exploitation of the relative importance of data modes made it possible to train a data-augmented network comprising a simplified version of the VGG-16, which resulted in a 10% decrease in training time, at a small cost in terms of final accuracy. Moreover, the compression benefits of the proposed model were demonstrated on both the MNIST and Fashion-MNIST datasets, with the TTL achieving a 66.63 fold compression while maintaining comparable performance to the uncompressed NN.

- **Chapter 8: Tensor-Train Recurrent Neural Networks for Financial Forecasting.** This is the last chapter of **Part III** and of this thesis. Differently from Chapter 7, a more practical-oriented approach was adopted, and results were focused on financial applications, analogously to Chapter 6. In this chapter, the Tensor-Train decomposition was merged with Recurrent Neural Networks (RNNs), known to be the de facto standard machine learning tool for sequence modelling, to yield the so called TT-RNN. This was applied to forecasting the Japanese Yen, using four different asset classes as predictors. Simulation results demonstrated the superiority of TT-RNNs over standard RNNs, in terms of enhanced regularization, model accuracy, and generated profits. Similarly to the previous chapter, it is shown through the analysis of the gradients of the TT-factors how the TT-RNN model aids the interpretability of results, thus mitigating the notorious "black-box" issue associated with neural networks.

9.2 Future Work

While this thesis has addressed the development of novel tensor algorithms and demonstrated their utility in several applications, this thesis also provides many opportunities for extending its scope. In this section, some pertinent ideas that allow for the generalisation of the concepts explored thus far shall be briefly outlined.

Find further practical applications for TN summation. While the theory for TN summation was developed in Chapter 5, and the technique was applied to feature fusion and eigenvalue extraction, opportunities for further applications exist. For example, it may be explored how the summation framework can be integrated with other tensor decompositions, such as the CPD and the TT. The question of how to integrate TN summation and neural networks is also still open.

Extensions of the Support Tensor Machine. This thesis has considered the least-squares interpretation of the STM, and proposed a way to allow the usage of the Radial Basis Function (RBF) kernel which further relies on a least-squares approximation. It would be interesting to expand on this method, and formalize the limits and benefits of such an approximation. Leveraging on the multilinearity of tensors could indeed show that there is no need to perform higher dimensional mappings, as all relevant information is already contained in the structure of the data. A further extension could be related to asymptotic behaviours. For instance Support Vector Machines are known to exhibit asymptotic behaviours as the sample size, M, and the number of features K, tend to infinity, provided that K/M stays constant [166]. Extending this to tensors comes as a natural research question, whereby one could also consider how the order, N, of the multi-dimensional data in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ affects the asymptotic behaviour. As with SVM, this may provide practical benefits in parameter tuning, significantly reducing computational costs.

Integration of Tensors and Reinforcement Learning. Reinforcement Learning (RL) has been a long-standing active area of research within the machine learning community, in particular its Actor-Critic agents [167]. The question of integrating tensors with RL remains an open problem. An RL framework is characterized by an agent, an environment, and a state, which is most of the times encoded as a vector. The effectiveness and applicability of a state tensor, able to leverage the compression properties and interpretability of tensors, is yet to be explored. This could provide novel opportunities in the field of RL and machine learning in general, whereby an agent could also find its roots in tensor decompositions. For instance, one could define a state tensor and employ as agents neural networks containing the Tucker Tensor Layer from **Chapter 7**.

Bridging the Gap between Graphs and Tensors. Graph signal processing has recently been rediscovered as an active area of research within the signal processing community [2, 168, 169]. Despite advancements, it is still unclear how tensors can be linked with graphs, or whether, in fact, they represent different sides to the same coin. One of the greatest challenges inherent to tensors is the question of how to tensorize the data, as tensorization may rely on some specific assumptions, as outlined in **Chapter 2**. On the other hand, graph are natural tools to represent data having some notion of spatial information. One could hence ask the question of whether

graphs may be employed to meaningfully tensorize spatial data, to allow for an effective use of tensor methods. This point is presented and expanded in **Appendix A**.

Appendix A

Graph Tensorization

A.1 Perspective

As outlined in the **Introduction** and in **Chapter 6**, there is no unique or right way to perform tensorization. Appropriate methods to tensorize data are usually data-specific, and aim to leverage on the structure within the data. For example, if the data are heavily correlated one could perform the Hankelization approach, or if some data elements are believed to be interrelated in some way, tensors can be artificially constructed via simple stacking, as was the case for the problem of forecasting the S&P 500.

An interesting scenario occurs if the data contains spatial information. This could be the case of power grids, EEG sensors, or weather stations. Spatial data can be effectively represented using graphs, mathematical structures describing the interaction of single elements in space. In the following, it is shown how tensors can be thought of as a special kind of graph, and how graph properties may be exploited to perform tensorization based on spatial information.

A.2 Theoretical Background

This section outlines the mathematical notation and notions on Graph Theory necessary for the analysis.

A.2.1 Preliminaries

Given square matrices $\mathbf{A} \in \mathbb{R}^{N\times N}$ and $\mathbf{B} \in \mathbb{R}^{M\times M}$, their *Kronecker sum*, denoted by $\oplus$, is defined as

$$\mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{I}_N + \mathbf{I}_M \otimes \mathbf{B} = \mathbf{C} \in \mathbb{R}^{NM\times NM} \tag{A.1}$$

where the symbol $\otimes$ denotes the Kronecker product. Note that, due to the rules of the Kronecker product, the Kronecker sum is non-commutative. It is also useful to recall here impor-

tant rules of the trace operator for square matrices,

$$\mathrm{tr}(\mathbf{A}+\mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B}) \tag{A.2}$$
$$\mathrm{tr}(c\mathbf{A}) = c\,\mathrm{tr}(\mathbf{A}) \tag{A.3}$$
$$\mathrm{tr}(\mathbf{A}) = \mathrm{tr}(\mathbf{A}^T) \tag{A.4}$$
$$\mathrm{tr}(\mathbf{ABC}) = \mathrm{tr}(\mathbf{CAB}) = \mathrm{tr}(\mathbf{BCA}) \text{ (cyclic property)} \tag{A.5}$$
$$\mathrm{tr}(\mathbf{A}\otimes\mathbf{B}) = \mathrm{tr}(\mathbf{A})\,\mathrm{tr}(\mathbf{B}) \tag{A.6}$$

Moreover, the symbol " : ", will refer to the *Frobenius inner product* between matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{N\times N}$, defined as

$$\mathbf{A} : \mathbf{B} \equiv \mathrm{tr}(\mathbf{A}^T\mathbf{B}) \tag{A.7}$$

Also, recall that the outer product of vectors $\mathbf{x}^{(n)} \in \mathbb{R}^{I_n}$, $n = 1, \ldots, N$, yields yields a rank-1 tensor of order N, that is

$$\boldsymbol{\mathcal{X}} = \mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \cdots \circ \mathbf{x}^{(N)} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N} \tag{A.8}$$

with entries

$$x_{i_1 i_2 \ldots i_N} = x^{(1)}_{i_1} x^{(2)}_{i_2} \cdots x^{(N)}_{i_N}, \tag{A.9}$$

whereby the vectors $\mathbf{x}^{(n)}$ are referred to as Canonical Polyadic (CP) factors.

A.2.2 Graphs

Following the notation employed in [2, 170], a *graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined as a set of N vertices $\mathcal{V} = \{1, 2, \ldots, N\}$, which are connected through a set of edges, $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$. The *order* of a graph, $\mathcal{G}$, is the number of vertices it has, denoted by $|\mathcal{G}| = N$, while the existence of an edge between vertices m and n is denoted by $(m, n) \in \mathcal{E}$.

The graph connectivity of an N-vertex graph is represented by the *adjacency matrix*, $\mathbf{A} \in \mathbb{R}^{N\times N}$, whereby the vertex connectivity is described by

$$A_{mn} = \begin{cases} 1, & (m,n) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{A.10}$$

In regards to the directionality of vertex connection, a graph can be *undirected* or *directed*. A graph is undirected if $(m, n) \in \mathcal{E} \iff (n, m) \in \mathcal{E}$, that is, if $\mathbf{A} = \mathbf{A}^T$. In general this property does not hold for directed graphs.

The edges may also convey information about the relative importance of connections among vertices via a *weighted* graph. Analogously to the adjacency matrix, the *weight matrix*, $\mathbf{W} \in \mathbb{R}^{N\times N}$, corresponds topologically to the set of edges, $\mathcal{E}$, with elements $W_{mn} \geq 0$. A non-zero element designates both the existence of an edge (m, n), and the relative importance of the connection between vertices and m and n, while an element $W_{mn} = 0$ implies no edge (m, n).

The *Cartesian product* $\mathcal{G}_1 \square \mathcal{G}_2$ of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$, $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$, yields a graph $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2 = (\mathcal{V}, \mathcal{B})$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ is the direct product of the sets $\mathcal{V}_1$, $\mathcal{V}_2$, and $((m_1, n_1), (m_2, n_2)) \in \mathcal{B}$ iff

$$\begin{aligned} &m_1 = m_2 \text{ and } (n_1, n_2) \in \mathcal{B}_2 \text{ or} \\ &n_1 = n_2 \text{ and } (m_1, m_2) \in \mathcal{B}_1 \end{aligned} \tag{A.11}$$

The adjacency matrix of $\mathcal{G}$, $\mathbf{A}$, is the Kronecker sum of the respective adjacency matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathbf{A}_1$ and $\mathbf{A}_2$. That is, $\mathbf{A} = \mathbf{A}_1 \oplus \mathbf{A}_2$. An illustrations of the graph Cartesian product is shown in Figure A.1.

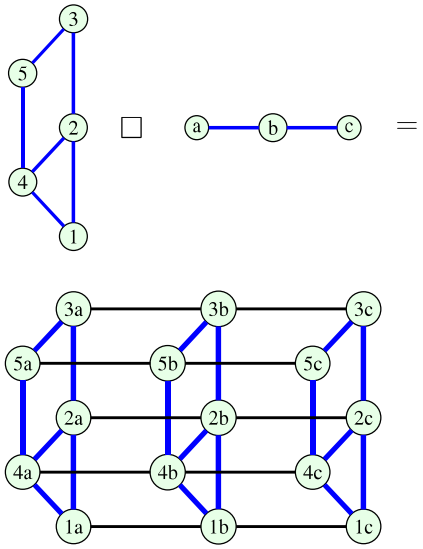


Figure A.1: Cartesian product of two graphs. Figure adapted from [2].

A.3 Graph-Tensor Duality

This section establishes a link between tensors and graphs:

1. By considering tensors as a collection of samples on regular lattices;
2. By highlighting how the connectivity of the Cartesian product of path graphs can be interpreted as rank-1 tensors.

A.3.1 Sampling on Regular Lattices

To establish a relationship between graph signals and tensors, we begin by considering a P-vertex graph, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For each vertex on the graph, a variable (signal), denoted by $x(n) \in \mathbb{R}$, which maps a vertex number, $n \in \mathcal{V}$, can be associated to a real number, that is, $x : \mathcal{V} \mapsto \mathbb{R}$. In other words, each vertex represents a scalar-valued field in a single-dimensional coordinate system. When considering all P vertices in $\mathcal{V}$, one can form the vector, $\mathbf{x} \in \mathbb{R}^P$, which defines the mapping $\mathbf{x} : \mathcal{V} \mapsto \mathbb{R}^P$.

If the topology of the graph takes place in an N-dimensional space, $\mathbb{R}^N$, then each vertex number, $n \in \mathcal{V}$, has a one-to-one correspondence to a unique coordinate vector in this space, denoted by $(i_1, \dots, i_N) \in \mathbb{R}^N$, where i_n is the coordinate associated with the m-th axis. In other words, there exists a unique mapping $n \mapsto (i_1, \dots, i_N)$, that is, each vertex can be defined equivalently as $x(n) \equiv x(i_1, \dots, i_N) \in \mathbb{R}$.

When discrete points in $\mathbb{R}^N$, are sampled using a regular lattice of dimensions $(I_1 \times \cdots \times I_N)$, thereby sampling a total of $\prod_{n=1}^{N} I_n \equiv P$ discrete points, the collection of samples naturally form the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, with the $(i_1, \dots, i_N)$-th entry defined as

$$\boldsymbol{\mathcal{X}}(i_1, \dots, i_N) = x(i_1, \dots, i_N) \quad i_n \in \mathbb{N}, \quad \forall n \tag{A.12}$$

Figure A.2 (same as Figure 8.1, but further reported here for better clarity) illustrates a collection of discrete points in a $3D$-coordinate system, which together form a 3-rd order tensor. This procedure is referred to as *tensorization*. As previously seen, tensorization can be performed:

1. By directly rearranging vectors or matrices into higher-order tensors, most effective if the original data admits low-rank decompositions [49, 50];
2. Through mathematical construction and augmentation [20, 23, 52];
3. Via experiment design by stacking together multifaceted data [53, 93].

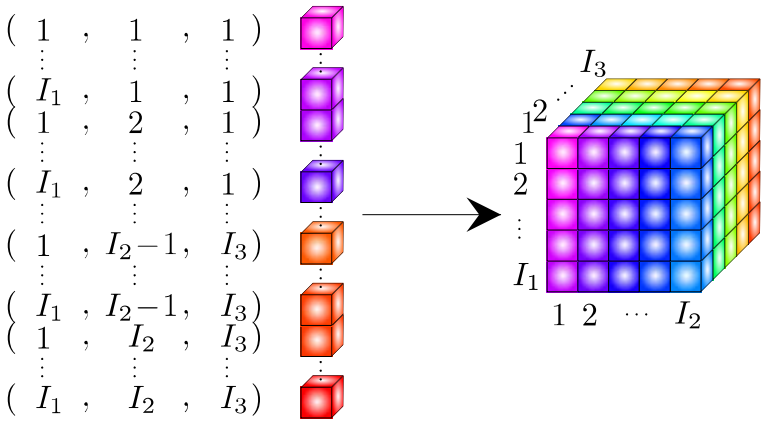


Figure A.2: Tensorization of discrete samples in $\mathbb{R}^3$.

Moreover, if the underlying continuous field, $x : \mathbb{R}^N \mapsto \mathbb{R}$, is defined as a *multilinear map* of the form

$$x : \underbrace{\mathbb{R} \times \cdots \times \mathbb{R}}_{N \text{ times}} \mapsto \mathbb{R} \tag{A.13}$$

then it is said to be *linearly separable*, and therefore admits the following decomposition

$$x(i_1, \dots, i_N) = x_1(i_1) \cdots x_N(i_N) \tag{A.14}$$

which is analogous to the rank-1 CPD in (A.8) and (A.9).

A.3.2 Cartesian Product of Path Graphs vs Rank-1 Tensors

It is next shown that the tensor structure inherent to $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be modelled naturally as a graph. This is achieved by exploiting the well-known property whereby lattice-structured graphs can be decomposed into constituent single-dimensional path graphs. An undirected *path graph* is a graph composed of N vertices, such that $(n, m) \in \mathcal{E} \iff m = n \pm 1$, for $n = 2, \ldots, N-1$; for the limit cases $n = 1$ and $n = N$, the conditions $(1, 2) \in \mathcal{E}$ and $(N, N-1) \in \mathcal{E}$ must hold. The Cartesian product of N disjoint I_n-vertex path graphs, $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$ for $n = 1, \ldots, N$, yields a graph with an N-dimensional regular lattice structure, denoted by $\mathcal{G} = \mathcal{G}_1 \,\square\, \cdots \,\square\, \mathcal{G}_N = (\mathcal{V}, \mathcal{E})$. In this way, the resulting vertex set takes the form $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_N$, and the resulting graph contains a total of $\prod_{n=1}^{N} I_n \equiv P$ vertices.

For illustration purposes, Figure A.3 shows the Cartesian product of 3 disjoint path graphs, which results in a graph, $\mathcal{G}$, with 3-dimensional lattice structure. The resulting graph would naturally represent the connectivity between the entries of a 3-rd order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{2 \times 3 \times 2}$. A visual comparison of Figure 8.1 and Figure A.3 asserts that they are equivalent representation of an original data vector $\mathbf{x} \in \mathbb{R}^P$, the elements of which are arranged on a regular lattice. Figure 8.1 shows a general tensorization of the data, whereas Figure A.3 presents an underlying physical structure inherent to the interaction of path graph factors. Specifically, it represents a rank-1 tensor in terms of its CP factors, which in turn can be represented as path graphs, that is

$$\boldsymbol{\mathcal{X}}_{\mathcal{G}} = \mathbf{x}^{(1)}_{\mathcal{G}_1} \circ \mathbf{x}^{(2)}_{\mathcal{G}_2} \circ \mathbf{x}^{(3)}_{\mathcal{G}_3} \tag{A.15}$$

where $|\mathcal{G}| = 12$, $|\mathcal{G}_1| = 2$, $|\mathcal{G}_2| = 3$ and $|\mathcal{G}_3| = 2$.

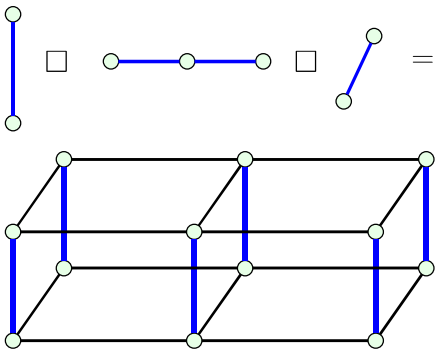
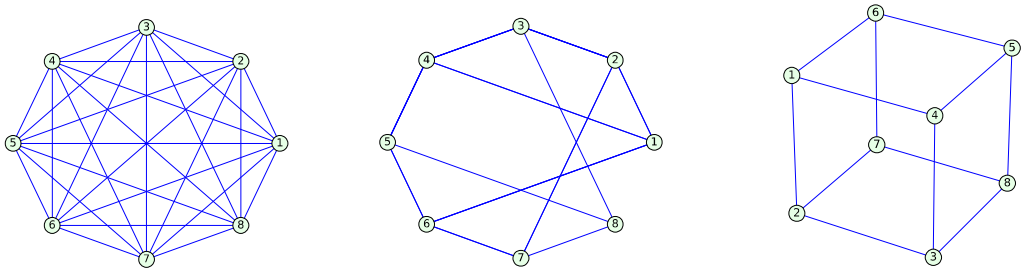


Figure A.3: Cartesian product of 3 disjoint path graphs.

Remark A.1. A new method for tensorization that has its foundations in data spatial information can hence be proposed.

Given data $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^P$ originally represented on a graph $\mathcal{G}$, the objective is to find the N path



(a) $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^P$ on graph $\mathcal{G}$, where $|\mathcal{G}| = P = 8$.

(b) $\mathcal{G} \approx \widetilde{\mathcal{G}} = \mathcal{G}_1 \square \mathcal{G}_2 \square \mathcal{G}_2$, where $|\mathcal{G}_n| = 2$ for $n = 1, 2, 3$.

(c) $\widetilde{\mathcal{G}}$ rearranged as a 3-rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{2\times 2\times 2}$

Figure A.4: Tensorizing data, $\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^P$, originally on graph $\mathcal{G}$, through the best path graph approximation.

graph factors that best describe $\mathcal{G}$ by

$$\mathcal{G} \approx \widetilde{\mathcal{G}} = \mathcal{G}_1 \square \mathcal{G}_2 \square \cdots \square \mathcal{G}_N \tag{A.16}$$

The tensorization is then inferred by

$$\mathbf{x}_{\mathcal{G}} \in \mathbb{R}^P \leftrightarrow \mathcal{G} \approx \widetilde{\mathcal{G}} = \bigsquare_{n=1}^{N} \mathcal{G}_n \leftrightarrow \boldsymbol{\mathcal{X}}_{\widetilde{\mathcal{G}}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \tag{A.17}$$

where $|\mathcal{G}_n| = I_n$, $|\mathcal{G}| = \prod_{n=1}^{N} I_n = P$. This is illustrated in Figure A.4.

Remark A.2. The fundamental assumption here is that the original data is characterized by a structured spatial information. In other words, it can be represented as a graph $\mathcal{G}$ with P vertices and associated adjacency matrix $\mathbf{A}$ or weight matrix $\mathbf{W}$. If the weight matrix of the n-th path graph, $\mathcal{G}_n$, is denoted by $\mathbf{A}_n \in \mathbb{R}^{I_n \times I_n}$, then the weight matrix of the resulting M-dimensional regular lattice graph, $\mathcal{G}$, is given by

$$\mathbf{W} = (\mathbf{A}_1 \oplus \cdots \oplus \mathbf{A}_M) \in \mathbb{R}^{P \times P} \tag{A.18}$$

A.4 Graph Separability

Following the discussion in the previous section, given a graph $\mathcal{G}$, $|\mathcal{G}| = P$, the task is to find its best N-path graph approximation, as expressed in (A.16). Due to (A.18), this is equivalent to finding the best Kronecker sum approximation of its weight matrix $\mathbf{W} \in \mathbb{R}^{P \times P}$.

Hence the problem can be formulated as,

$$\begin{aligned}&\underset{\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}}{\arg\min} \left|\left|\mathbf{W} - \Big(\bigoplus_{n=1}^{N} \mathbf{A}^{(n)}\Big)\right|\right|_F^2 \\ &\text{s.t. } \mathbf{W} \text{ and } \mathbf{A}^{(n)} \text{are weight matrices,} \\ &\text{for } n = 1, \ldots, N\end{aligned} \tag{A.19}$$

where $\mathbf{W} \in \mathbb{R}^{P \times P}$, $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ and $P = \prod_{n=1}^{N} I_n$. The above can be solved by considering a specific case for which the solution can be generalized. For $N = 2$, we can write

$$\begin{aligned}&\underset{\mathbf{A},\mathbf{B}}{\arg\min}\, J(\mathbf{A}, \mathbf{B}) = \\ &\underset{\mathbf{A},\mathbf{B}}{\arg\min} \left|\left|\mathbf{W} - \Big(\mathbf{A} \oplus \mathbf{B}\Big)\right|\right|_F^2 \\ &\text{s.t. } \mathbf{W}, \mathbf{A}, \mathbf{B} \text{ are weight matrices}\end{aligned} \tag{A.20}$$

where $\mathbf{A} \in \mathbb{R}^{I_1 \times I_1}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times I_2}$. The cost function $J(\mathbf{A}, \mathbf{B})$ can be rewritten as

$$\begin{aligned}&J(\mathbf{A}, \mathbf{B}) = \left|\left|\mathbf{W} - \mathbf{A} \otimes \mathbf{I}_{I_2} - \mathbf{I}_{I_1} \otimes \mathbf{B}\right|\right|_F^2 \\ &= \operatorname{tr}(\mathbf{W}^2) + 2\operatorname{tr}(\mathbf{A} \otimes \mathbf{B}) + \operatorname{tr}(\mathbf{A}^2 \otimes \mathbf{I}_{I_2}) + \operatorname{tr}(\mathbf{I}_{I_1} \otimes \mathbf{B}^2) - 2\operatorname{tr}\big((\mathbf{A} \otimes \mathbf{I}_{I_2})\mathbf{W}\big) - 2\operatorname{tr}\big((\mathbf{I}_{I_1} \otimes \mathbf{B})\mathbf{W}\big)\end{aligned} \tag{A.21}$$

where the cyclic property of the trace has been applied as well as the facts that $\mathbf{A} = \mathbf{A}^T$, $\mathbf{B} = \mathbf{B}^T$, and $\mathbf{W} = \mathbf{W}^T$. By removing the terms independent of $\mathbf{A}$ and $\mathbf{B}$ and by noting that $\operatorname{tr}(\mathbf{A}) = \operatorname{tr}(\mathbf{B}) = 0$, the minimization problem in (A.20) degenerates to

$$\begin{aligned}&\underset{\mathbf{A},\mathbf{B}}{\arg\min}\, J^*(\mathbf{A}, \mathbf{B}) = \\ &\underset{\mathbf{A},\mathbf{B}}{\arg\min} \Big(I_2 \operatorname{tr}(\mathbf{A}^2) + I_1 \operatorname{tr}(\mathbf{B}^2) - 2\operatorname{tr}\big((\mathbf{A} \otimes \mathbf{I}_{I_2})\mathbf{W}\big) - 2\operatorname{tr}\big((\mathbf{I}_{I_1} \otimes \mathbf{B})\mathbf{W}\big)\Big)\end{aligned} \tag{A.22}$$

where the trace of a Kronecker product property in (A.6) has been employed.

Because $J^*(\mathbf{A}, \mathbf{B})$ is a linear combination of functions of $\mathbf{A}$ and of functions of $\mathbf{B}$, its global minimum can be computed by minimizing its partial derivatives separately [155, 171, 172]. The minimum to (A.22) w.r.t. $\mathbf{A}$ is attained by computing

$$\frac{\partial J^*}{\partial \mathbf{A}} = 2I_2\mathbf{A} - 2\mathbf{\Omega}^{\mathbf{A}} = 0 \implies \mathbf{A} = \frac{1}{I_2}\mathbf{\Omega}^{\mathbf{A}} \tag{A.23}$$

where $\mathbf{\Omega}^{\mathbf{A}} = \frac{\partial \operatorname{tr}((\mathbf{A} \otimes \mathbf{I}_{I_2})\mathbf{W})}{\partial \mathbf{A}} \in \mathbb{R}^{I_1 \times I_1}$, which can be computed by making use of the Frobenius inner product notation. Define

$$\phi = \mathbf{W}^T : \mathbf{A} \otimes \mathbf{I}_{I_2} = \mathbf{W} : \mathbf{A} \otimes \mathbf{I}_{I_2} \tag{A.24}$$

The Pitsianis decomposition [173] is employed to express $\mathbf{W}$ as a sum of Kronecker products

$$\mathbf{W} = \sum_i \mathbf{Y}_i \otimes \mathbf{Z}_i \tag{A.25}$$

where $\mathbf{Y}_i \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{Z}_i \in \mathbb{R}^{I_2 \times I_2}$. Substituting (A.25) in (A.24) and taking the differential yields

$$\begin{aligned} \mathrm{d}\phi &= \sum_i \mathbf{Y}_i \otimes \mathbf{Z}_i : \mathrm{d}\mathbf{A} \otimes \mathbf{I}_{I_2} \\ &= \sum_i (\mathbf{Z}_i : \mathbf{I}_{I_2})(\mathbf{Y}_i : \mathrm{d}\mathbf{A}) = \mathrm{d}\mathbf{A} \sum_i \mathrm{tr}(\mathbf{Z}_i)\mathbf{Y}_i \end{aligned} \tag{A.26}$$

which results in

$$\frac{\mathrm{d}\phi}{\mathrm{d}\mathbf{A}} = \mathbf{\Omega}^{\mathbf{A}} = \sum_i \mathrm{tr}(\mathbf{Z}_i)\mathbf{Y}_i \tag{A.27}$$

Analogously,

$$\frac{\partial J^*}{\partial \mathbf{B}} = 2I_1\mathbf{A} - 2\mathbf{\Omega}^{\mathbf{B}} = 0 \implies \mathbf{B} = \frac{1}{I_1}\mathbf{\Omega}^{\mathbf{B}} \tag{A.28}$$

where $\mathbf{\Omega}^{\mathbf{B}} = \frac{\partial\, \mathrm{tr}((\mathbf{I}_{I_1} \otimes \mathbf{B})\mathbf{W})}{\partial \mathbf{B}} \in \mathbb{R}^{I_2 \times I_2}$, which is computed by considering

$$\begin{aligned} \phi &= \mathbf{W} : \mathbf{I}_{I_1} \otimes \mathbf{B} \\ &= \sum_i \mathbf{Y}_i \otimes \mathbf{Z}_i : \mathbf{I}_{I_1} \otimes \mathbf{B} = \sum_i (\mathbf{Y}_i : \mathbf{I}_{I_1})(\mathbf{Z}_i : \mathbf{B}) \end{aligned} \tag{A.29}$$

where $\mathbf{Y}_i \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{Z}_i \in \mathbb{R}^{I_2 \times I_2}$. Computing the differential and taking the gradient yields

$$\frac{\mathrm{d}\phi}{\mathrm{d}\mathbf{B}} = \sum_i \mathrm{tr}(\mathbf{Y}_i)\mathbf{Z}_i \tag{A.30}$$

Reconsider now the original problem in (A.19). Upon its expansion following the same procedure as for $N = 2$, we arrive at

$$\begin{aligned} &\underset{\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}}{\arg\min}\ J(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}) = \\ &\underset{\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}}{\arg\min}\ \mathrm{tr}(\mathbf{W}^2) - 2\Bigg[\mathrm{tr}\left(\mathbf{A}^{(1)} \otimes \mathbf{I}_{\frac{P}{I_N}}\right) + \sum_{i=1}^{N-1} \mathrm{tr}\left(\mathbf{I}_{\prod_{n=1}^{i-1} I_n} \otimes \mathbf{A}^{(i)} \otimes \mathbf{I}_{\prod_{n=i+1}^{N} I_n}\right) + \\ &+ \mathrm{tr}\left(\mathbf{I}_{\frac{P}{I_N}} \otimes \mathbf{A}^{(N)}\right)\Bigg]\mathbf{W} + \sum_{i=1}^{N} \prod_{\substack{n=i \\ n\neq i}}^{N} I_n\, \mathrm{tr}((\mathbf{A}^{(i)})^2) \end{aligned} \tag{A.31}$$

The solution is found by solving for $\mathbf{A}^{(n)}$ in $\frac{\partial J}{\partial \mathbf{A}^{(n)}} = 0$, for $n = 1, \ldots, N$, where

$$\frac{\partial J}{\partial \mathbf{A}^{(n)}} = 2\prod_{i\neq n}^{N} I_i \mathbf{A}^{(n)} - 2\mathbf{\Omega}^{\mathbf{A}^{(n)}} \tag{A.32}$$

$$\boldsymbol{\Omega}^{\mathbf{A}^{(n)}} = \begin{cases} \frac{\partial \operatorname{tr}\left(\left(\mathbf{A}^{(1)} \otimes \mathbf{I}_{\frac{P}{I_N}}\right)\mathbf{W}\right)}{\partial \mathbf{A}^{(1)}}, & n = 1 \\ \frac{\partial \operatorname{tr}\left(\left(\mathbf{I}_{\frac{P}{I_1}} \otimes \mathbf{A}^{(N)}\right)\mathbf{W}\right)}{\partial \mathbf{A}^{(N)}}, & n = N \\ \frac{\partial \operatorname{tr}\left(\left(\mathbf{I}_Q \otimes \mathbf{A}^{(n)} \otimes \mathbf{I}_K\right)\mathbf{W}\right)}{\partial \mathbf{A}^{(n)}}, & \text{otherwise} \end{cases} \tag{A.33}$$

with $Q = \prod_{i=1}^{n-1} I_i$, and $K = \prod_{i=n+1}^{N} I_i$. Gradients $\boldsymbol{\Omega}^{\mathbf{A}^{(1)}}$ and $\boldsymbol{\Omega}^{\mathbf{A}^{(N)}}$ are given according to (A.27) and (A.30), respectively, whereas $\boldsymbol{\Omega}^{\mathbf{A}^{(n)}}$ can found by considering

$$\begin{aligned} \phi &= \mathbf{W} : \mathbf{I}_Q \otimes \mathbf{A} \otimes \mathbf{I}_K = \sum_i \mathbf{Y}_i \otimes \mathbf{Z}_i : \mathbf{I}_Q \otimes \mathbf{A}^{(n)} \otimes \mathbf{I}_K \\ &= \sum_i \mathbf{Y}_i \otimes \sum_j \mathbf{X}_j^{(1)} \otimes \mathbf{X}_j^{(2)} : \mathbf{I}_Q \otimes \mathbf{A}^{(n)} \otimes \mathbf{I}_K \\ &= \sum_i \sum_j \mathbf{Y}_i \otimes \mathbf{X}_j^{(1)} \otimes \mathbf{X}_j^{(2)} : \mathbf{I}_Q \otimes \mathbf{A}^{(n)} \otimes \mathbf{I}_K \\ &= \sum_i \sum_j (\mathbf{Y}_i : \mathbf{I}_Q)(\mathbf{X}_j^{(1)} : \mathbf{A}^{(n)})(\mathbf{X}_j^{(2)} : \mathbf{I}_K) \end{aligned} \tag{A.34}$$

which implies

$$\frac{\mathrm{d}\phi}{\mathrm{d}\mathbf{A}^{(n)}} = \boldsymbol{\Omega}^{\mathbf{A}^{(n)}} = \sum_i \sum_j \operatorname{tr}(\mathbf{Y}_i) \operatorname{tr}(\mathbf{X}_j^{(2)}) \mathbf{X}_j^{(1)} \tag{A.35}$$

where $\mathbf{Y}_i \in \mathbb{R}^{Q \times Q}$, $\mathbf{Z}_i \in \mathbb{R}^{I_n K \times I_n K}$, $\mathbf{X}_j^{(1)} \in \mathbb{R}^{I_n \times I_n}$ and $\mathbf{X}_j^{(2)} \in \mathbb{R}^{K \times K}$.

The elementwise entries of the partial derivatives (A.27), (A.30) and (A.35) are obtained by inspection and provided in the following.

1. The partial derivative

$$\boldsymbol{\Omega}^{\mathbf{A}} = \frac{\partial \operatorname{tr}((\mathbf{A} \otimes \mathbf{I}_{I_2})\mathbf{W})}{\partial \mathbf{A}} \in \mathbb{R}^{I_1 \times I_1} \tag{A.36}$$

with $\mathbf{A} \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{W} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ has elementwise entries given by

$$\Omega_{ij}^{\mathbf{A}} = \begin{cases} 0, & i = j \\ \sum_{k=1}^{I_2} W_{(I_2(i-1)+k)(I_2(j-1)+k)}, & i \neq j \end{cases} \tag{A.37}$$

2. The partial derivative

$$\boldsymbol{\Omega}^{\mathbf{B}} = \frac{\partial \operatorname{tr}((\mathbf{I}_{I_1} \otimes \mathbf{B})\mathbf{W})}{\partial \mathbf{B}} \in \mathbb{R}^{I_2 \times I_2} \tag{A.38}$$

with $\mathbf{B} \in \mathbb{R}^{I_2 \times I_2}$ and $\mathbf{W} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ has elementwise entries given by

$$\Omega_{ij}^{\mathbf{B}} = \begin{cases} 0, & \text{if } i = j \\ \sum_{k=0}^{I_1 - 1} W_{(i+kI_1)(j+kI_1)}, & \text{if } i \neq j \end{cases} \tag{A.39}$$

3. The partial derivative

$$\mathbf{\Omega}^{\mathbf{A}^{(n)}} = \frac{\partial \operatorname{tr}\left(\left(\mathbf{I}_Q \otimes \mathbf{A}^{(n)} \otimes \mathbf{I}_K\right)\mathbf{W}\right)}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{I_n \times I_n} \tag{A.40}$$

with $Q = \prod_{i=1}^{n-1} I_i$, and $K = \prod_{i=n+1}^{N} I_i$, $\mathbf{W} \in \mathbb{R}^{QI_nK \times QI_nK}$ has elementwise entries given by

$$\Omega_{ij}^{\mathbf{A}^{(n)}} = \sum_{q=0}^{Q-1} \sum_{k=1}^{K} W_{\left(K(i-1)+k+qI_nK\right)\left(K(i-1)+k+qI_nK\right)} \tag{A.41}$$

A.4.1 A Note on Computational Costs

Despite the rigour of the above method, the computational requirements for solving such a problem are prohibitive, and require further study. This is due to the fact that the original weight matrix, $\mathbf{W} \in \mathbb{R}^{P \times P}$ is dependent on the numbering of the corresponding graph's nodes, which has $P!$ possible combinations. It has not yet been shown whether the above method is permutation invariant. This, combined with the facts that the same argument applies to the path graph's factor matrices and that the Kronecker sum in non-commutative, leads to a number of possible solutions which is, as it stands, a major obstacle to Graph Tensorization.

A.4.2 Concluding Remarks

A method to tensorize spatial data, referred to as Graph Tensorization, has been proposed. This was achieved by exploiting the duality between the Cartesian product of path graphs and rank-1 tensors, allowing to find the desired tensorization as the solution to an optimization problem involving the original graph's weight matrix and the Kronecker sum of the corresponding weight matrices of the path graph factors. Further research on the computational complexity on such method is still required to ensure its applicability.

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. 7, 19, 20, 21, 22, 23, 24, 25, 27, 41, 46, 49

[2] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, and T. Constantinides, "Graph signal processing–Part I: Graphs, graph spectra, and spectral clustering," *arXiv preprint arXiv:1907.03467*, 2019. 10, 140, 143, 144

[3] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," *ArXiv e-prints*, Mar. 2014. 19, 24, 38, 39, 41, 83

[4] M. M. Gobble, "Big data: The next big thing in innovation," *Research-Technology Management*, vol. 56, no. 1, pp. 64–67, 2013. 19

[5] P. P. Tallon, "Corporate governance of big data: Perspectives on value, risk, and cost," *Computer*, vol. 46, no. 6, pp. 32–38, 2013. 19

[6] "The 4Vs of Big Data," https://www.ibmbigdatahub.com/infographic/four-vs-big-data. 19

[7] H. Fanaee-T and J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey," *Knowledge-Based Systems*, vol. 98, pp. 130–147, 2016. 20

[8] J. A. Schouten, *Ricci-calculus: An Introduction to Tensor Analysis and its Geometrical Applications*. Springer Science & Business Media, 2013, vol. 10. 20

[9] A. J. Kox, M. J. Klein, and R. Schulmann, "The collected papers of Albert Einstein; Volume 6 the Berlin years: Writings 1914-1917," *European Journal of Physics*, vol. 18, no. 1, p. 52, 1997. 20

[10] 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. 20, 25, 34

[11] 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. 20, 34

[12] ——, "The three basic factor-analytic research designs—their interrelations and derivatives," *Psychological Bulletin*, vol. 49, no. 5, p. 499, 1952. 20

[13] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," in *Problems in Measuring Change*, C. W. Harris, Ed. Madison WI: University of Wisconsin Press, 1963, pp. 122–137. 20, 37

[14] ——, "The extension of factor analysis to three-dimensional matrices," in *Contributions to Mathematical Psychology.*, H. Gulliksen and N. Frederiksen, Eds. New York: Holt, Rinehart and Winston, 1964, pp. 110–127. 20

[15] ——, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966. 20

[16] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" Multimodal Factor Analysis," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970. 20, 34

[17] J. D. Carroll and J.-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. 20, 34

[18] A. Cichocki, D. P. Mandic, A. H. Phan, C. F. Caiafa, G. Zhou, Q. Zhao, and L. D. Lathauwer, "Tensor decompositions for signal processing applications," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015. 20, 21, 23, 24, 68, 91

[19] C. J. Appellof and E. R. Davidson, "Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents," *Analytical Chemistry*, vol. 53, no. 13, pp. 2053–2056, 1981. 20

[20] C. L. Nikias and A. P. Petropulu, *Higher-Order Spectral Analysis: A Nonlinear Signal Processing Framework*. Prentice Hall, 1993. 21, 56, 145

[21] J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non-Gaussian signals," in *IEE proceedings F (radar and signal processing)*, vol. 140, no. 6. IET, 1993, pp. 362–370. 21

[22] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994. 21

[23] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. Academic Press. Elsevier, 2010. 21, 145

[24] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000. 21, 25, 33, 38, 46, 91, 106

[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. 21

[26] J. Ballani, L. Grasedyck, and M. Kluge, "Black box approximation of tensors in hierarchical Tucker format," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 639–657, 2013. 21

[27] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011. 21, 24, 25, 39, 40, 57

[28] P. Nomikos and J. F. MacGregor, "Monitoring batch processes using multiway principal component analysis," *AIChE Journal*, vol. 40, no. 8, pp. 1361–1375, 1994. 21

[29] P. Barbieri, G. Adami, S. Piselli, F. Gemiti, and E. Reisenhofer, "A three-way principal factor analysis for assessing the time variability of freshwaters related to a municipal water supply," *Chemometrics and Intelligent Laboratory Systems*, vol. 62, no. 1, pp. 89–100, 2002. 21

[30] K. P. Singh, A. Malik, V. K. Singh, N. Basant, and S. Sinha, "Multi-way modeling of hydro-chemical data of an alluvial river system—A case study," *Analytica Chimica Acta*, vol. 571, no. 2, pp. 248–259, 2006. 21

[31] M. A. Engle, M. Gallo, K. T. Schroeder, N. J. Geboy, and J. W. Zupancic, "Three-way compositional analysis of water quality monitoring data," *Environmental and Ecological Statistics*, vol. 21, no. 3, pp. 565–581, 2014. 21

[32] S. Lee, H. Liu, M. Kim, J. T. Kim, and C. 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. 21

[33] I. Stanimirova and V. Simeonov, "Modeling of environmental four-way data from air quality control," *Chemometrics and Intelligent Laboratory Systems*, vol. 77, no. 1-2, pp. 115–121, 2005. 21

[34] K. P. Singh, A. Malik, V. K. Singh, and S. Sinha, "Multi-way data analysis of soils irrigated with wastewater–A case study," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 1, pp. 1–12, 2006. 21

[35] J. Andrade, M. Kubista, A. Carlosena, and D. Prada, "3-Way characterization of soils by Procrustes rotation, matrix-augmented principal components analysis and parallel factor analysis," *Analytica Chimica Acta*, vol. 603, no. 1, pp. 20–29, 2007. 21

[36] L. Tran, C. Navasca, and J. Luo, "Video detection anomaly via low-rank and sparse decompositions," in *2012 Western New York Image Processing Workshop*. IEEE, 2012, pp. 17–20. 21

[37] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 89–98, 2011. 21

[38] E. Acar, C. A. Bingol, H. Bingol, R. Bro, and B. Yener, "Seizure recognition on epilepsy feature tensor," in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 4273–4276. 22, 56

[39] 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, 2015. 22

[40] J. C. Ho, J. Ghosh, S. R. Steinhubl, W. F. Stewart, J. C. Denny, B. A. Malin, and J. Sun, "Limestone: High-throughput candidate phenotype generation via tensor factorization," *Journal of Biomedical informatics*, vol. 52, pp. 199–211, 2014. 22

[41] A. Baum, A. S. Meyer, J. L. Garcia, M. Egebo, P. W. Hansen, and J. D. Mikkelsen, "Enzyme activity measurement via spectral evolution profiling and PARAFAC," *Analytica Chimica Acta*, vol. 778, pp. 1–8, 2013. 22

[42] A. Ramanathan, P. K. Agarwal, and C. J. Langmead, "Using tensor analysis to characterize contact-map dynamics of proteins," Ph.D. dissertation, PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2008. 22

[43] Y. Bai, J. Tezcan, Q. Cheng, and J. Cheng, "A multiway model for predicting earthquake ground motion," in *Proceedings of the 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. IEEE, 2013, pp. 219–224. 22

[44] R. Bellman, "Curse of dimensionality," *Adaptive control processes: A guided tour. Princeton, NJ*, 1961. 22

[45] S. Klus and C. Schütte, "Towards tensor-based methods for the numerical approximation of the Perron-Frobenius and Koopman operator," *arXiv preprint arXiv:1512.06527*, 2015. 23

[46] S. Dolgov and D. Savostyanov, "Alternating minimal energy methods for linear systems in higher dimensions," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. A2248–A2271, 2014. 23, 31

[47] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*. New York: John Wiley & Sons Ltd., 2004. 23, 56

[48] W. Hackbusch, *Tensor spaces and numerical tensor calculus*. Springer Science & Business Media, 2012, vol. 42. 24

[49] 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. 24, 145

[50] 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. 24, 145

[51] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. John Wiley & Sons, 2009. 24, 54

[52] F. Miwakeichi, E. Martınez-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. 24, 68, 145

[53] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 810–823, 2000. 24, 145

[54] 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. 24

[55] 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. 24, 36

[56] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors – Part II: Uniqueness of the overall decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 876–903, 2013. 24

[57] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 442–450. 24, 25, 27, 106, 113, 127, 128, 130

[58] E. Stoudenmire and D. J. Schwab, "Supervised learning with tensor networks," *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 4799–4807, 2016. 24, 25

[59] A. Haliassos, K. Konstantinidis, and D. P. Mandic, "Supervised learning for non-sequential data with the canonical polyadic decomposition," *arXiv preprint arXiv:2001.10109*, 2020. 24

[60] 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. 24, 33, 41, 73, 84, 85, 91, 92

[61] T. G. Kolda, "Orthogonal tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 243–255, 2001. 25, 33

[62] ——, "Multilinear operators for higher-order decompositions," Sandia National Laboratories, Albuquerque, NM, Livermore, CA, Tech. Rep., 2006. 25

[63] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. 25, 26, 31, 33, 36, 37, 48, 56, 62, 91

[64] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000. 25, 33, 38

[65] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms—Part I: Lemmas for partitioned matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1022–1032, 2008. 25, 65

[66] ——, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-$(L_r, L_r, 1)$ terms," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1451–1474, 2011. 25

[67] 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. 25

[68] A. H. Phan and A. Cichocki, "Fast and efficient algorithms for nonnegative Tucker decomposition," pp. 772–782, 2008. 25

[69] S. Smith and G. Karypis, "Accelerating the Tucker decomposition with compressed sparse tensors," *Proceedings of the 23rd European Conference on Parallel Processing*, pp. 653–668, 2017. 25

[70] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," *arXiv preprint arXiv:1606.05535*, 2016. 25, 31, 56

[71] S. Aja-Fernández, R. de Luis Garcia, D. Tao, and X. Li, *Tensors in Image Processing and Computer Vision*. Springer Science & Business Media, 2009. 25

[72] H. Knutsson, C.-F. Westin, and M. Andersson, "Representing local structure using tensors II," *Proceedings of the 17th Scandinavian Conference on Image Analysis*, pp. 545–556, 2011. 25

[73] D. Picard and P.-H. Gosselin, "Improving image similarity with vectors of locally aggregated tensors," *Proceedings of the 18th IEEE International Conference on Image Processing*, pp. 669–672, 2011. 25

[74] G. Foster, "Time series analysis by projection. Tensor methods for time series analysis," *The Astronomical Journal*, vol. 111, p. 555, 1996. 25

[75] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of EEG signals: A brief review," *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015. 25

[76] J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *arXiv preprint arXiv:1707.08308*, vol. 2, no. 4, 2017. 25, 27, 106, 127

[77] X. Cao, G. Rabusseau, and J. Pineau, "Tensor regression networks with various low-rank tensor approximations," *arXiv preprint arXiv:1712.09520*, 2017. 25, 27, 106

[78] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," *arXiv preprint arXiv:1412.6553*, 2014. 25, 27, 106

[79] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015. 25, 106, 127

[80] G. G. Calvi, I. Kisil, A. Cichocki, A. H. Phan, and D. P. Mandic, "A tale of nodes and edges: Visualizing tensors and their basic operations," *Submitted to IEEE Signal Processing Magazine*, 2020. 26, 41, 48, 51, 52, 54

[81] G. G. Calvi, I. Kisil, and D. P. Mandic, "Feature fusion via tensor network summation," *Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, pp. 2623–2627, 2018. 26, 73, 91

[82] P. Comon, "Tensor decompositions: State of the art and applications, keynote address in IMA Conf." *Mathematics in Signal Processing, Warwick, UK*, 2000. 26, 33

[83] A.-H. Phan, A. Cichocki, I. Oseledets, G. G. Calvi, S. Ahmadi-Asl, and D. P. Mandic, "Tensor networks for latent variable analysis: Higher order canonical polyadic decomposition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 2174–2188, 2019. 26, 56, 57, 61

[84] G. G. Calvi, B. Scalzo Dees, and D. P. Mandic, "A lower bound on the tensor rank based on its maximally square matrix unfolding," *Elsevier Signal Processing*, no. 107862, 2020. 26, 56, 57, 62

[85] 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. 27, 33, 91

[86] G. G. Calvi, V. Lucic, and D. P. Mandic, "Support tensor machine for financial forecasting," *Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8152–8156, 2019. 27, 91, 95, 127, 128, 132

[87] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986. 27, 106, 112

[88] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao, and D. P. Mandic, "Compression and interpretability of deep neural networks via Tucker Tensor Layer: From first principles to tensor valued back-propagation," *Submitted to Neural Computation*, 2020. 27, 106

[89] Y. L. Xu, G. G. Calvi, and D. P. Mandic, "Tensor-train recurrent neural networks for financial forecasting," *Submitted to the Proceedings of the 46th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021. 27, 128, 130, 131, 132

[90] M. Boussé, 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. 33

[91] 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. 33

[92] 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. 33, 56

[93] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear image analysis for facial recognition," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 2. IEEE, 2002, pp. 511–514. 33, 145

[94] T. Kolda and B. Bader, "The TOPHITS model for higher-order web link analysis," *Proceedings of the International Workshop on Link Analysis, Counterterrorism and Security*, vol. 7, pp. 26–29, 2006. 34

[95] J. B. Kruskal, "Rank, decomposition, and uniqueness for 3-way and N-way arrays," *Multiway Data Analysis*, pp. 7–18, 1989. 36

[96] 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. 36

[97] R. Bro and H. A. Kiers, "A new efficient method for determining the number of components in PARAFAC models," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 17, no. 5, pp. 274–286, 2003. 36

[98] I. V. Oseledets and E. E. Tyrtyshnikov, "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. 39

[99] R. J. Trudeau, *Introduction to Graph Theory*. Courier Corporation, 2013. 50

[100] J. Haastad, "Tensor rank is NP-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990. 56, 62

[101] A. Yeredor, "Blind source separation via the second characteristic function," *Signal Processing*, vol. 80, no. 5, pp. 897–902, 2000. 56

[102] P. Comon and M. Rajih, "Blind identification of under-determined mixtures based on the characteristic function," *Signal Processing*, vol. 86, no. 9, pp. 2271–2281, 2006. 56

[103] F. Ju, Y. Sun, J. Gao, Y. Hu, and B. Yin, "Vectorial dimension reduction for tensors based on Bayesian inference," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4579–4592, 2017. 56

[104] Y. Luo, D. Tao, K. Ramamohanarao, C. Xu, and Y. Wen, "Tensor canonical correlation analysis for multi-view dimension reduction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3111–3124, 2015. 56

[105] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014. 57

[106] J. Zhang, J. Yu, and D. Tao, "Local deep-feature alignment for unsupervised dimension reduction," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2420–2432, 2018. 57

[107] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969. 57

[108] P. Tichavskỳ, A.-H. Phan, and A. Cichocki, "Numerical CP decomposition of some difficult tensors," *Journal of Computational and Applied Mathematics*, vol. 317, pp. 362–370, 2017. 57

[109] 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 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6299–6303. 57, 73

[110] A.-H. Phan, P. Tichavskỳ, and A. Cichocki, "Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4834–4846, 2013. 57

[111] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 23, no. 7-8, pp. 393–405, 2009. 57

[112] E. Sanchez and B. R. Kowalski, "Tensorial resolution: A direct trilinear decomposition," *Journal of Chemometrics*, vol. 4, no. 1, pp. 29–45, 1990. 59

[113] G. Matsaglia and G. PH Styan, "Equalities and inequalities for ranks of matrices," *Linear and Multilinear Algebra*, vol. 2, no. 3, pp. 269–292, 1974. 64

[114] 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. 65

[115] S. Mertens, "Phase transition in the number partitioning problem," *Physical Review Letters*, vol. 81, no. 20, p. 4281, 1998. 67

[116] F. F. Ferreira and J. F. Fontanari, "Probabilistic analysis of the number partitioning problem," *Journal of Physics A: Mathematical and General*, vol. 31, no. 15, p. 3417, 1998. 67

[117] S. Mertens, "The easiest hard problem: Number partitioning," *Computational Complexity and Statistical Physics*, vol. 125, no. 2, pp. 125–139, 2006. 67

[118] I. Kisil, A. Moniri, G. G. Calvi, B. Scalzo Dees, and D. P. Mandic, "HOTTBOX: Higher Order Tensor ToolBOX," https://github.com/hottbox. 69, 79

[119] 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, 2015. 73

[120] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition.*, vol. 2. IEEE, 2003, pp. II–409. 73

[121] Y. Saad, "Projection and deflation method for partial pole assignment in linear state feedback," *IEEE Transactions on Automatic Control*, vol. 33, no. 3, pp. 290–297, 1988. 73, 83

[122] 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. 82

[123] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen, "Computations in quantum tensor networks," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 750–781, 2013. 83

[124] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. 90, 94

[125] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001. 90

[126] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of Complexity*. Springer, 2015, pp. 11–30. 90

[127] M. P. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler, "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proceedings of the National Academy of Sciences*, vol. 97, no. 1, pp. 262–267, 2000. 91

[128] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the European Conference on Machine Learning*. Springer, 1998, pp. 137–142. 91

[129] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999. 91

[130] K.-j. Kim, "Financial Time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003. 91

[131] Y. Liu, F. Wu, Y. Zhuang, and J. Xiao, "Active post-refined multimodality video semantic concept detection with tensor representation," in *Proceedings of the 16th ACM International Conference on Multimedia*. ACM, 2008, pp. 91–100. 91

[132] L.-J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506–1518, 2003. 91

[133] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers & Operations Research*, vol. 32, no. 10, pp. 2513–2522, 2005. 91

[134] R. Rosillo, J. Giner, and D. de la Fuente, "The effectiveness of the combined use of VIX and support vector machines on the prediction of S&P 500," *Neural Computing and Applications*, vol. 25, no. 2, pp. 321–332, 2014. 91

[135] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999. 91, 92, 95

[136] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999. 91, 98

[137] M. M. Rounaghi and F. N. Zadeh, "Investigation of market efficiency and financial stability between S&P 500 and London Stock Exchange: Monthly and yearly forecasting of time series stock returns using ARMA model," *Physica A: Statistical Mechanics and its Applications*, vol. 456, pp. 10–21, 2016. 91

[138] A. C. García, T. Chanwimalueang, G. G. Calvi, A. Hemakom, R. M. Ricós, and D. P. Mandic, "Modelling economic stress through financial systemic balance index," pp. 565–569, 2016. 91

[139] B. G. Malkiel and E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *The journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970. 91

[140] J.-B. Yin, T. Li, and H.-B. Shen, "Gaussian kernel optimization: Complex problem and a simple solution," *Neurocomputing*, vol. 74, no. 18, pp. 3816–3822, 2011. 96, 98

[141] J.-B. Lei, J.-B. Yin, and H.-B. Shen, "GFO: A data driven approach for optimizing the Gaussian function based similarity metric in computational biology," *Neurocomputing*, vol. 99, pp. 307–315, 2013. 96

[142] S. R. Bentes and S. Vaz, "Relationship between implied volatility and the underlying in major international stock markets," *International Journal of Academic Research*, vol. 5, no. 6, 2013. 97

[143] G. Grudnitski and L. Osburn, "Forecasting S&P and gold futures prices: An application of neural networks," *Journal of Futures Markets*, vol. 13, no. 6, pp. 631–643, 1993. 97

[144] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 2005, pp. 625–632. 98

[145] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015. 105

[146] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649. 105, 127

[147] D. P. Mandic and J. Chambers, *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001. 105, 127, 130

[148] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "IMAGENET classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105. 105

[149] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 105, 107, 125

[150] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "IMAGENET large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 106

[151] Z. Zhong, F. Wei, Z. Lin, and C. Zhang, "ADA-Tucker: Compressing deep neural networks via adaptive dimension adjustment Tucker decomposition," *Neural Networks*, vol. 110, pp. 104–115, 2019. 106, 127

[152] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998. 107, 120

[153] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017. 107, 122

[154] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009. 107, 123

[155] J. R. Magnus and H. Neudecker, "Matrix Differential Calculus with Applications to Simple, Hadamard, and Kronecker Products," *Journal of Mathematical Psychology*, vol. 29, no. 4, pp. 474–492, 1985. 108, 109, 110, 148

[156] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 127

[157] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014. 127

[158] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017. 127

[159] T. Lei, Y. Zhang, and Y. Artzi, "Training RNNs as fast as CNNs," *arXiv:1709.02755*, 2017. 127

[160] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016. 127

[161] Z. He, S. Gao, L. Xiao, D. Liu, H. He, and D. Barber, "Wider and deeper, cheaper and faster: Tensorized LSTMS for sequence learning," *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1–11, 2017. 127

[162] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3891–3900, 2017. 128, 130

[163] R. Yu, S. Zheng, A. Anandkumar, and Y. Yue, "Long-term forecasting using tensor-train RNNs," *arXiv preprint arXiv:1711.00073*, 2017. 128

[164] D. Shah, H. Isah, and F. Zulkernine, "Stock market analysis: A review and taxonomy of prediction techniques," *International Journal of Financial Studies*, vol. 7, no. 2, p. 26, 2019. 128

[165] V. Khrulkov, A. Novikov, and I. Oseledets, "Expressive power of recurrent neural networks," *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 133

[166] H. Huang, "Asymptotic behavior of support vector machine for spiked population model," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1472–1492, 2017. 140

[167] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012. 140

[168] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, and A. G. Constantinides, "Graph signal processing–Part II: Processing and analyzing signals on graphs," *arXiv preprint arXiv:1909.10325*, 2019. 140

[169] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, S. Li, and A. G. Constantinides, "Graph signal processing–Part III: Machine learning on graphs, from graph topology to applications," *arXiv preprint arXiv:2001.00426*, 2020. 140

[170] B. S. Dees, L. Stankovic, M. Dakovic, A. G. Constantinides, and D. P. Mandic, "Unitary shift operators on a graph," *arXiv preprint arXiv:1909.05767*, 2019. 143

[171] J. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Transactions on Circuits and Systems*, vol. 25, no. 9, pp. 772–781, 1978. 148

[172] K. B. Petersen, M. S. Pedersen *et al.*, "The matrix cookbook," *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008. 148

[173] C. F. Van Loan and N. Pitsianis, "Approximation with Kronecker products," in *Linear Algebra for Large Scale and Real-Time Applications*. Springer, 1993, pp. 293–314. 149